Holiday Walkthrough

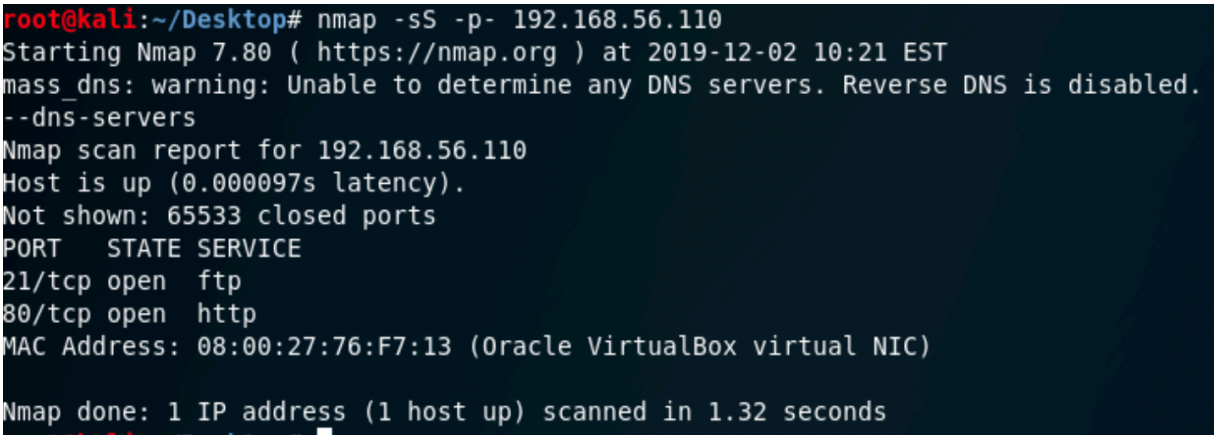
Initial enumeration

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:
| FTP server status:
|      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)
```

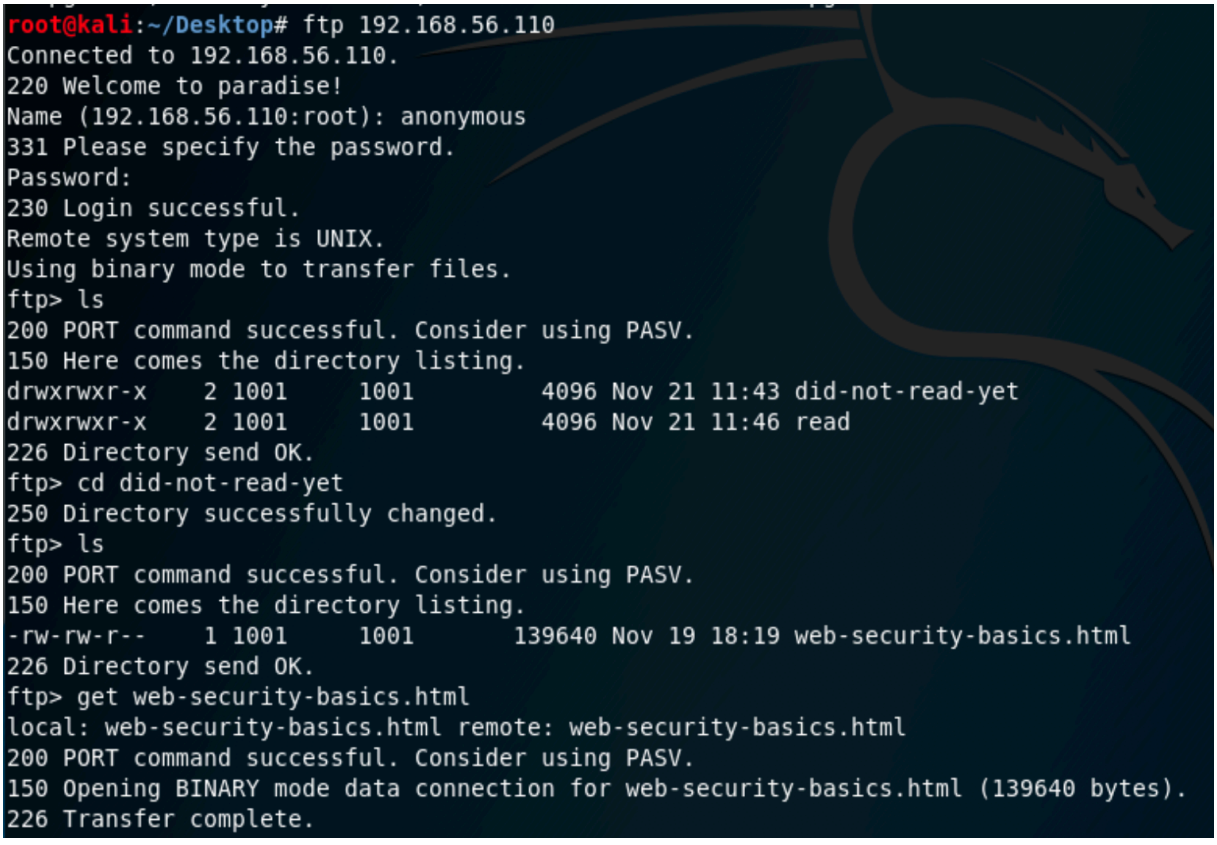
Picture 1. Initial nmap results

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.



