**TASK 2: Vulnerability Assessment and Penetration Testing (VAPT) Report**

**Project Title:** Vulnerability Assessment and Penetration Testing of<https://testphp.vulnweb.com>

**Date:** June 2025

## **1. Executive Summary**

This report details a comprehensive Vulnerability Assessment and Penetration Testing (VAPT) performed on https://testphp.vulnweb.com, an intentionally vulnerable web application hosted by Acunetix. The purpose was to identify security weaknesses, demonstrate exploitability, and recommend remediation strategies. The tests covered OWASP Top 10 vulnerabilities, authentication flaws, session management issues, and business logic bypasses using both automated and manual methods.

## 

## **2. Methodology Overview**

The testing followed a structured VAPT methodology:

1. Pre-engagement activities
2. Information gathering (reconnaissance)
3. Vulnerability assessment
4. Manual penetration testing
5. Post-authentication testing
6. Business logic and session testing
7. Reporting

## **3. Pre-engagement Activities**

### **3.1 Scope and Boundaries**

* **Target Domain:**<https://testphp.vulnweb.com>
* **Environment:** Publicly accessible vulnerable test application (no production impact)
* **Rules of Engagement:** Non-destructive testing only; no data modification
* **Authorization:** Platform is provided by Acunetix for ethical testing and educational purposes

### **3.2 Tools and Environment**

* **Operating System:** Kali Linux VM
* **Tools Used:**
* **Nmap** – for port and service scanning
* **WhatWeb** – for fingerprinting technologies
* **Nikto** – for discovering web server misconfigurations
* **SQLMap** – for testing SQL injection
* **Burp Suite** – for intercepting traffic and testing input/output vulnerabilities like XSS and Session Fixation
* **Gobuster** – for directory brute-forcing
* **FFUF (**Directory Bruteforcing)
* **Chrome DevTools**
* **Python**
* **Firefox Developer Tools :** for testing business logic flaws, session cookies, and file upload

## **4. Information Gathering (Reconnaissance)**

### **4.1 Passive Reconnaissance**

* Identified web technologies using browser plugins (Wappalyzer).
* Observed comments, hidden form fields, and page behavior.

### **4.2 Active Reconnaissance**

* **Directory Bruteforcing:** Performed with FFUF to discover hidden paths and resources.
* **User-Agent and Referrer analysis:** Conducted using Burp Suite.

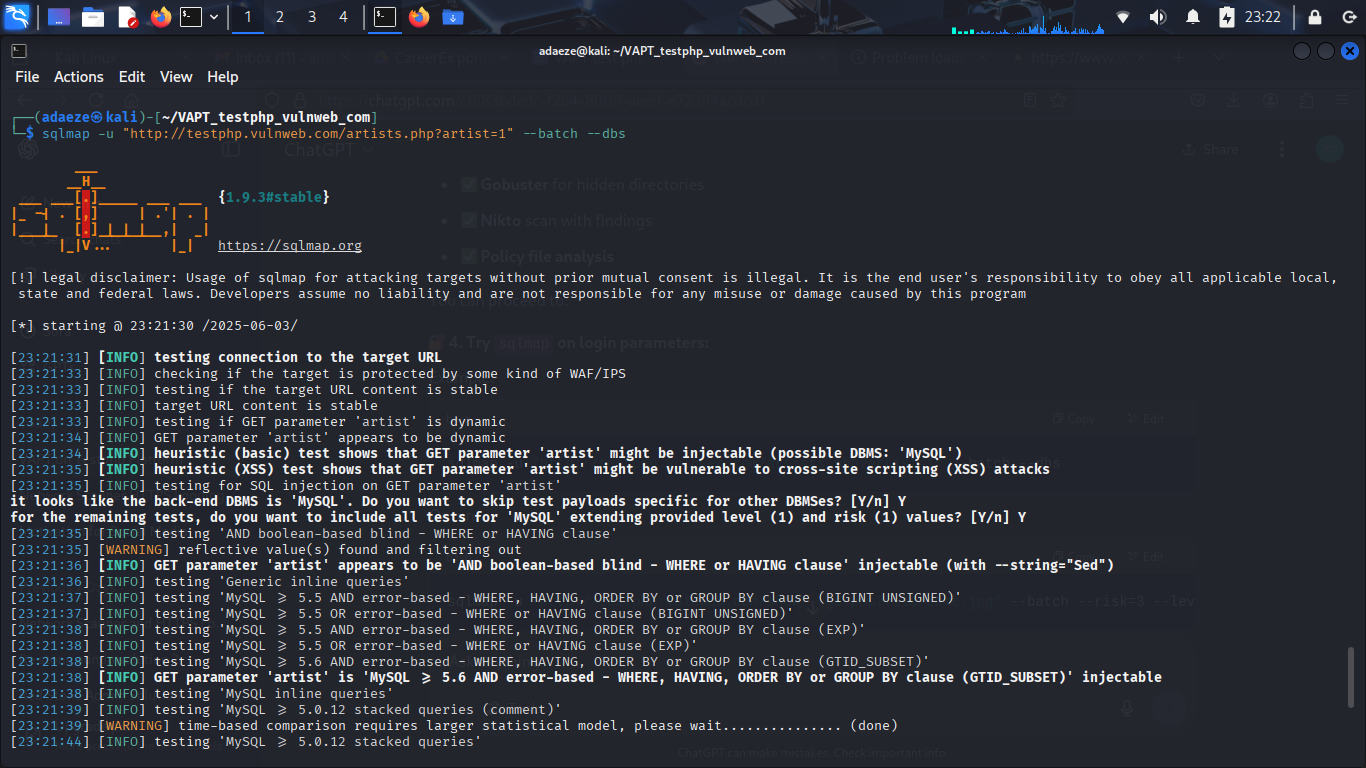
## **5. Vulnerability Assessment**

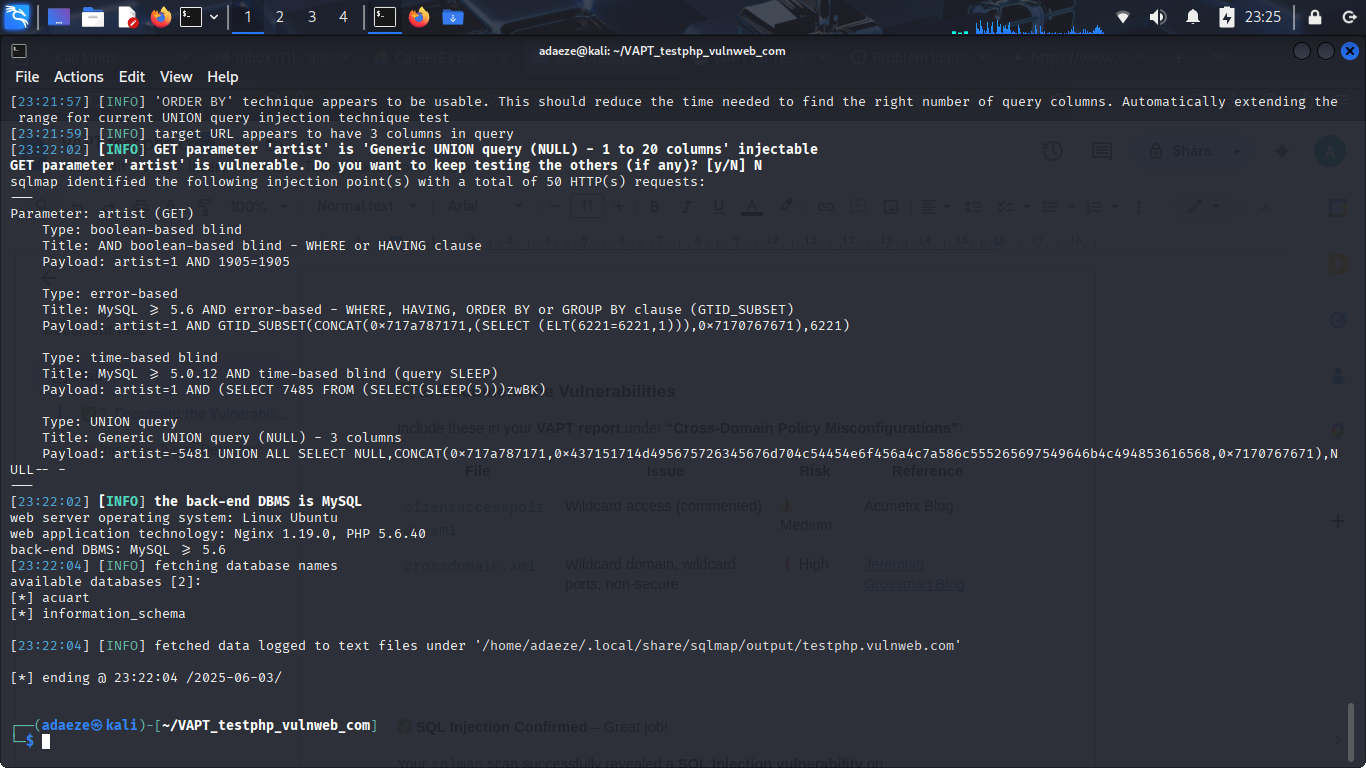
### **Automated Scans Performed:**

* **SQL Injection:** SQLmap revealed injectable parameters in login and other forms.
* **Tools**: SQLMap, Burp Suite
* **Payload**: testphp.vulnweb.com/artists.php?artist=1' OR '1'='1
* **Impact**:  
  1. Unauthorized access to backend databases; potential for data       exfiltration and modification.

   2.Gave access to the backend database and user credentials.

* **Evidence**:  
   SQLMap successfully dumped database names and user credentials.
* **Fix Recommendation**:  
  1. Use **prepared statements / parameterized queries  
  2.**Apply input validation and escaping  
  3.Implement **Web Application Firewall (WAF)** for query filtering
* **Demonstration**: Show SQLMap dump or DB enumeration screenshot.





* **Impact**: Gave access to backend database and user credentials.

**Cross-Site Scripting (XSS):** Stored XSS found on guestbook page.

* **Location**: Profile Update Form, Guestbook
* **Tool Used**: Burp Suite, Browser
* **Payload**: <script>alert('XSS')</script>

Injected in Guestbook and Profile Update forms.

* **Impact**:

1. Could lead to cookie theft or session hijacking.
2. Attackers can hijack sessions, deface content, or redirect users to malicious sites.

* **Fix Recommendation**:

1.Sanitize all user inputs (strip <script>, onload, etc.)

2.Escape output content using context-aware escaping (HTML, JS)

3.Implement **Content Security Policy (CSP)** headers

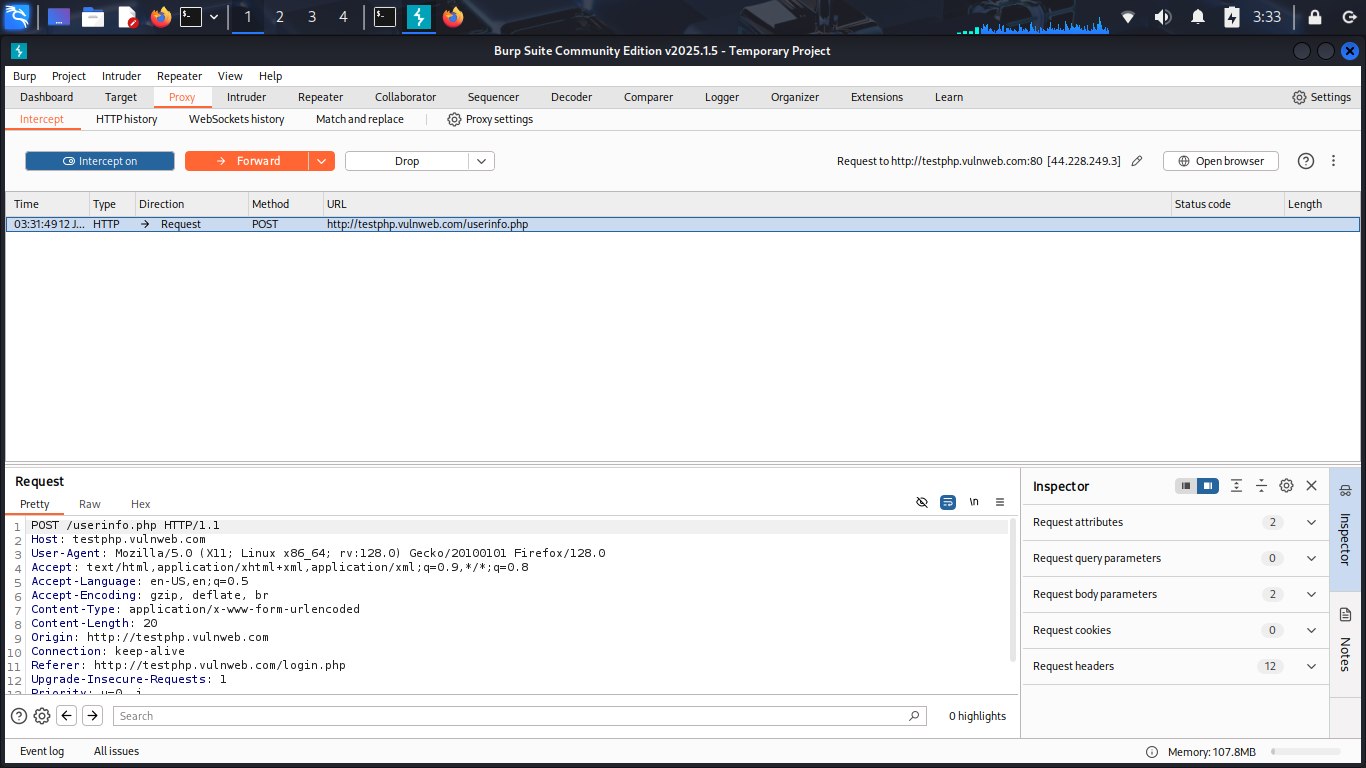
Demonstration;



