

Recon:

We run our nmap scan with the command `nmap -sV -sC -T4 10.10.70.161` in which our results come back fairly quickly, which shows us that port 22 and port 80 are open:

```
blackout@kali:~$ nmap -sV -sC -T4 10.10.70.161
Starting Nmap 7.80 ( https://nmap.org ) at 2021-11-07 21:49 UTC
Nmap scan report for 10.10.70.161
Host is up (0.028s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_   2048 58:14:75:69:1e:a9:59:5f:b2:3a:69:1c:6c:78:5c:27 (RSA)
|_   256 23:f5:fb:e7:57:c2:a5:3e:c2:26:29:0e:74:db:37:c2 (ECDSA)
|_   256 f1:9b:b5:8a:b9:29:aa:b6:aa:a2:52:4a:6e:65:95:c5 (ED25519)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: Can You Find Them All?
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

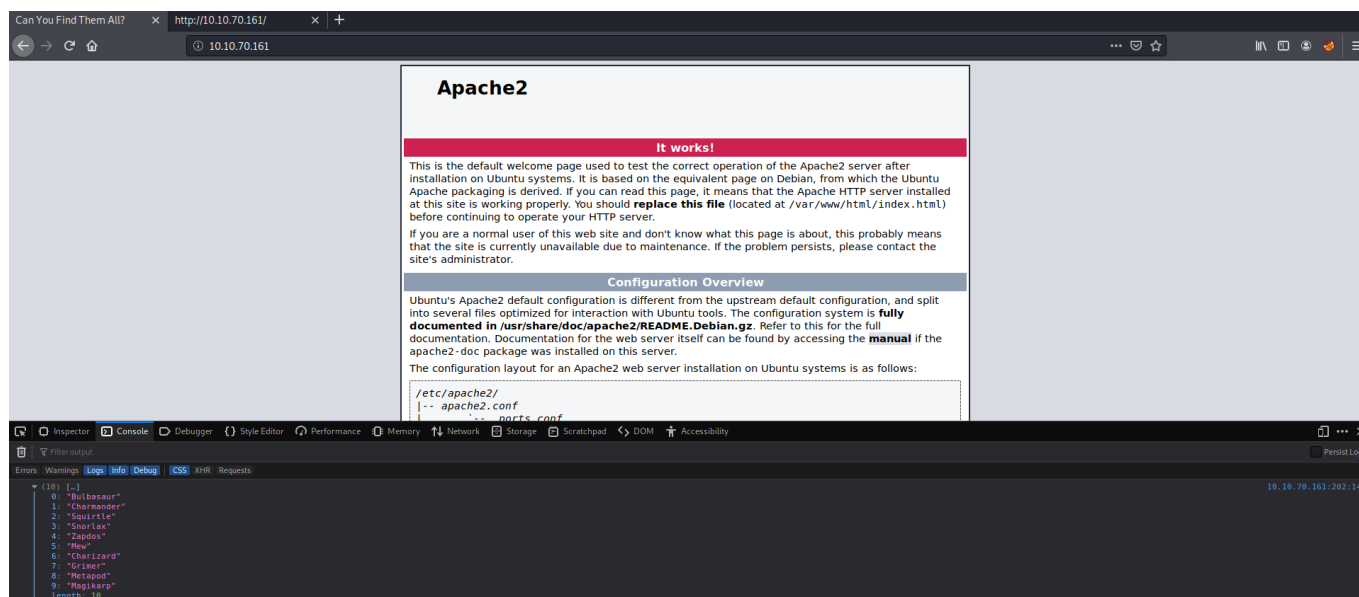
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.41 seconds
blackout@kali:~$
```

Discovery:

Lets check out port 80 and see what we can find - We find an apache webpage which nothing looks out of place until we check the source of the page and see possibly the credentials for SSH with the user being `Pokemon` and the password being `hack_the_pokemon` but we also see a HTML comment saying to check the

console for an extra surprise, which is just a bunch of pokemon:

```
<!--  
<div class="content_section_text">  
  <p>  
    By default, Ubuntu does not allow access through the web browser to  
<em>any</em> file apart of those located in <tt>/var/www</tt>,  
    <a href="http://httpd.apache.org/docs/2.4/mod/mod_userdir.html">public_html</a>  
    directories (when enabled) and <tt>/usr/share</tt> (for web  
    applications). If your site is using a web document root  
    located elsewhere (such as in <tt>/srv</tt>) you may need to whitelist your  
    document root directory in <tt>/etc/apache2/apache2.conf</tt>.  
  </p>  
  <p>  
    The default Ubuntu document root is <tt>/var/www/html</tt>. You  
    can make your own virtual hosts under /var/www. This is different  
    to previous releases which provides better security out of the box.  
  </p>  
</div>  
  
<div class="section_header">  
  <div id="bugs"></div>  
  Reporting Problems  
</div>  
<div class="content_section_text">  
  <p>  
    Please use the <tt>ubuntu-bug</tt> tool to report bugs in the  
    Apache2 package with Ubuntu. However, check <a  
    href="https://bugs.launchpad.net/ubuntu/+source/apache2">existing  
    bug reports</a> before reporting a new bug.  
  </p>  
</div>  
<pokemon>:<hack_the_pokemon>  
  <!--(Check console for extra surprise!)-->  
</div>
```