Picture 2. No additional TCP ports found open.

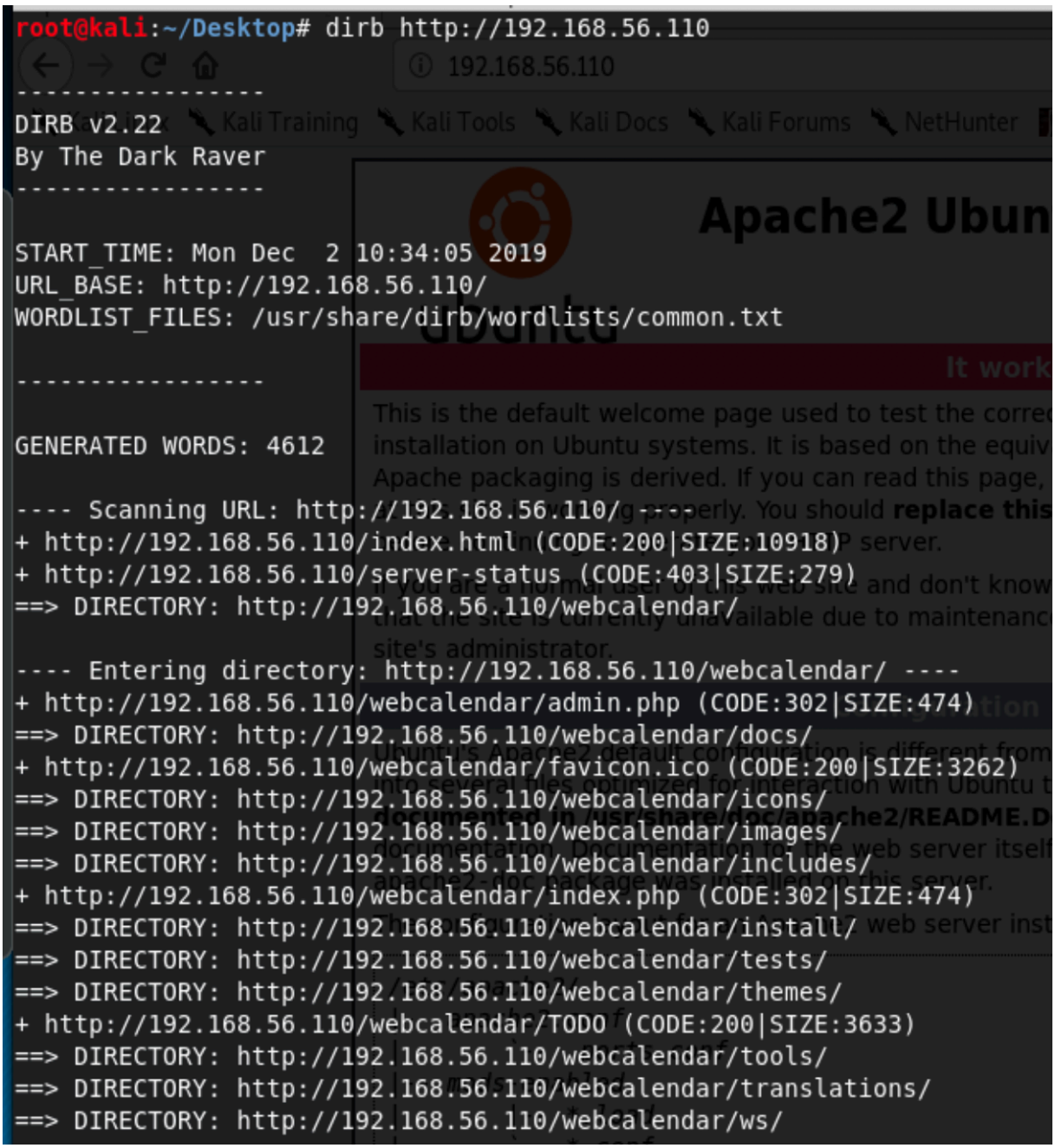
The very first thing nmap told me, was that that from the two services open and running, ftp had anonymous login allowed, so I went ahead and logged in with the username **anonymous** and a blank password. However, the documents I found in the ftp folder did not give me too much, uploading files was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



Picture 3. Interacting with FTP

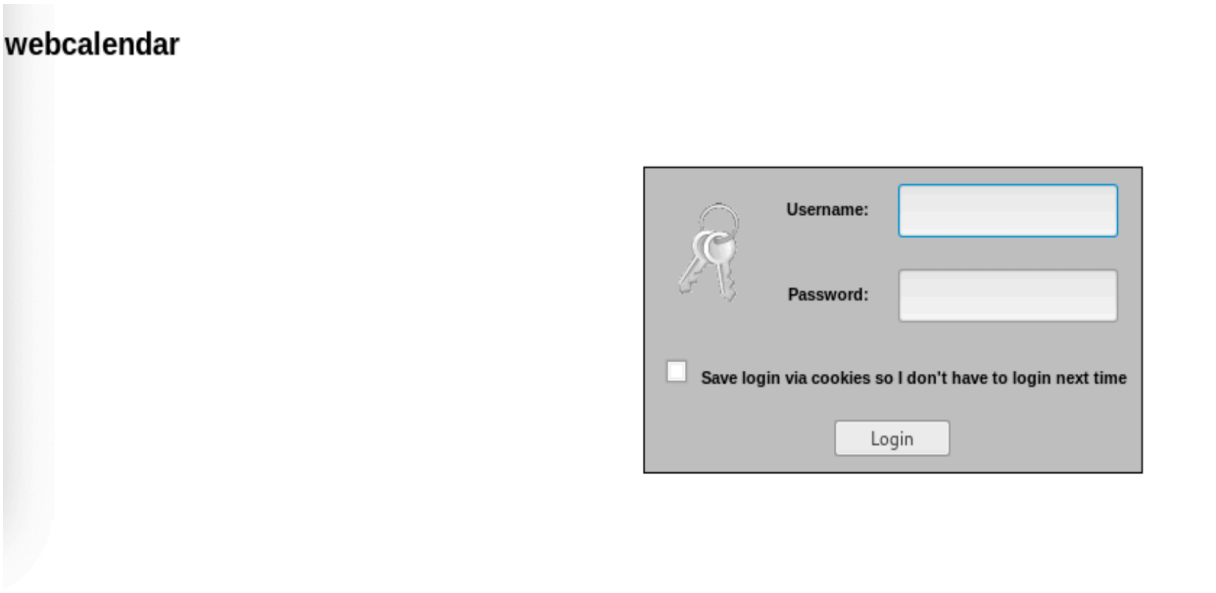
Since this service did not give us too much in terms of access, we carry on to HTTP. If we simply browse to the IP we are greeted by the default apache index page, nothing too interesting, so I started looking for different web content using dirb. Since I did not know what I was looking for yet, I simply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

anything. Most of the directories dirb found are typically found on webservers, however **webcalendar** looked particulary interesting.