**6. Manual Penetration Testing**

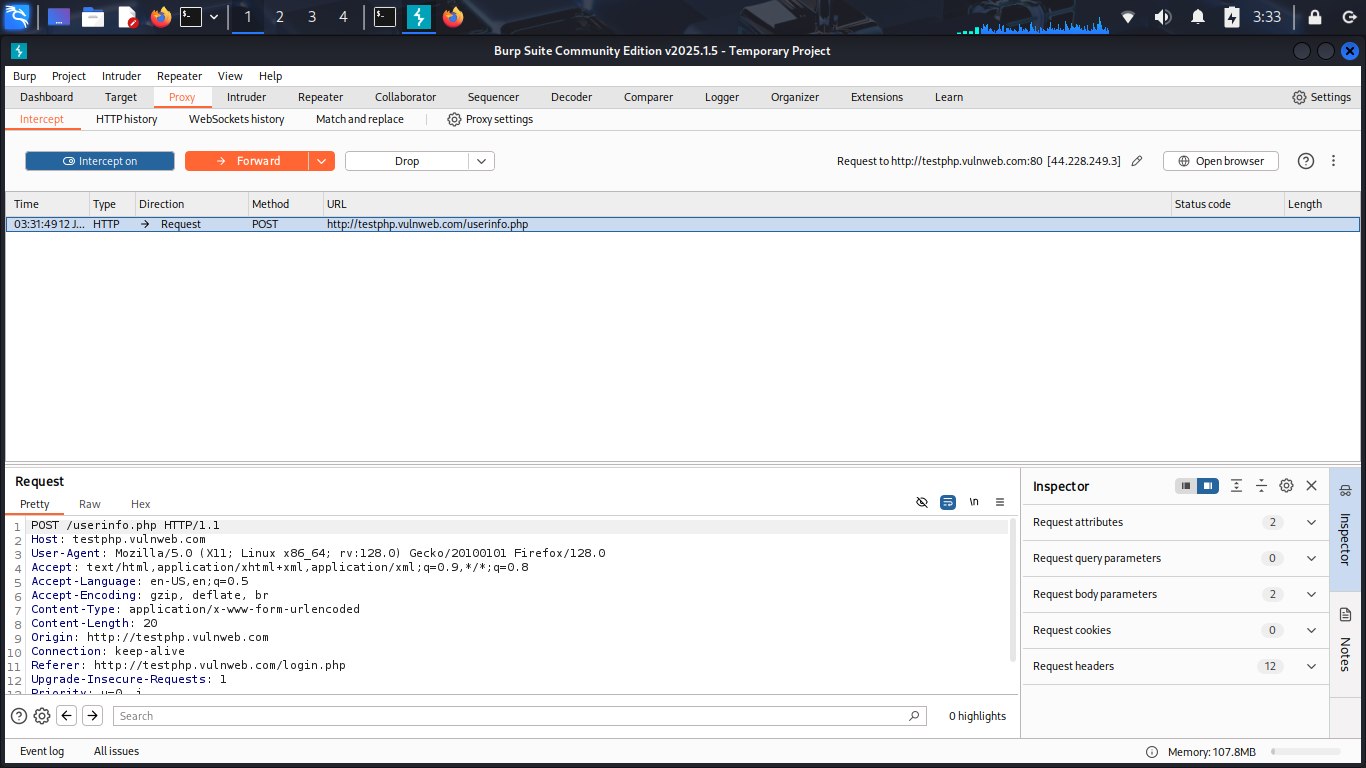
### **6.1 Authentication Testing**

* **Login Bypass:** Manual SQL injection tested using ' OR 1=1-- in login fields.
* **Session Cookies:** Observed lack of Secure and HttpOnly flags.
* **Session Fixation:** Session ID remained static across login/logout.



#### **Session Fixation**

* **Tools**: Burp Suite + Firefox Private Browser
* **Demo**: Show how adding a token (login=attackertoken123) to cookie allows access.



* **Impact**:

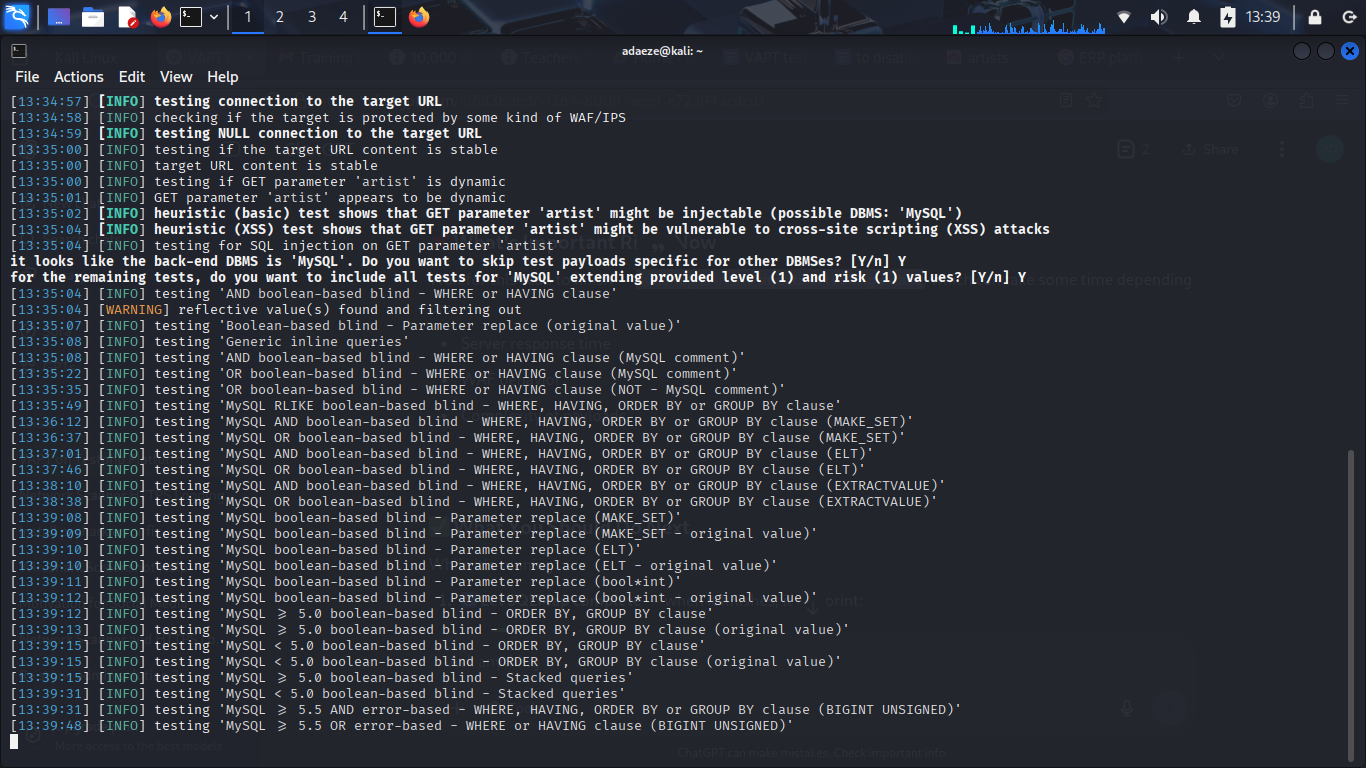
1.Reusing session ID allowed unauthorized access.

2.Allows attackers to force a user to use a known session ID, leading to session hijack.

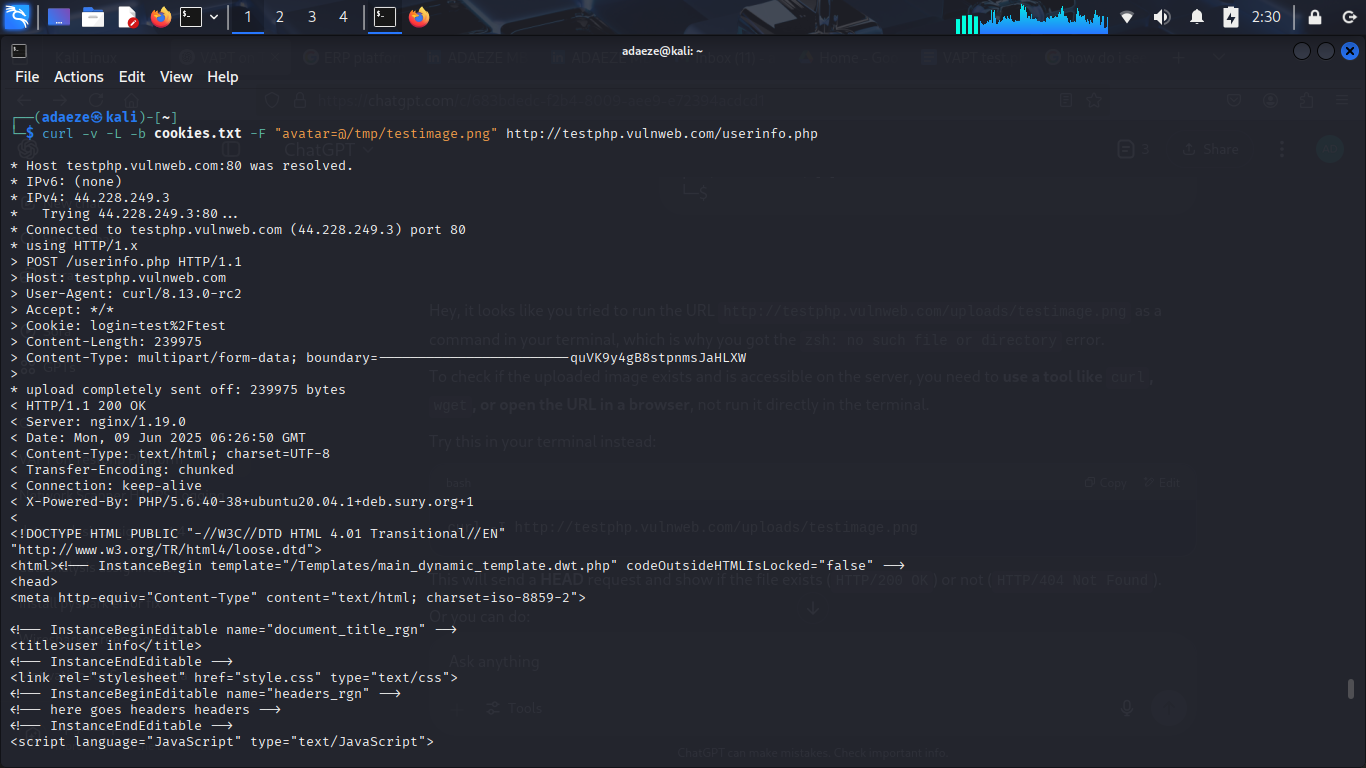
* **Fix Recommendation**:  
  + Always **regenerate session ID upon login**
  + Set HttpOnly, Secure, and SameSite flags on cookies
  + Implement short session timeout