Foothold:

Now we try and login with the credentials we found earlier and we successfully

logged in to the user **Pokemon**:

```
blackout@kali:~$ ssh pokemon@10.10.70.161
The authenticity of host '10.10.70.161 (10.10.70.161)' can't be established.
ECDSA key fingerprint is SHA256:mXXTCQORSu35gV+cSi+nCjY/W0oabQFNjxuXUDrsUHI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.70.161' (ECDSA) to the list of known hosts.
pokemon@10.10.70.161's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-112-generic x86_64)

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

84 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.

pokemon@root:~$
```

Using the command **ls** to list the folders/files in the current directory, we see a bunch of folders which we **cd** into the **desktop** directory and find an interesting file called **P0kEm0n.zip**:

```
pokemon@root:~$ ls
Desktop Documents Downloads examples.desktop Music Pictures Public Templates Videos
pokemon@root:~$ cd Desktop
pokemon@root:~/Desktop$ ls
P0kEm0n.zip
pokemon@root:~/Desktop$
```

Which we unzip this by using the command **unzip P0kEm0n.zip** - This creates another directory called **P0kEm0n** which we enter this directory as well and find a txt file called **graa-type.txt**, which we cat this to read and see what's inside the file - It apperas to be hex:

```
pokemon@root:~/Desktop$ ls
P0kEm0n.zip
pokemon@root:~/Desktop$ unzip P0kEm0n.zip
Archive: P0kEm0n.zip
  creating: P0kEm0n/
  inflating: P0kEm0n/grass-type.txt
pokemon@root:~/Desktop$ ls
P0kEm0n P0kEm0n.zip
pokemon@root:~/Desktop$ cd P0kEm0n
pokemon@root:~/Desktop/P0kEm0n$ ls
grass-type.txt
pokemon@root:~/Desktop/P0kEm0n$ cat grass-type.txt
50 6f 4b 65 4d 6f 4e 7b 42 75 6c 62 61 73 61 75 72 7d
pokemon@root:~/Desktop/P0kEm0n$
```

We then put this into cyberchef, which automatically decodes this for us and gives us our first flag:

The image shows the CyberChef web interface. On the left, under the 'Recipe' tab, there is a 'From Hex' block with a 'Delimiter' dropdown set to 'Space'. On the right, the 'Input' pane contains a hex string: 50 6f 4b 65 4d 6f 4e 7b 42 75 6c 62 61 73 61 75 72 7d. The 'Output' pane at the bottom shows the result of the conversion: PoKeMoN{Bulbasaur}.

Find the Grass-Type Pokemon:

PoKeMoN{Bulbasaur}

I was wondering where the other flag may be and thought maybe it would have a similar name to the last flag so I used the command `locate` to find the flag and as this flag is located to water I tried this command `locate water-type.txt` and I was successful and managed to find the flag - It was in the `/var/www/html` which is the web directory and then we read the file - However it seems that the flag is encrypted:

```
pokemon@root:~/Desktop/P0kEm0n$ locate water-type.txt
/var/www/html/water-type.txt
pokemon@root:~/Desktop/P0kEm0n$ cat /var/www/html/water-type.txt
EcGudfxq_EcGmP{EcGudfxq}
Sun Nov  7 21:47:20 2021 Outgoing Control Channel Authentication: Using 512 bit me
Sun Nov  7 21:47:20 2021 Incoming Control Channel Authentication: Using 512 bit me
Sun Nov  7 21:47:20 2021 TCP/UDP: Preserving recently used remote address: [AF INE
```

```
hash-identifier
```

```

blackout@kali:~$ hash-identifier locate water-type.txt
#####
pol# mon@root: Desktop/PokeMon$ cat /var/www/html/water-type.txt #
Ec# ifxodE{PmGcE_qxfdugcEpokemon@root: Desktop/PokeMon$ #
tr# -ly #
pol# mon@root: Desktop/PokeMon$ cat /var/www/html/water-type.txt #
-# #
pol# mon@root: Desktop/PokeMon$ cat /var/www/html/water-type.txt #
pol# mon@root: Desktop/PokeMon$ cat /var/www/html/water-type.txt #
pol# mon@root: Desktop/PokeMon$ cat /var/www/html/water-type.txt #
inc# .html: water-type.txt v1.2 #
pol# mon@root: Desktop/PokeMon$ cat /var/www/html/water-type.txt | rev By Zion3R #
pol# mon@root: Desktop/PokeMon$ cat /var/www/html/water-type.txt | rev www.Blackploit.com #
pol# mon@root: Desktop/PokeMon$ cat /var/www/html/water-type.txt | rev Root@Blackploit.com #
#####

```

```

HASH: Ecgudfxq_EcGmP{Ecgudfxq}

```

```

Not Found.