Picture 4. Dirb finds the webcalendar installation

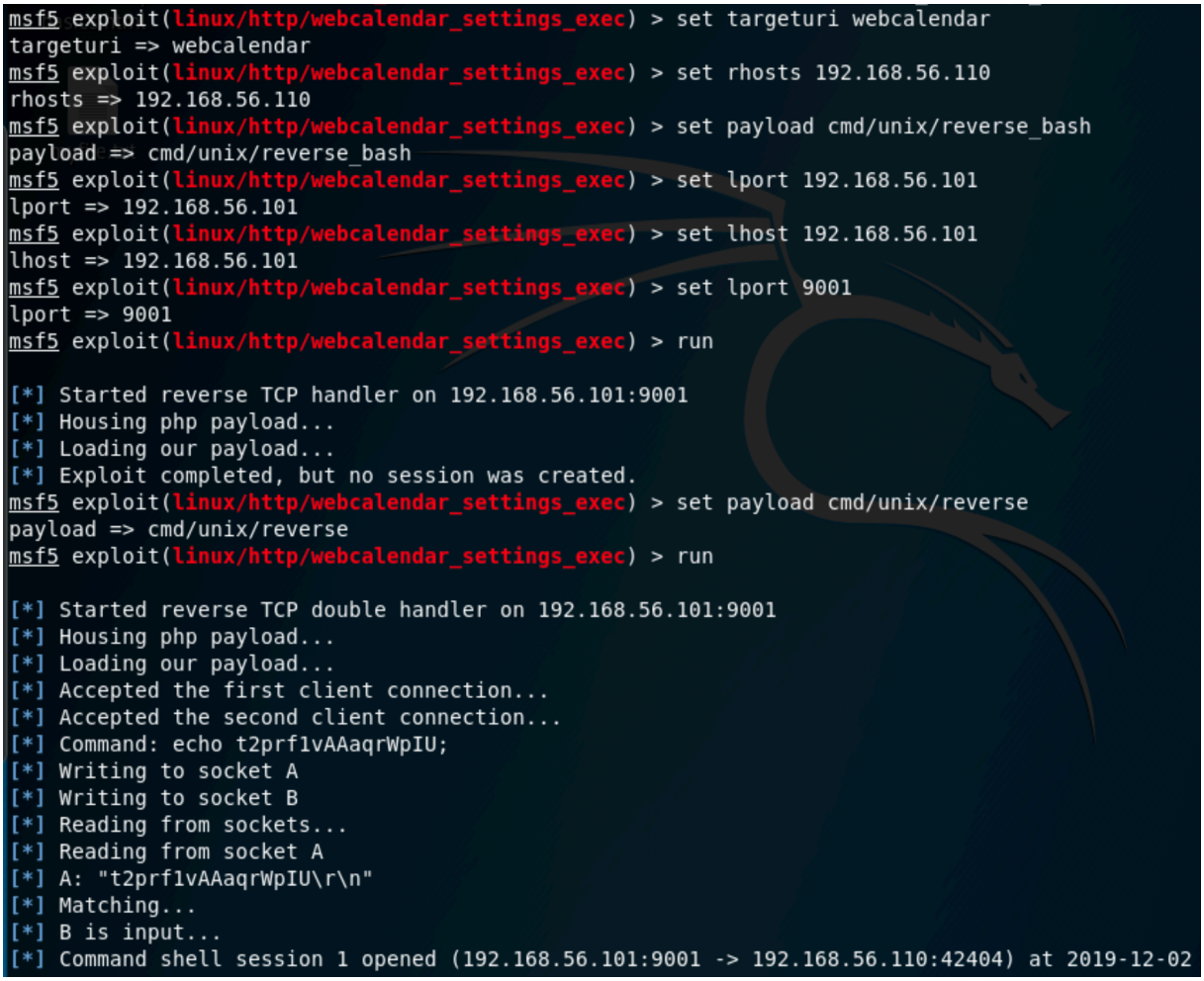
Browsing to 192.168.56.110/webcalendar/ reveals that webcalendar is an application that is indeed installed and running on the system. I first tried gaining access by trying some simple passwords combinations like admin/admin or admin/password but as these did not work out, I checked the version number 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.



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.