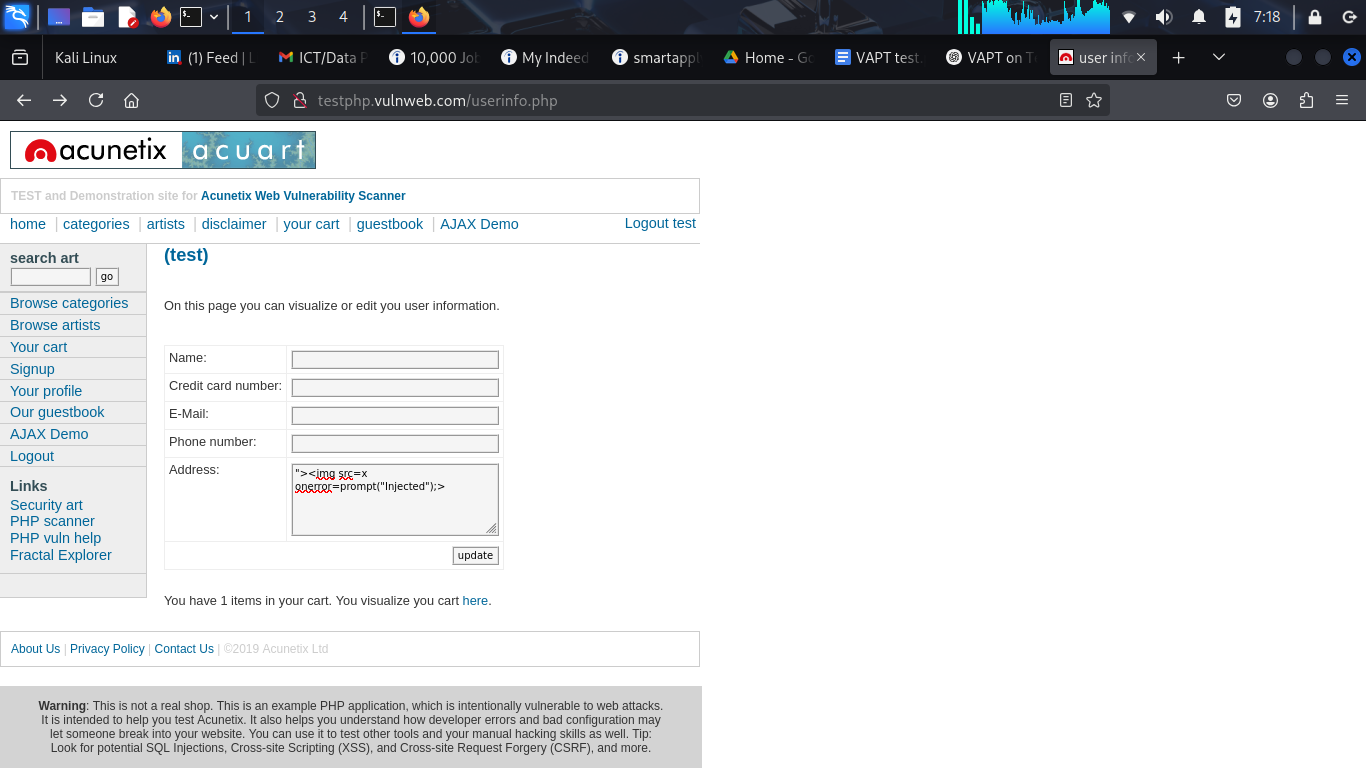
### **6.2 Input Validation Testing**

* **SQL Injection:**
  + Payload used: ' OR 1=1--
  + SQLmap confirmed injection point
* **XSS:**
  + Payload: <script>alert('XSS')</script> successfully executed in browser.









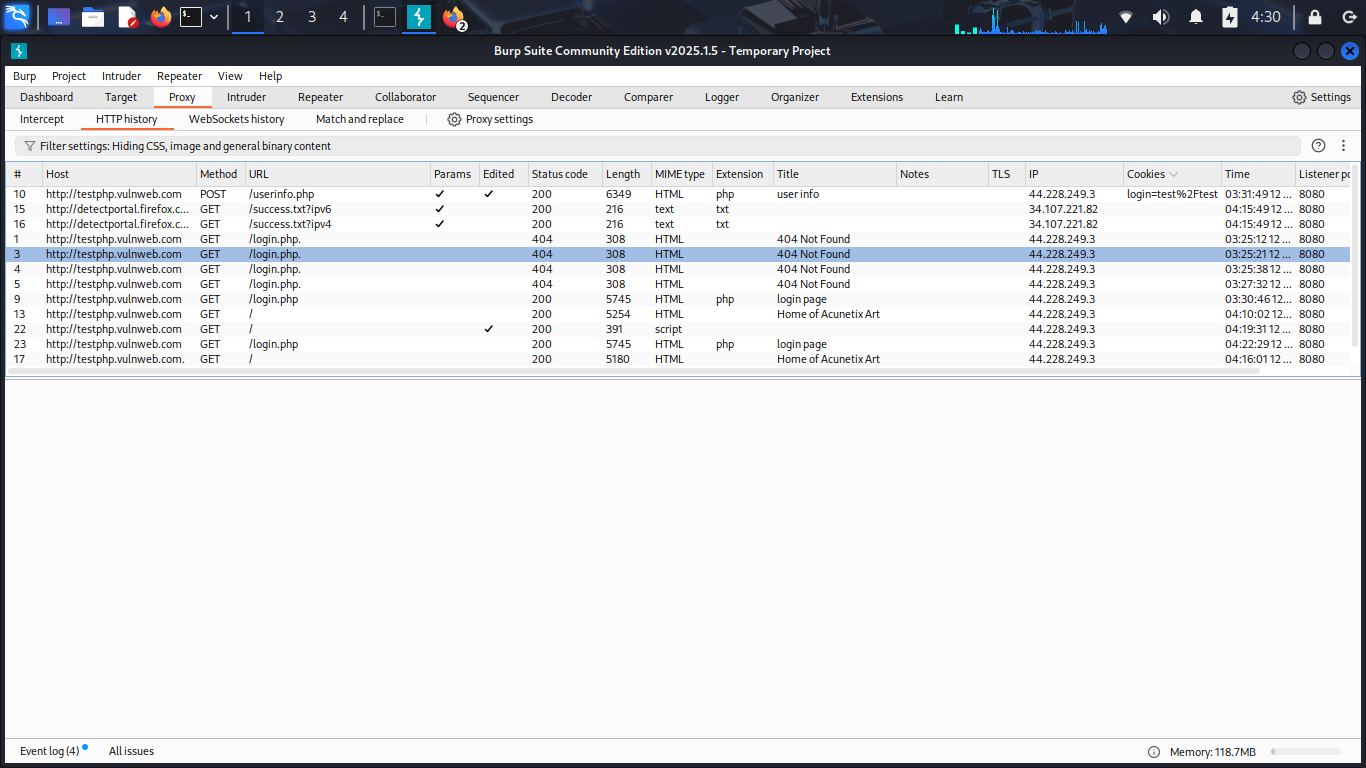
## **7. Post-Authentication Testing**

### **7.1 Role-Based Access Control (RBAC)**

* No apparent user roles, but no access control mechanisms observed.

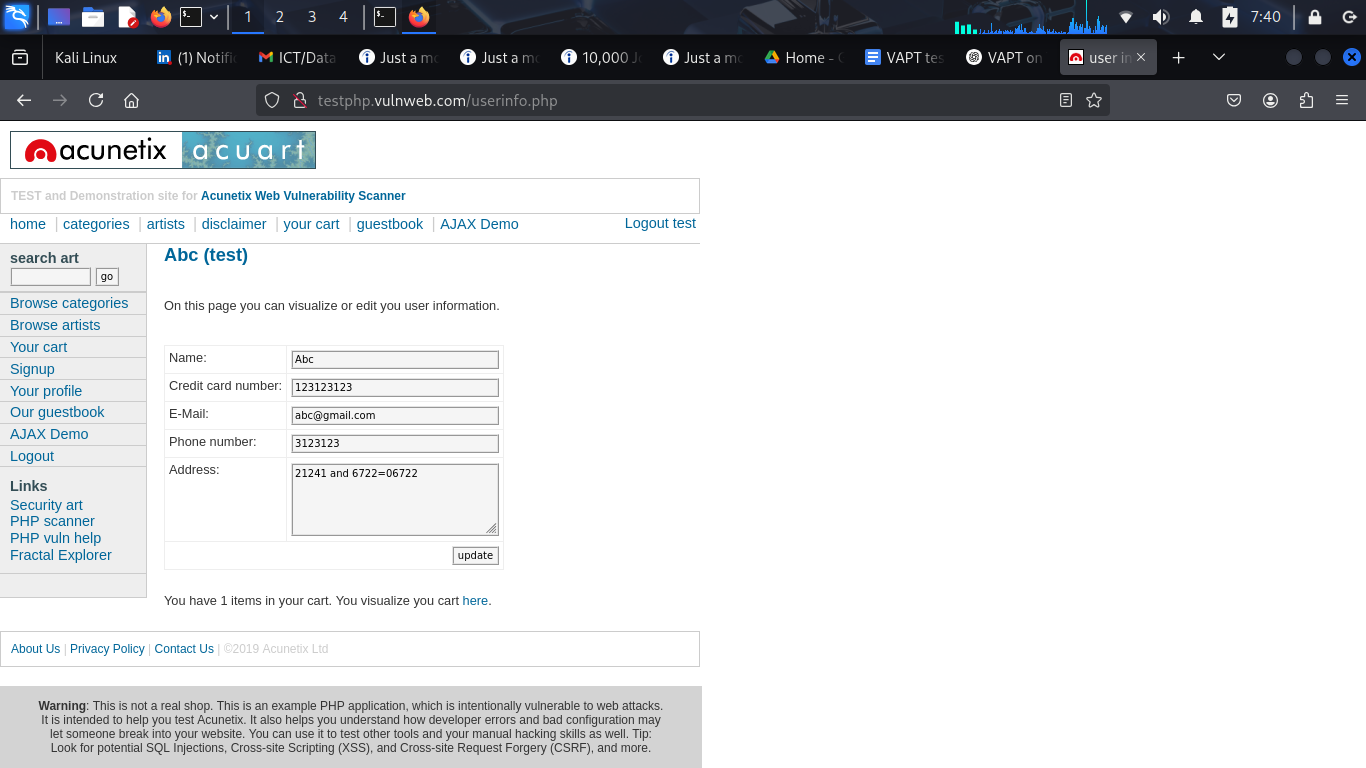
### **7.2 Insecure Direct Object Reference (IDOR)**

* URL manipulation tested (e.g., /profile?id=1) but app returned same data — no IDOR confirmed.



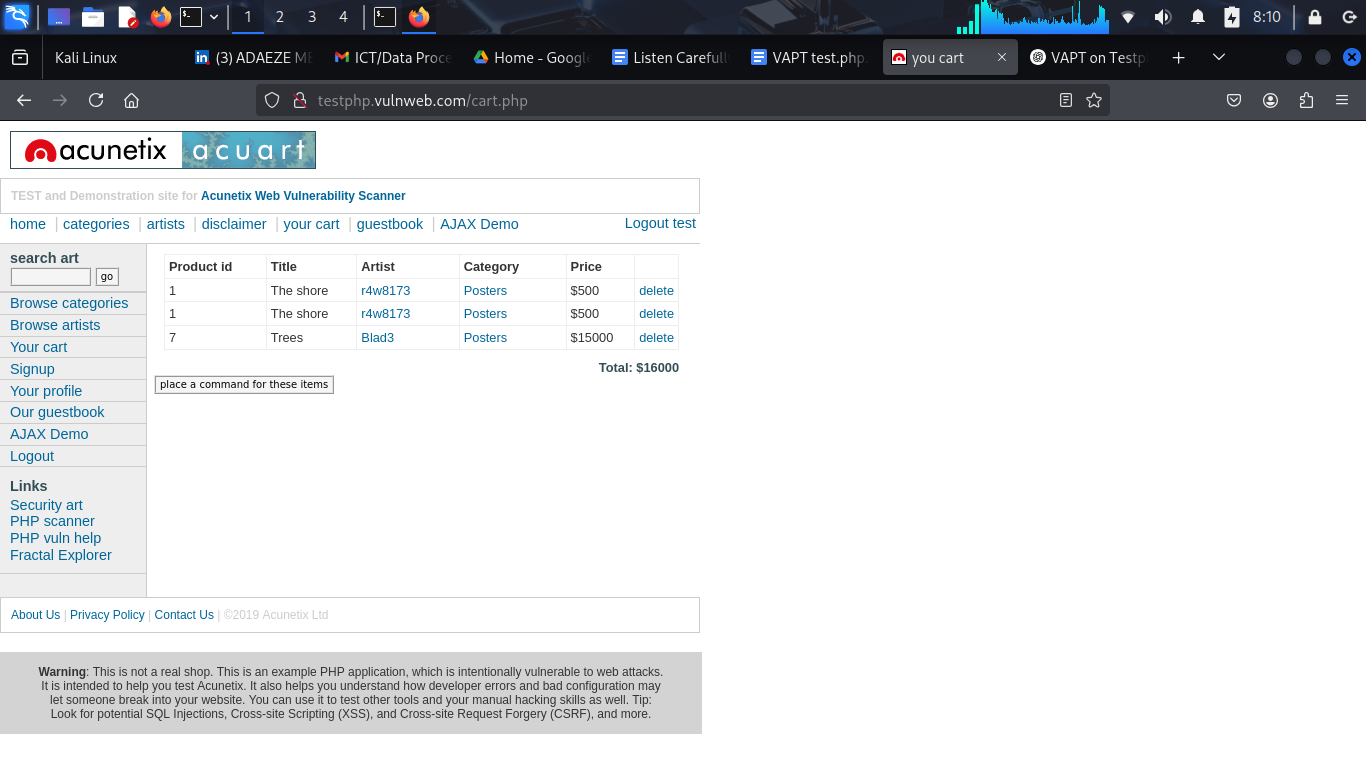
### **7.3 File Upload Handling**

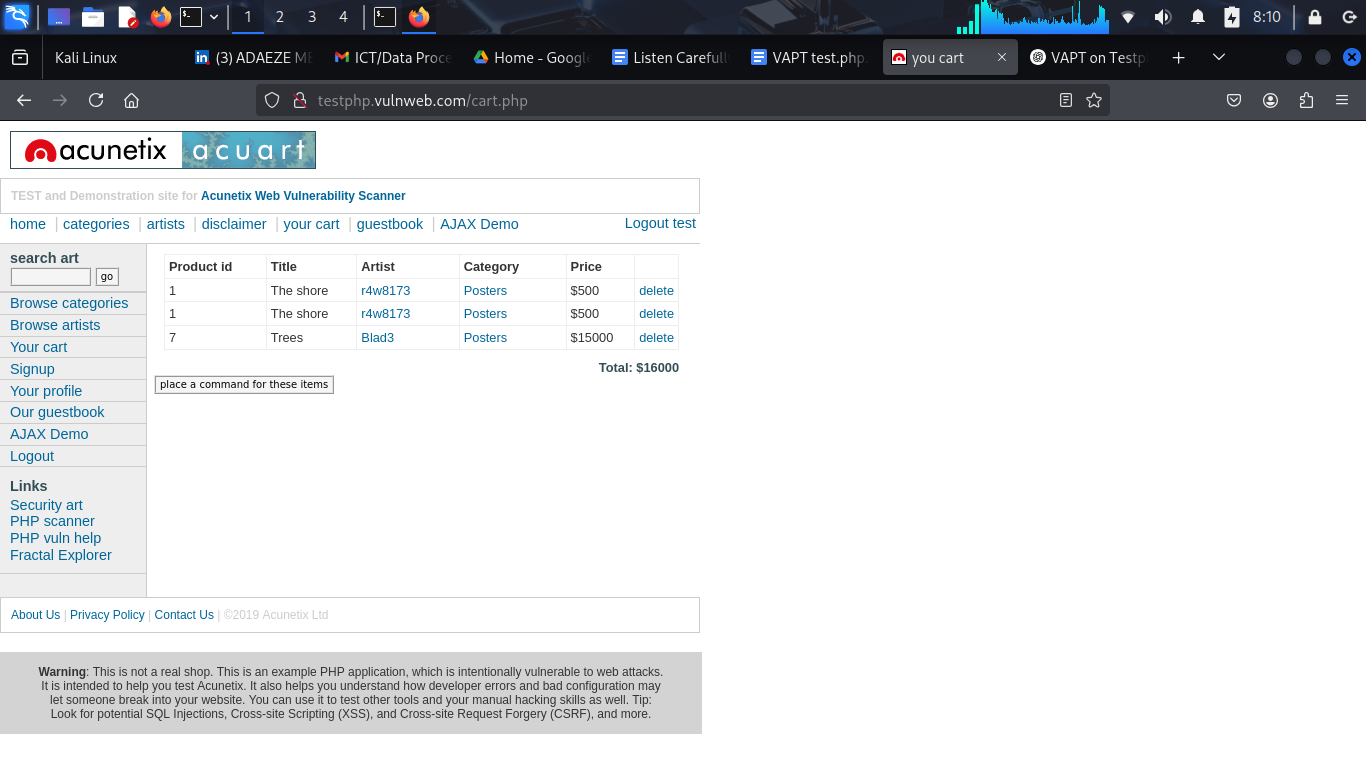
* File upload feature allowed .php.jpg upload, but no execution observed. Possible bypass vulnerability.

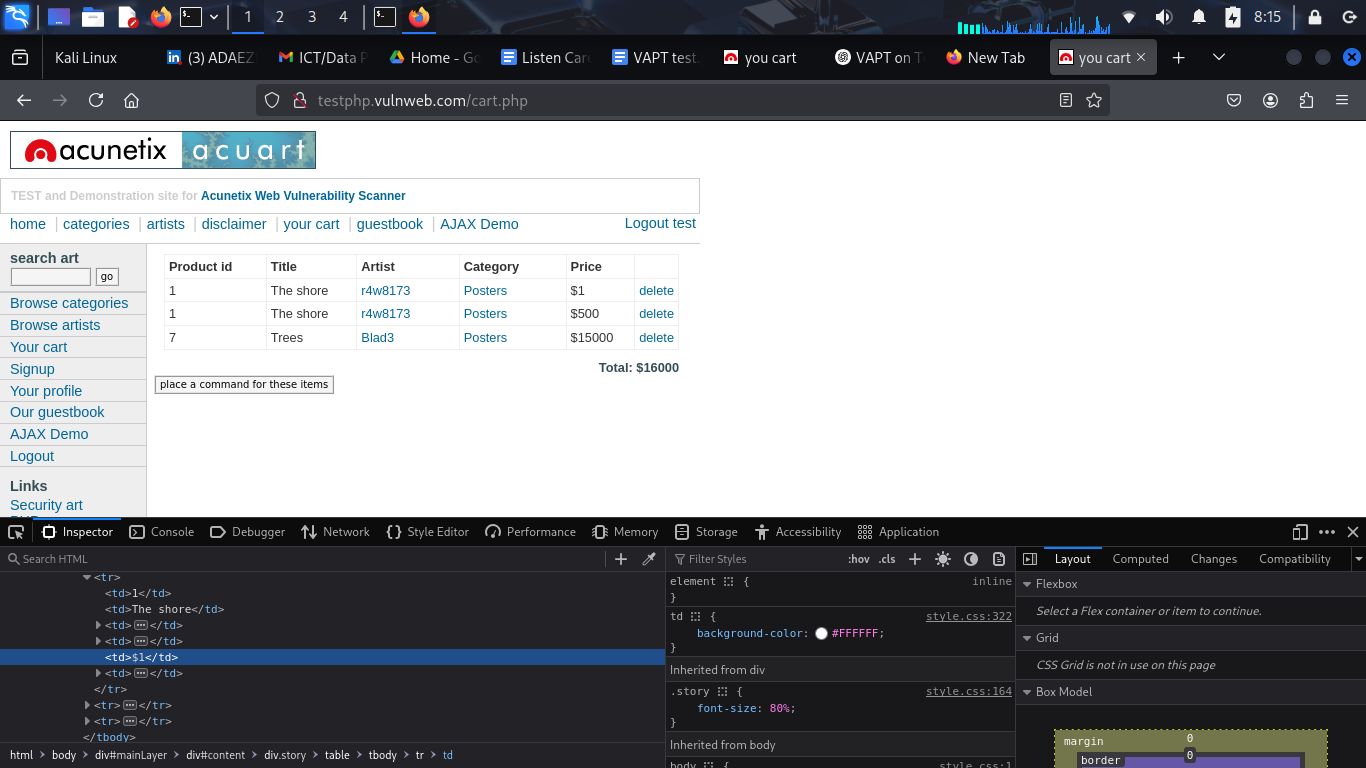


## **8. Business Logic Testing**

* **Cart Manipulation:** Successfully altered item prices in cart via DevTools.
* **Workflow Abuse:** Able to check out items with modified pricing logic.







* **Impact**:

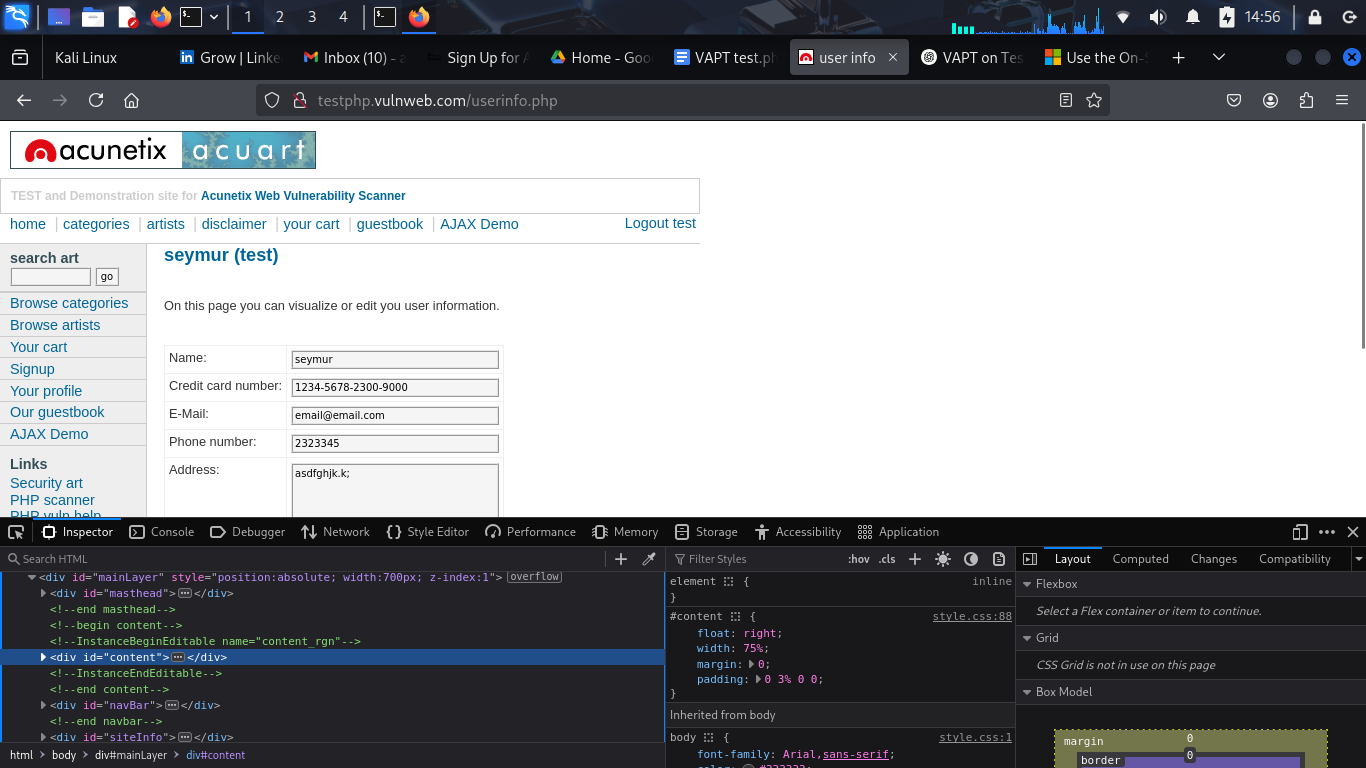
1.Insecure client-side validation.