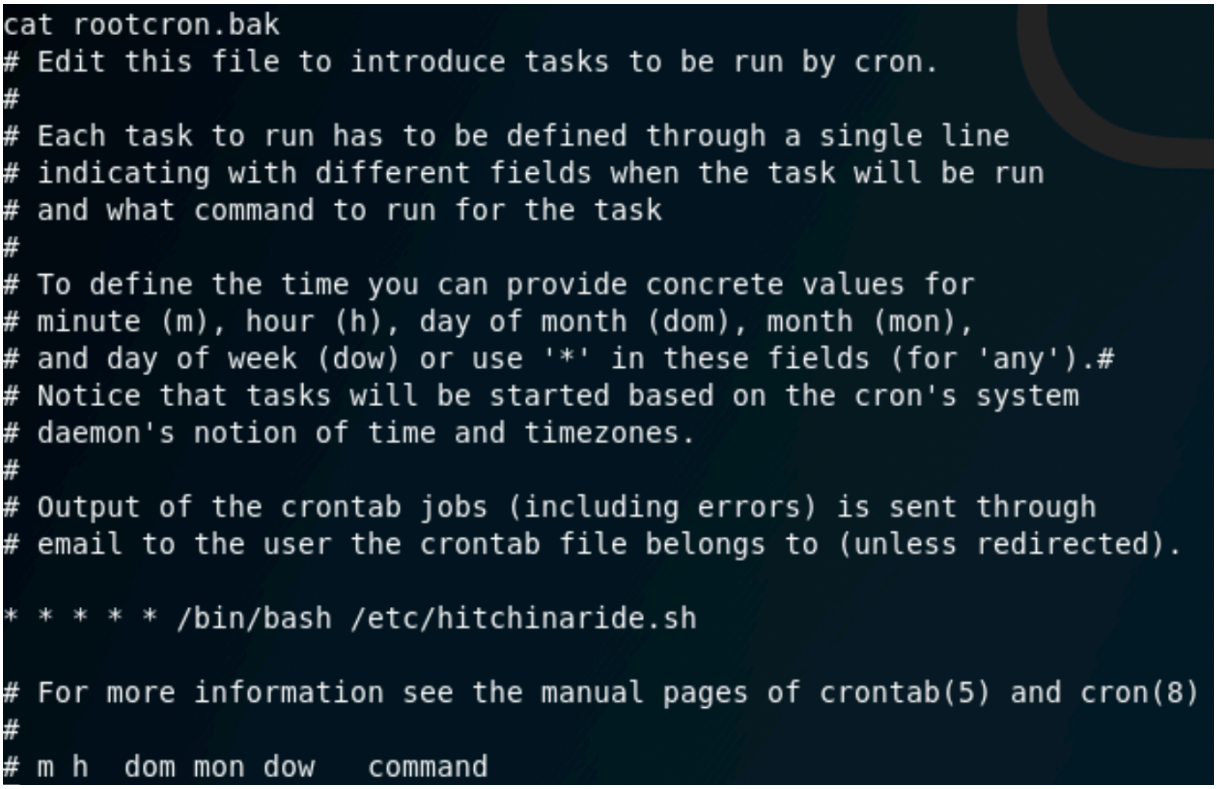


Picture 6. Successful exploitation

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.

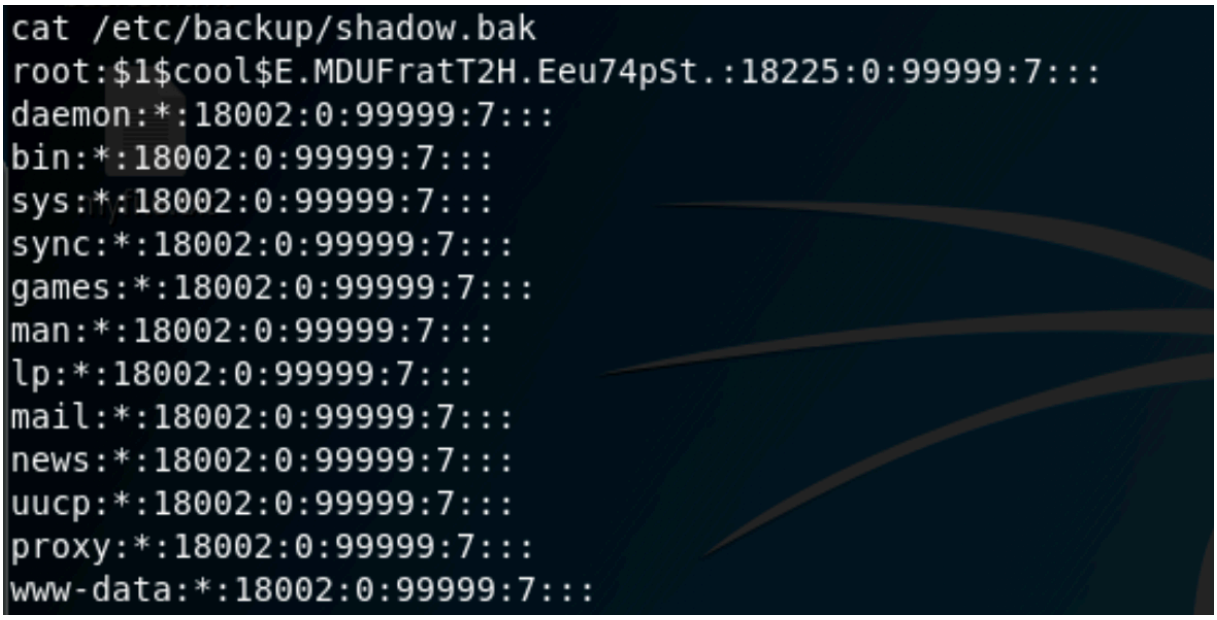
Post shell information gathering

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.