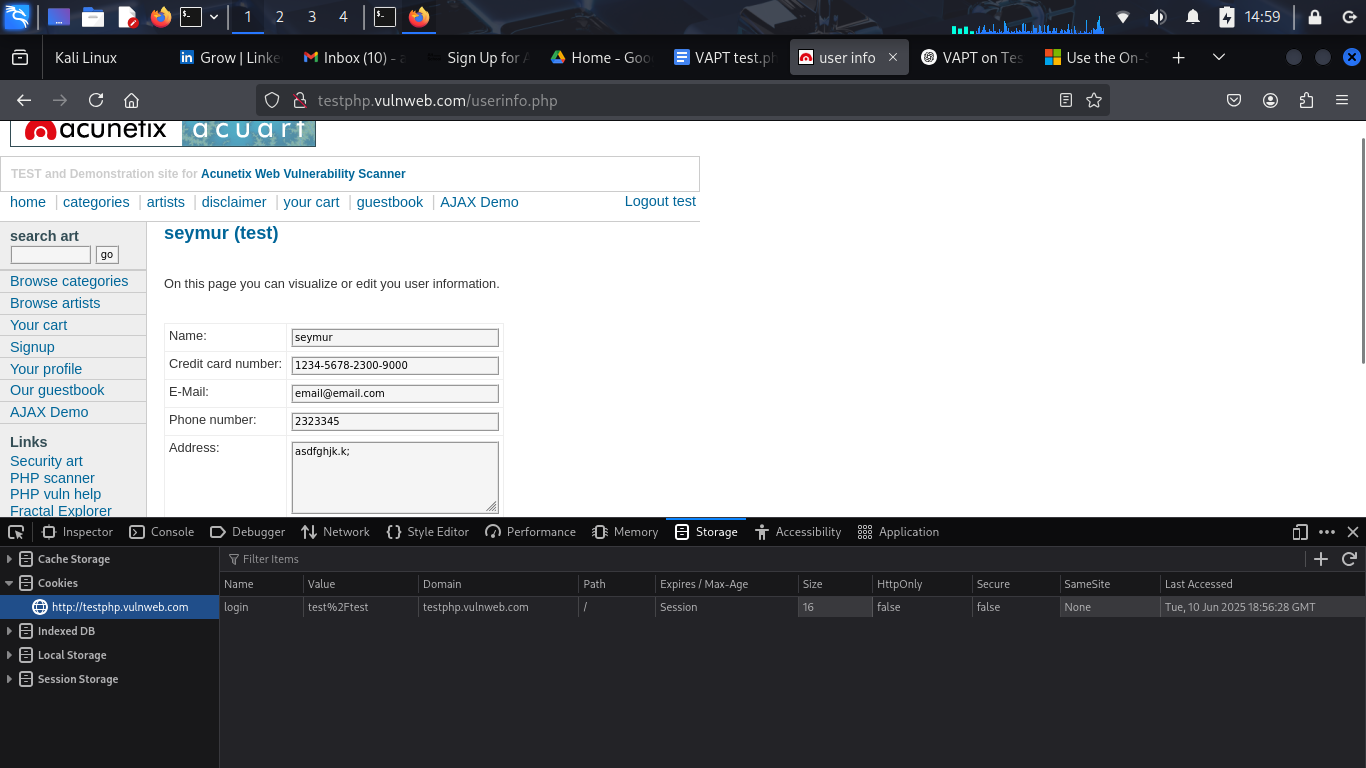
2.Bypassed pricing validation; attacker can manipulate transaction values.

* **Fix Recommendation**:  
  + Move **price calculations to server-side**
  + Do not trust client-side data for critical transactions
  + Use digital signature or HMAC for cart items to detect tampering

## **9. Session Management Testing**

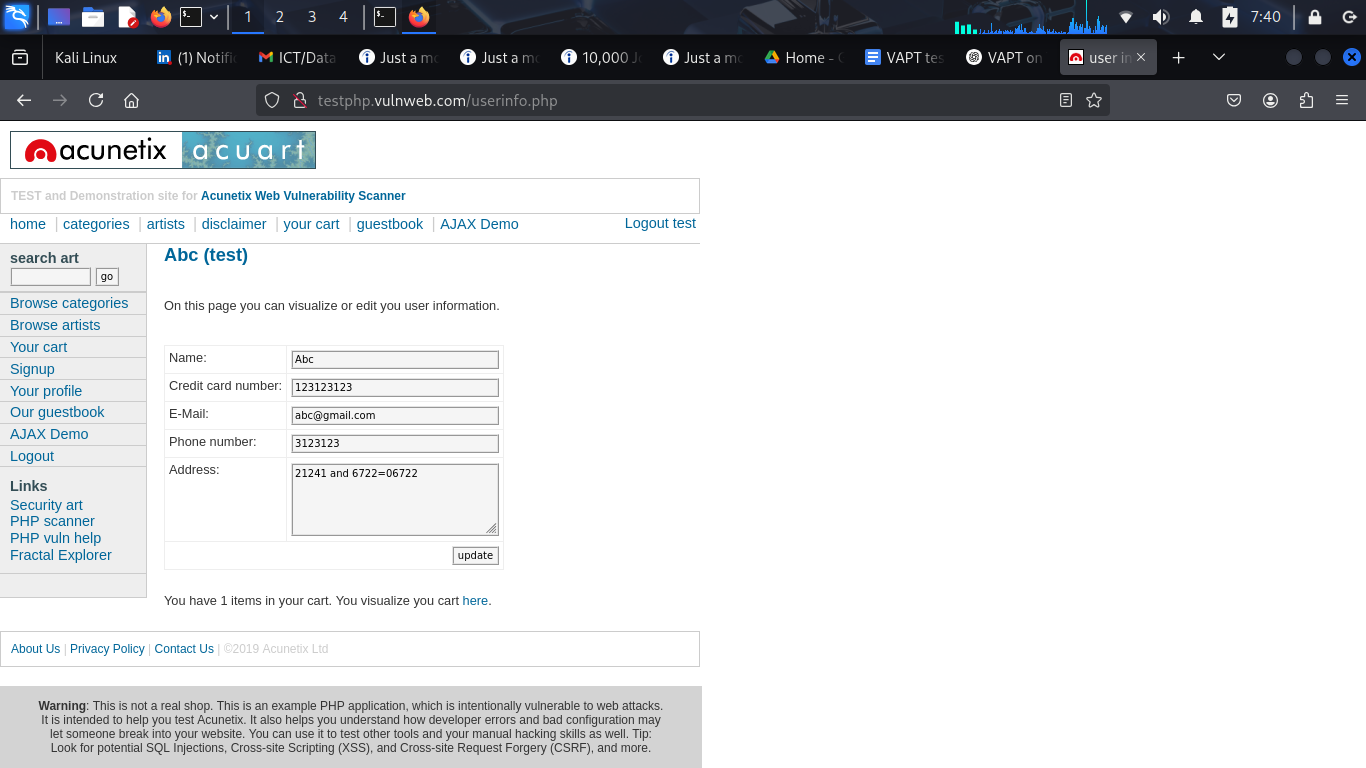
* **Weak Session ID:** Observed low-entropy session ID format.
* **No Expiry on Logout:** Session remained valid post logout.
* **Session Fixation:** Able to reuse session cookie post-authentication.

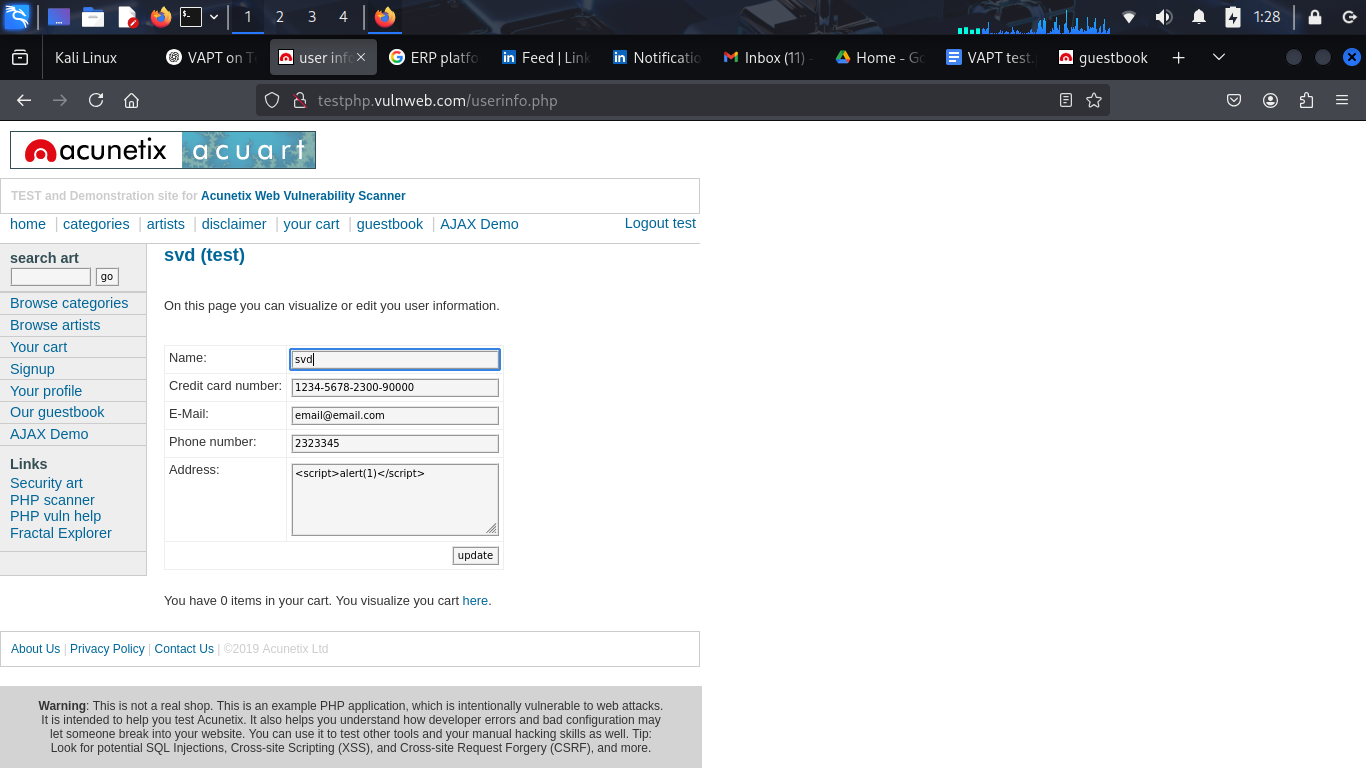




#### **File Upload Testing**

* **Tool Used**: Firefox Developer Tools, Burp Suite
* **Action**: Tried uploading .php or .png with payload
* **Observation**: File upload function is restricted (no feedback), possible hidden endpoint.





## 

## 

* **Impact**: If upload accepted, could be used for RCE (**Remote Code Execution**)

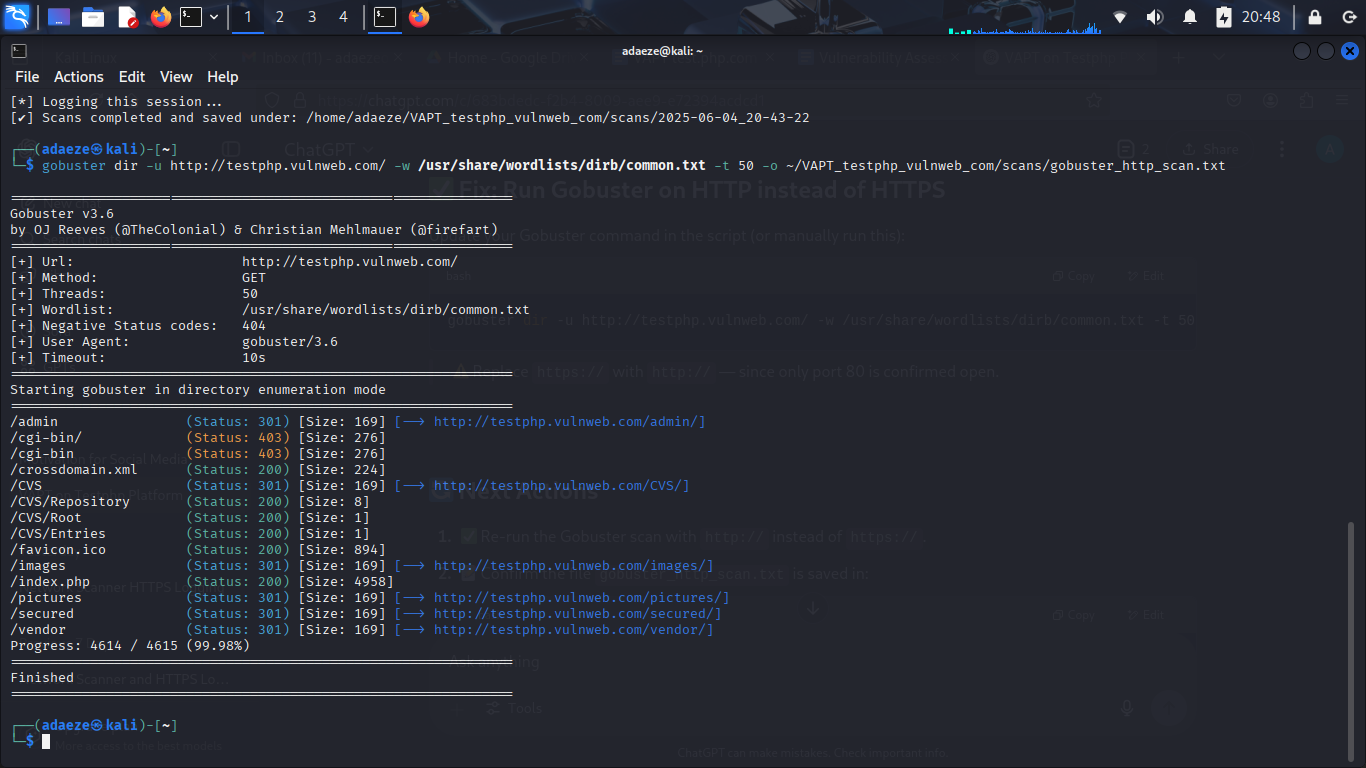
**Remote Code Execution (RCE)** is a **critical vulnerability** that allows an attacker to **execute arbitrary code** on a remote system (server, application, or computer) — **without authorization**.

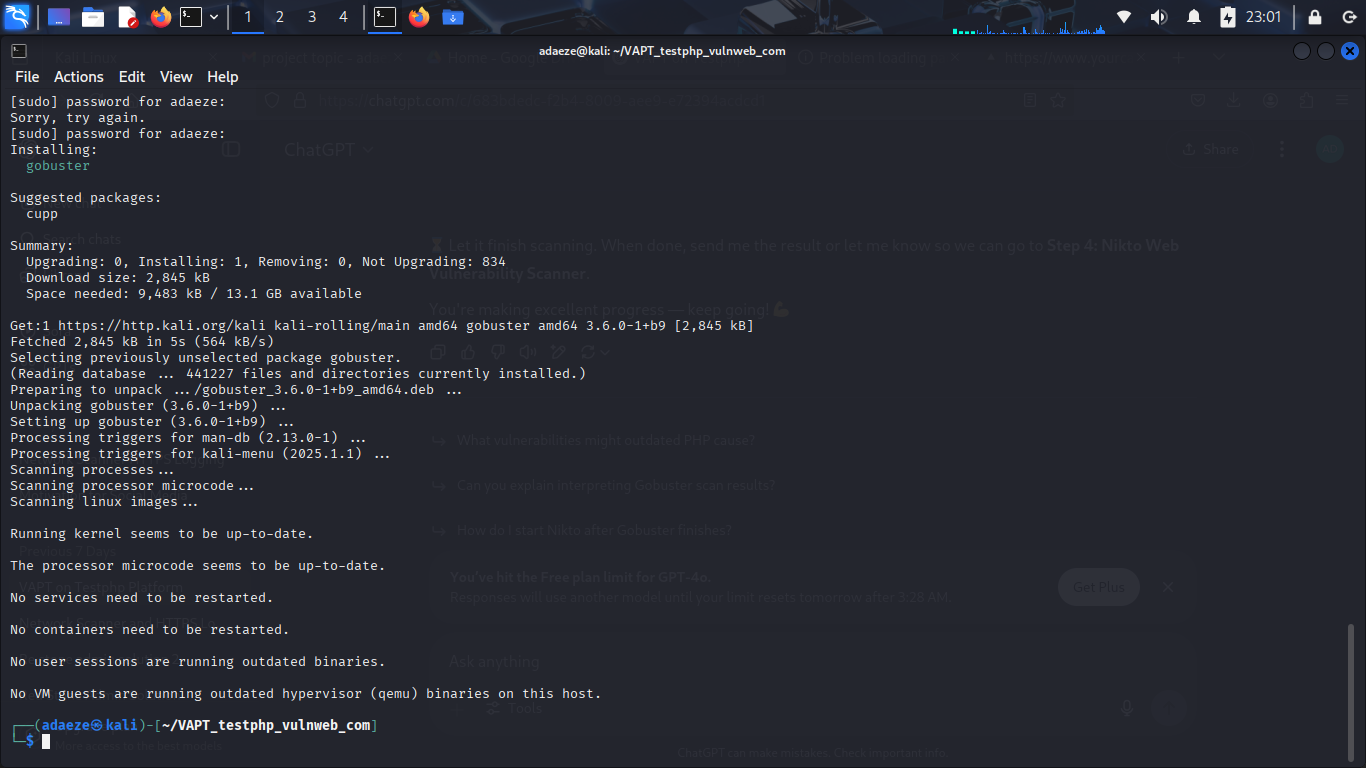
**Fix Recommendation**:

* + Validate file **MIME type and extension**
  + Store uploaded files **outside webroot**
  + Rename files and disable script execution in upload directories

#### **Directory Enumeration**

* **Tool**: Gobuster

**Output**: Found /admin/, /cart/, /CVS/etc.  


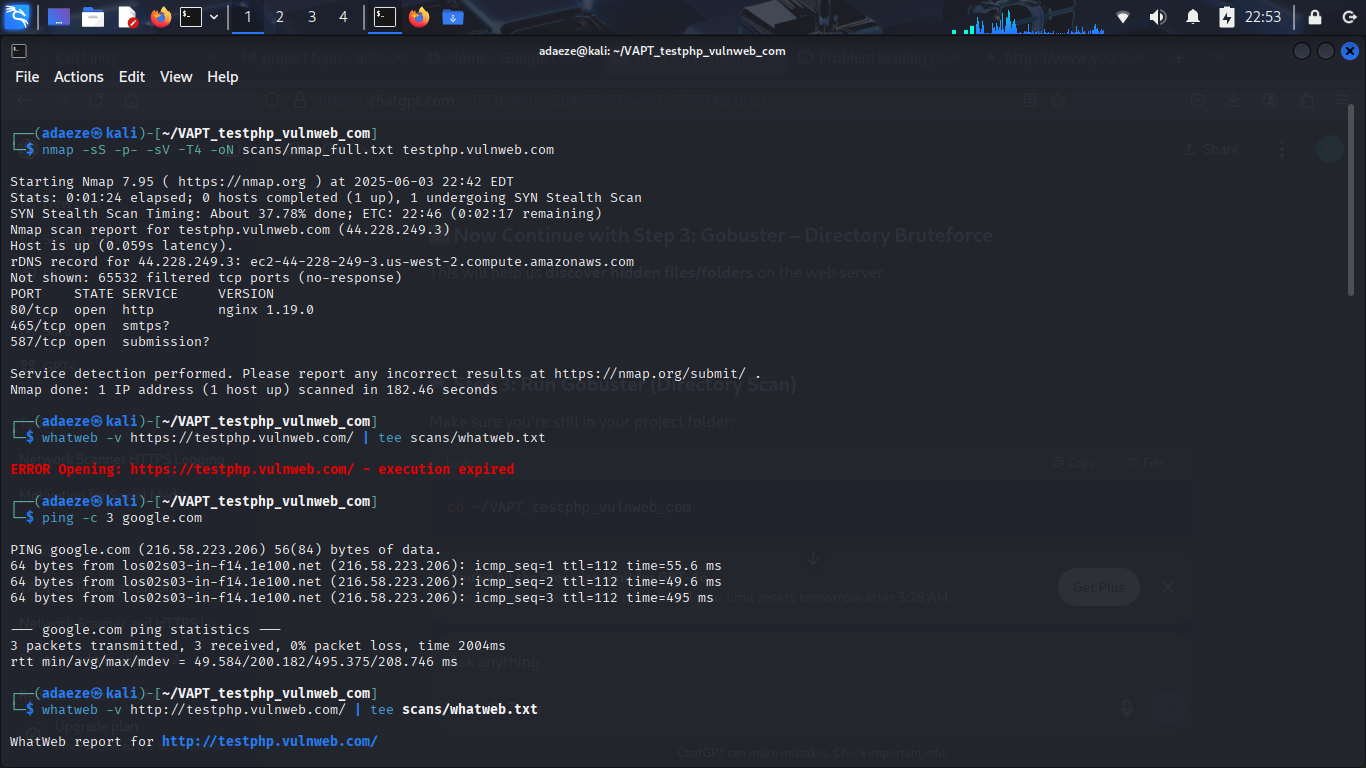


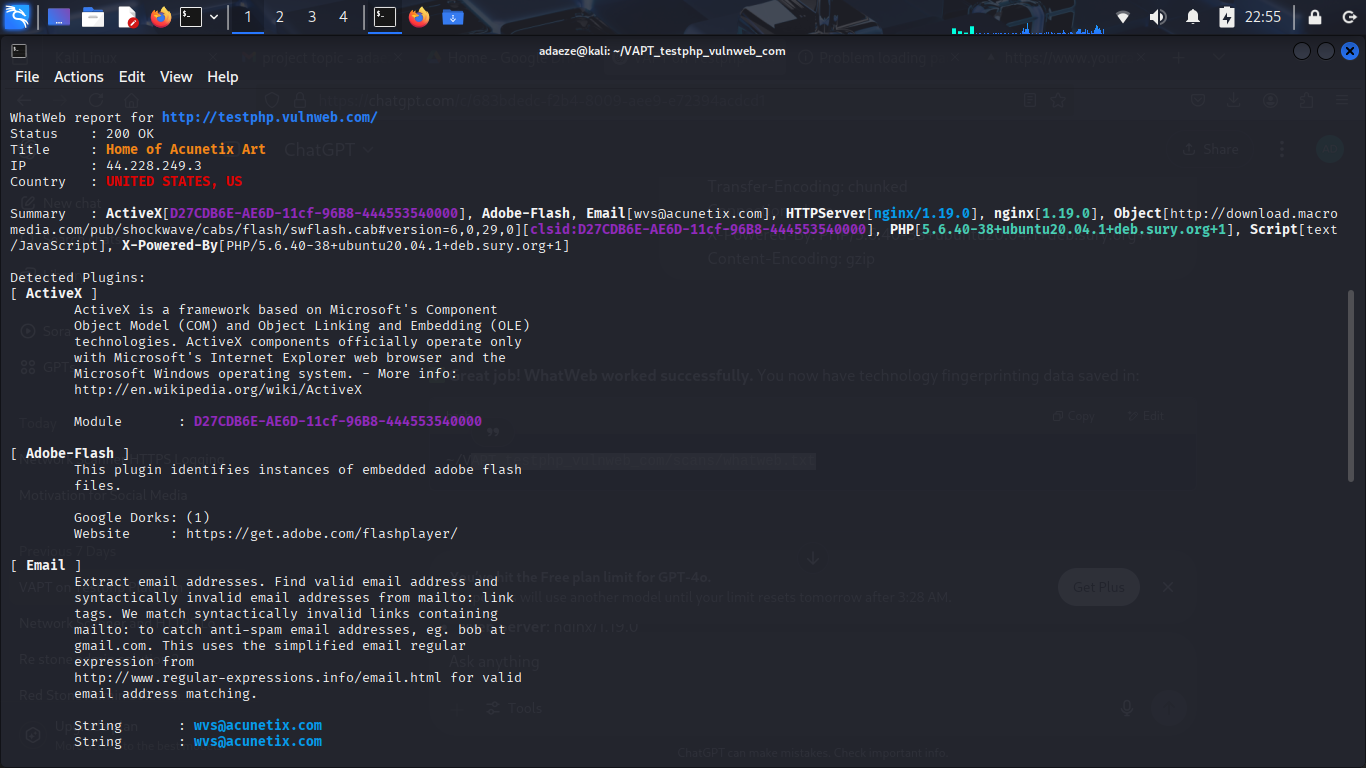
* **Impact**: Unprotected admin directories found.
* Revealed potentially sensitive unprotected directories
* **Fix Recommendation**:  
  + Use **.htaccess** or equivalent to deny directory listings
  + Disable unnecessary directories or apply authentication
  + Review and restrict directory and file permissions