Picture 7. rootcron.bak

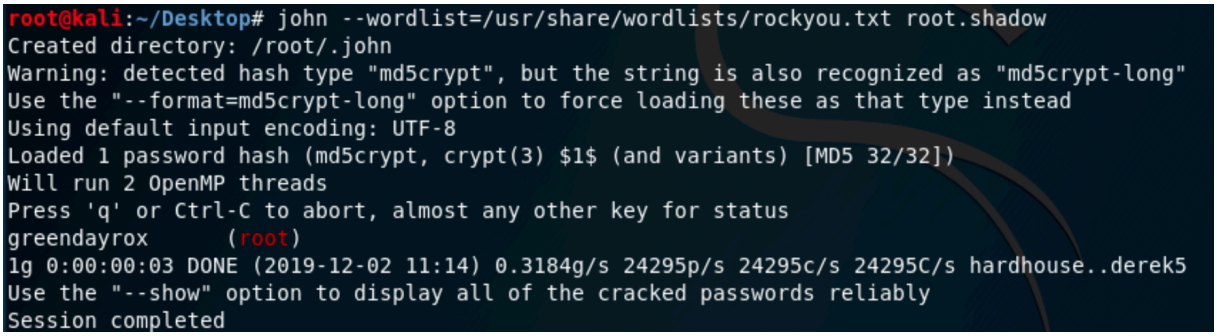
Obviously I proceeded to check out **/etc/hitchinaride.sh** and apperently this script copies the highly sensitive **/etc/shadow** file to a backup in **/etc/backup/shadow.bak**. The file is readable by our user and is formatted in the real shadow format, most importantly with root's password stored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



Picture 8. shadow.bak

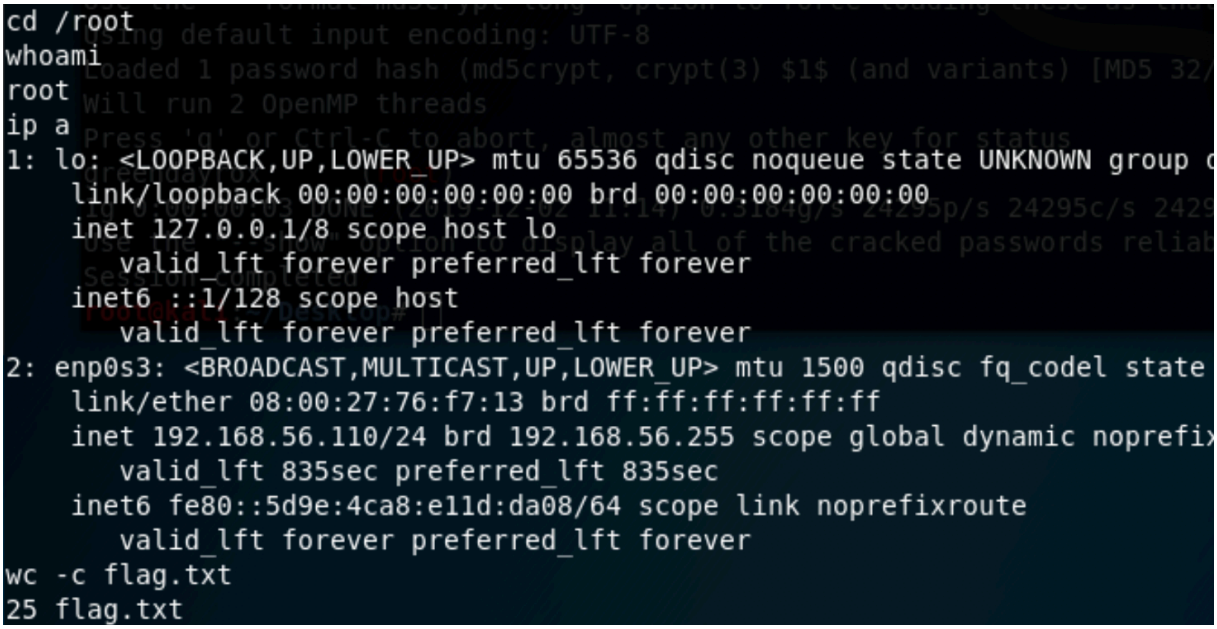
Privilege escalation

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.



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.



Picture 10. Root shell with complete control