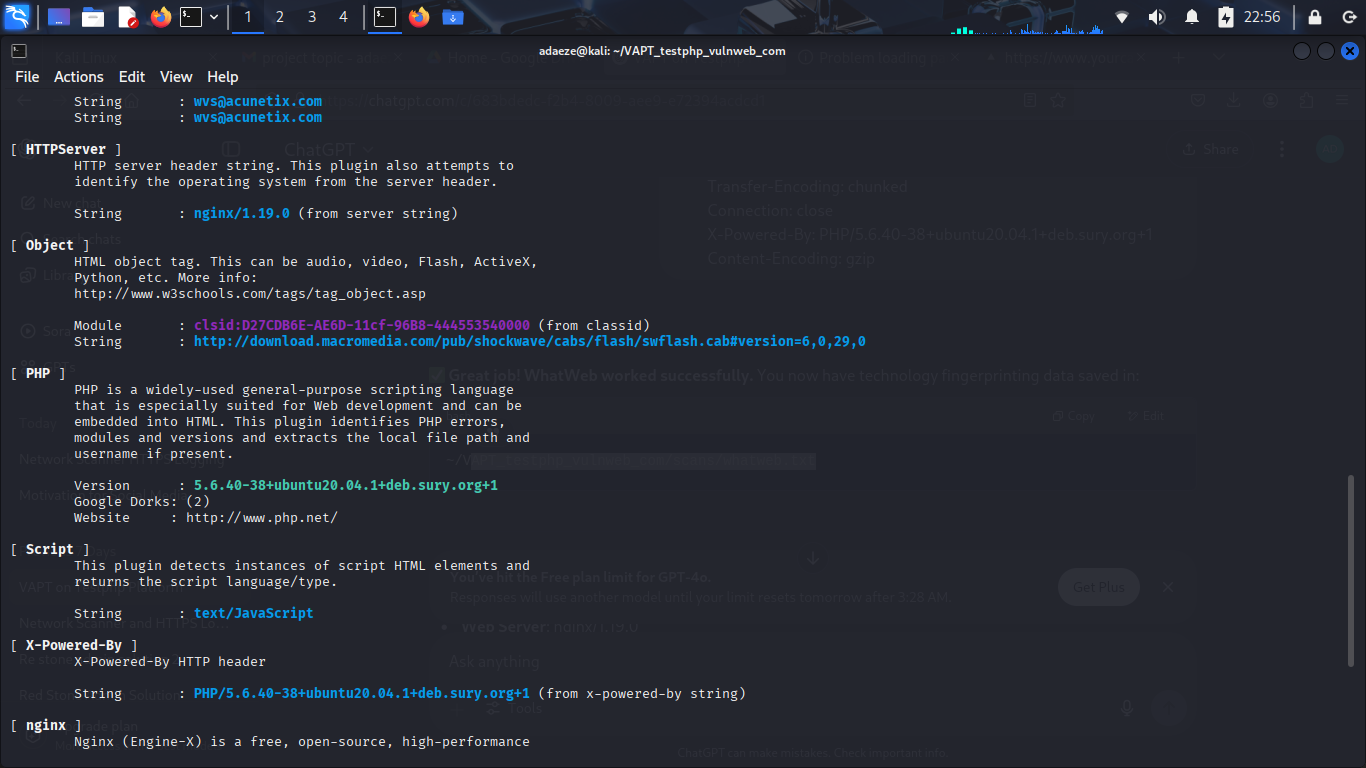
**3.7 Outdated Components**

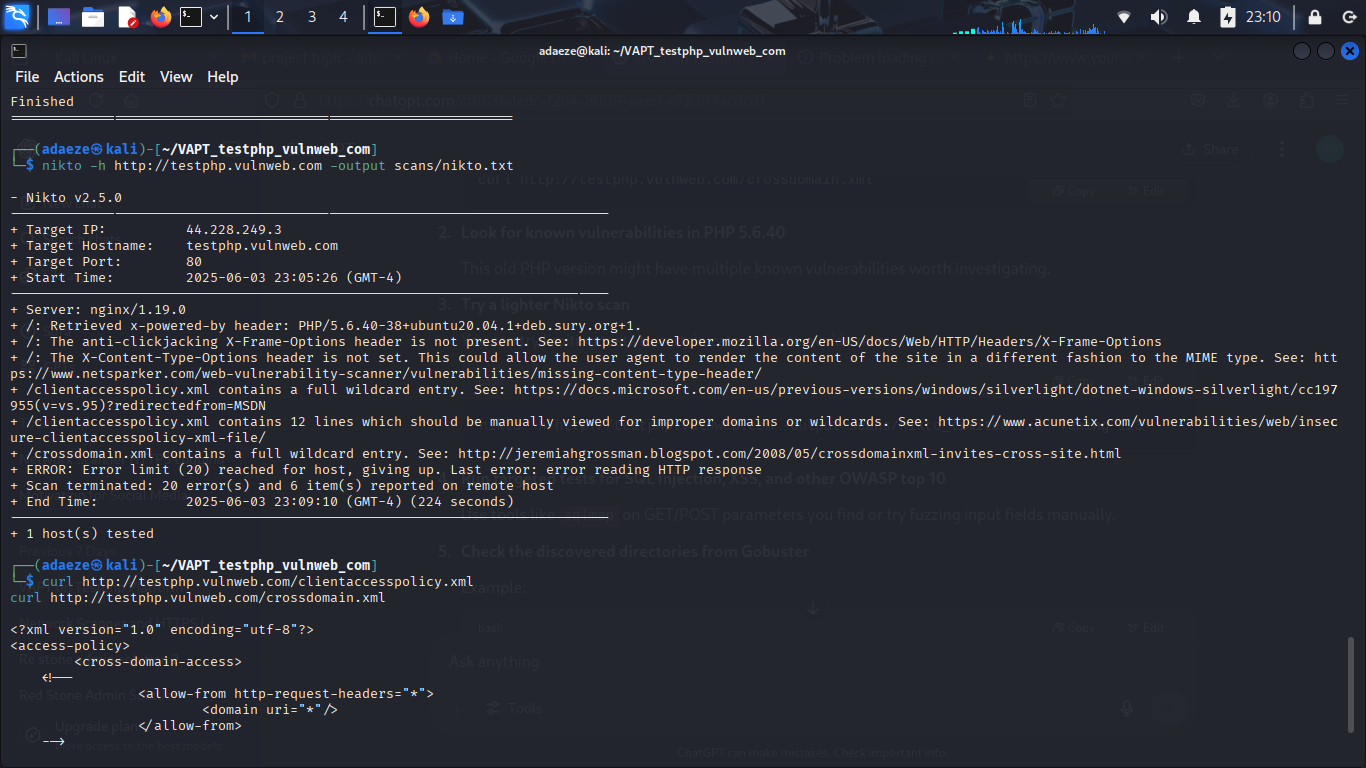
* **Tool**: WhatWeb, Nikto
* **Result**: Detected Apache 2.2.8 and outdated PHP version (5.2.x)

Demonstration:









* **Impact**: Known vulnerabilities may be exploited.

Public vulnerabilities may exist for these components

* **Fix Recommendation**:  
  + Upgrade to supported versions (Apache 2.4+, PHP 8+)
  + Regularly patch and update all dependencies
  + Monitor CVEs for used packages

## **10. Findings Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Severity** | **Status** | **Tool/Method Used** |
| SQL Injection | High | Confirmed | SQLmap, Manual |
| Stored XSS | Medium | Confirmed | Manual (Guestbook) |
| Insecure Cookies | Medium | Confirmed | Burp Suite |
| Session Fixation | Medium | Confirmed | Manual, Cookie Replay |
| Business Logic Flaw | High | Confirmed | DevTools |
| File Upload Bypass Attempt | Medium | Inconclusive | Manual |
| Missing Security Headers | Low | Confirmed | Browser, Burp |
| IDOR | Low | Not Confirmed | Manual |

## **11. Recommendations**

1. **Sanitize Input:** Implement parameterized queries for SQL.
2. **XSS Protection:** Escape output and use CSP headers.
3. **Secure Cookies:** Use HttpOnly and Secure attributes.
4. **Session Handling:** Regenerate session ID on login, expire on logout.
5. **File Upload:** Validate MIME types, restrict extensions.
6. **Access Controls:** Introduce role-based access and IDOR protections.
7. **Header Security:** Add missing security headers.
8. **Business Logic:** Validate price fields server-side, not in frontend.

## **12. Supporting Evidence**

(Screenshots will be attached in the final zipped submission)

* Screenshot of SQLMap execution
* Screenshot of XSS alert
* Screenshot of insecure cookies
* Screenshot of cart manipulation via browser tools

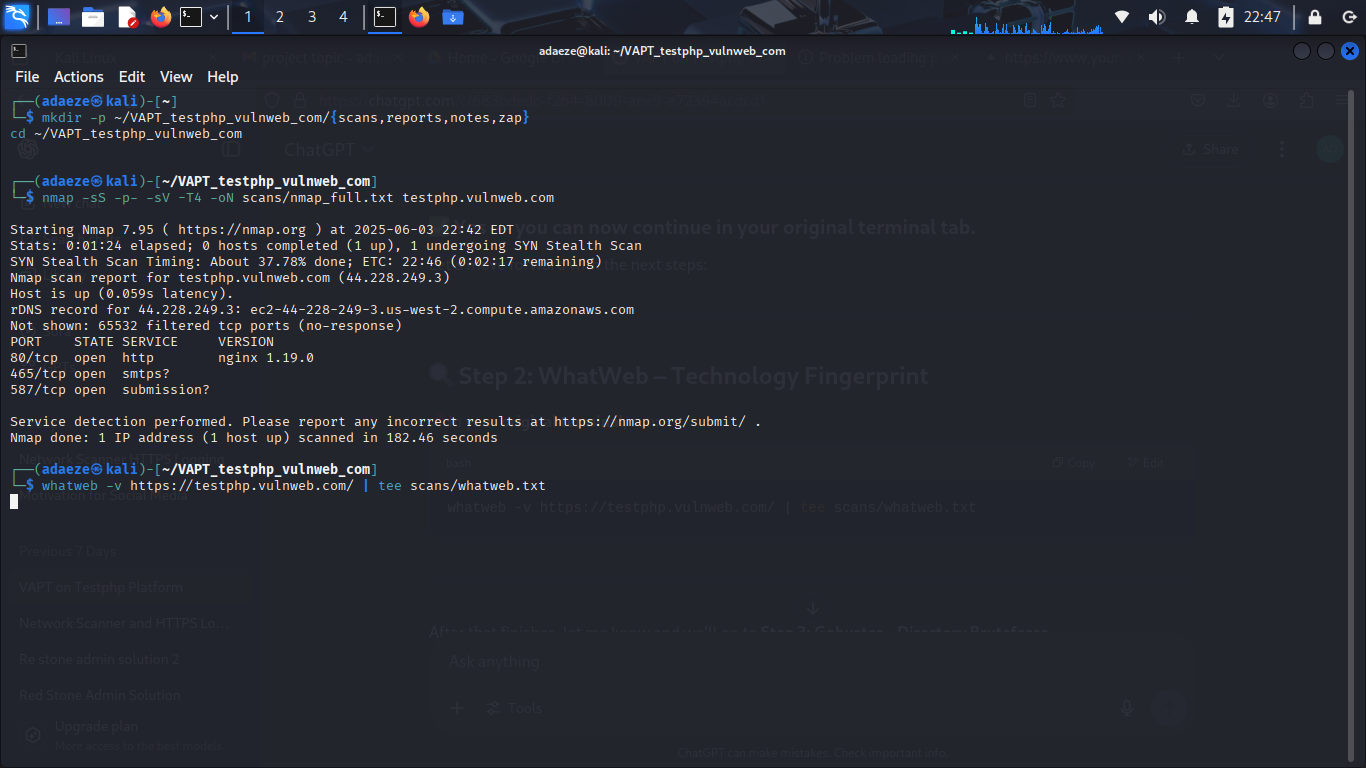
## **13. Conclusion**

The VAPT exercise on https://testphp.vulnweb.com successfully identified a range of vulnerabilities consistent with real-world threat vectors. The findings demonstrate the importance of secure coding practices, robust input validation, and strong session management. The report serves as a practical demonstration of the OWASP Top 10 and helps reinforce ethical hacking methodology.

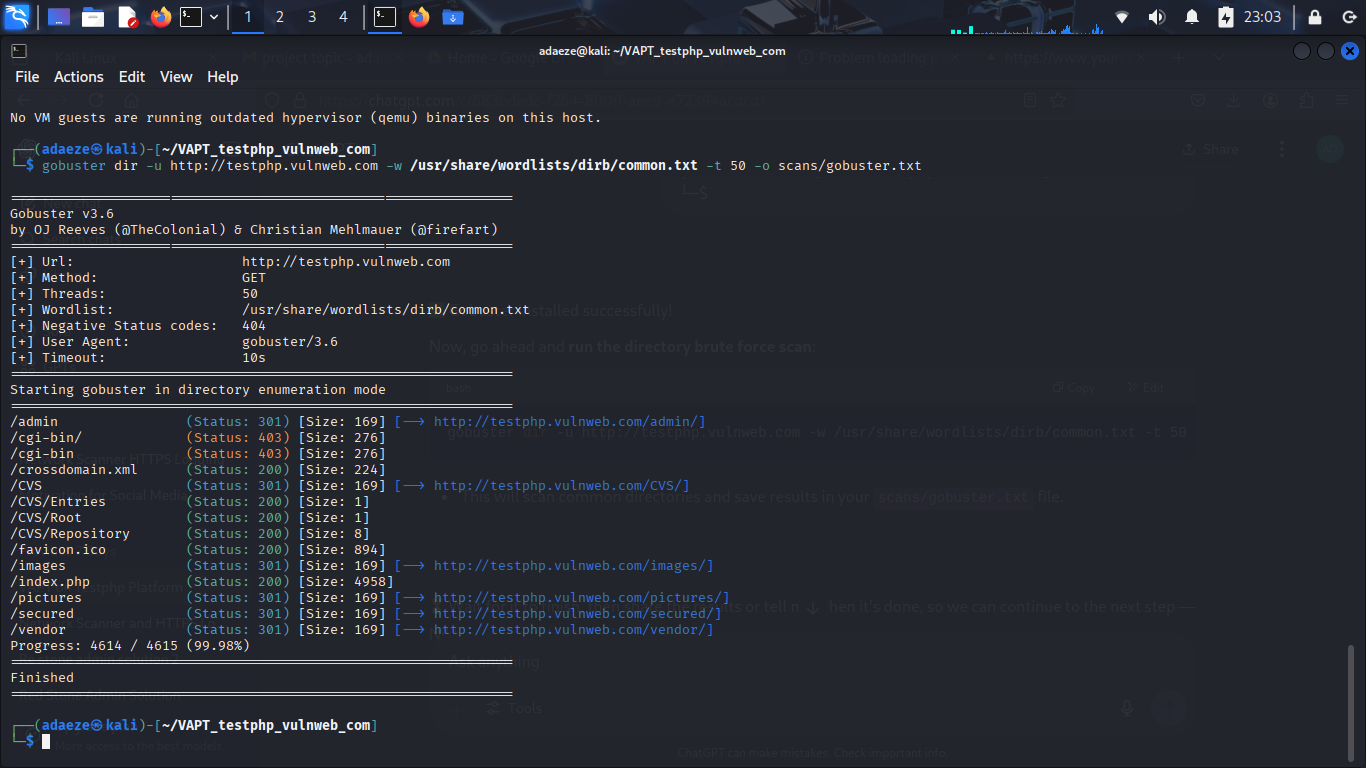
**End of Report.**

Other screen shots for VAPT on <https://testphp.vulnweb.com>

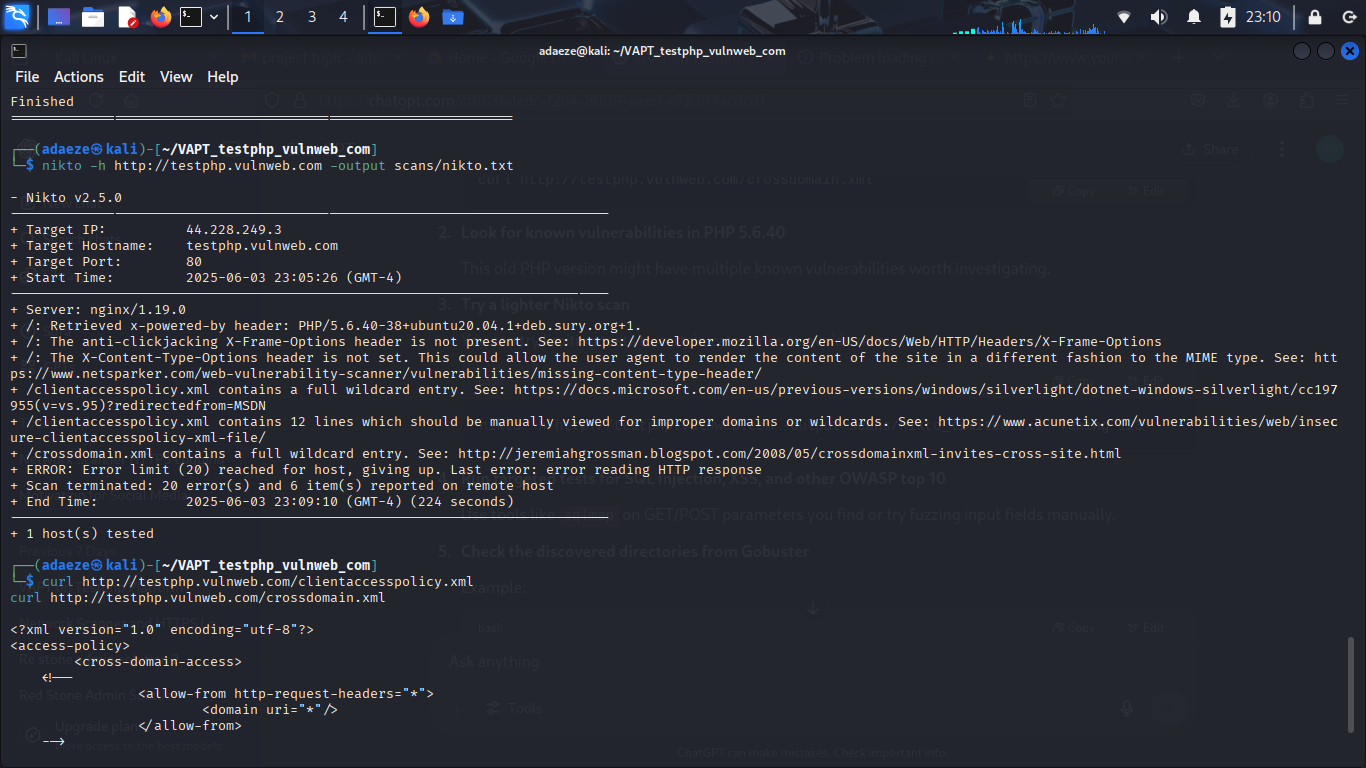
Scanning with NMap



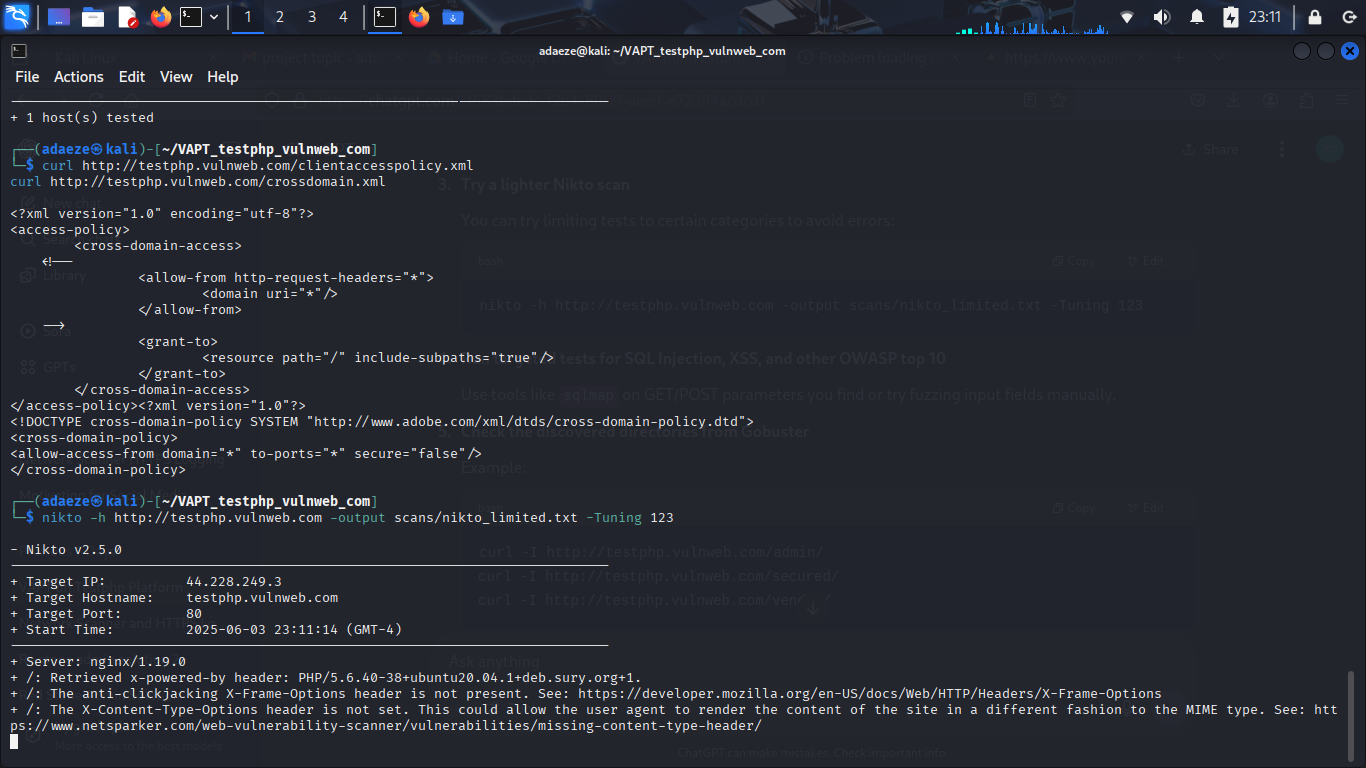
Bruteforce scan



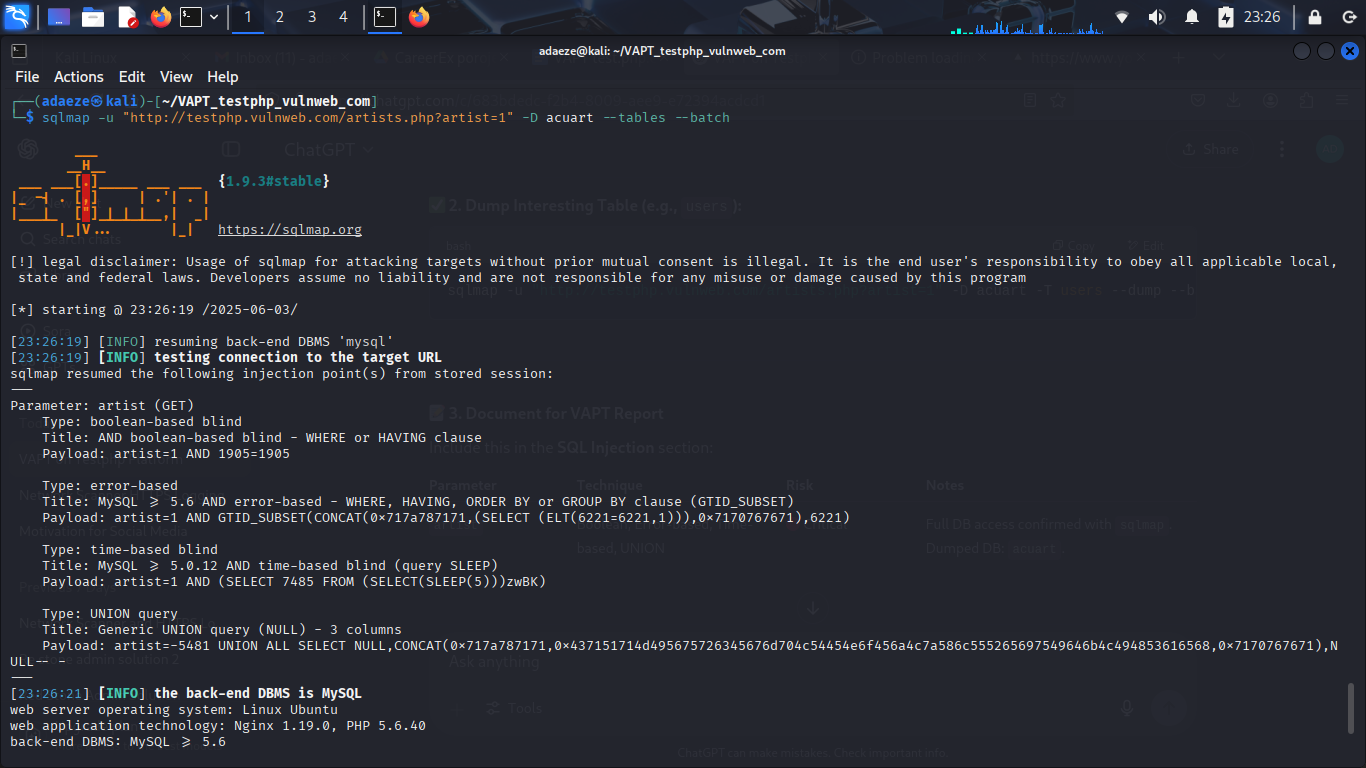
Nikto scanning



Nikto Manual check on wildcard XML



VAPT List Tables in acuart



VAPT dump interesting tables