```

```

HASH: █

```

I put it into cyberchef but this time it couldn't automatically find the correct decoder/decipher then I realised that it wasn't a hash but a cipher so I searched up on Google cipher identifier and found out that it was ROT13:



ROT13

14

Recipe

ROT13

☒ Rotate lower case chars

☒ Rotate upper case chars

☐ Rotate numbers

Amount

14

⏏

⏏

⏏

Input

start: 0
end: 24
length: 24

Ecgudfxq_EcGmP{Ecgudfxq}

Output

Squirtle_SqUaD{Squirtle}

Find the Water-Type Pokemon:

Squirtle_SqUaD{Squirtle}

To find the next flag I do the exact same thing I did last time to find it by using the `locate` command - This type was fire so I used `locate fire-type.txt` and found it in an interesting directory `/etc/why_am_i_here?/fire-type.txt` - I then read the file and it seems that the flag is encoded with base64 - It was easy to identify that it was base64 due to the two `==` at the end of the encoding:

```
pokemon@root:/var/www/html$ locate fire-type.txt
/etc/why_am_i_here?/fire-type.txt
pokemon@root:/var/www/html$ cd /etc/why_am_i_here?/fire-type.txt
-bash: cd: /etc/why_am_i_here?/fire-type.txt: Not a directory
pokemon@root:/var/www/html$ cd /etc/why_am_i_here?
pokemon@root:/etc/why_am_i_here?$ ls
fire-type.txt
pokemon@root:/etc/why_am_i_here?$ cat fire-type.txt
UDBrM20wbntDaGFybWFuZGVyYfQ=pokemon@root:/etc/why_am_i_here?$
```

We then can use the command `cat fire-type.txt | base64 -d` which we'll decode the text and make us get our third flag:

```
pokemon@root:/etc/why_am_i_here?$ cat fire-type.txt 129921 5-[129921-212992]
UDBrM20wbntDaGFyBWfFuZGVyfQ=pokemon@root:/etc/why_am_i_here?$ cat fire-type.txt | base64 -d
P0k3m0n{Charmander}pokemon@root:/etc/why_am_i_here?$
```

Find the Fire-Type Pokemon:

P0k3m0n{Charmander}

Privilege Escalation:

We now need to find our final flag so we go back to the `/home` directory which we find another user called `ash` and the last flag but however we are not able to read it as we are not root:

```
pokemon@root:/etc/why_am_i_here?$ sudo -l
[sudo] password for pokemon:
Sorry, user pokemon may not run sudo on root.
pokemon@root:/etc/why_am_i_here?$ cd /home
pokemon@root:/home$ ls
ash  pokemon  roots-pokemon.txt
pokemon@root:/home$ cat roots-pokemon.txt
cat: roots-pokemon.txt: Permission denied
pokemon@root:/home$
```

We can not `cd` into the `ash` directory either and we also get a `permission denied` when trying to enter the directory - As `Ash` is only able to read this file we use the command `grep -r "ash" . 2>/dev/null` to show all his permissions and how we can get access to his account - Which this command successfully shows us his password and we are able to log into `ash`:

```
Binary file ./pokemon/.mozilla/firefox/default/sessionstore-backups/upgrade.json matches
./pokemon/.bashrc:~/.bashrc: executed by bash(1) for non-login shells.
./pokemon/.bashrc:# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
./pokemon/.bashrc:# See bash(1) for more options
./pokemon/.bashrc:# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
./pokemon/.bashrc:# ~/.bash_aliases, instead of adding them here directly.
./pokemon/.bashrc:# See /usr/share/doc/bash-doc/examples in the bash-doc package.
./pokemon/.bashrc:if [ -f ~/.bash_aliases ]; then
./pokemon/.bashrc:    . ~/.bash_aliases
./pokemon/.bashrc:# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
./pokemon/.bashrc:# sources /etc/bash.bashrc).
./pokemon/.bashrc: if [ -f /usr/share/bash-completion/bash_completion ]; then
./pokemon/.bashrc:    . /usr/share/bash-completion/bash_completion
./pokemon/.bashrc: elif [ -f /etc/bash_completion ]; then
./pokemon/.bashrc:    . /etc/bash_completion
./pokemon/.bashrc:echo 'unset HISTFILE' >> ~/.bashrc
./pokemon/.bashrc:echo 'export LESSHISTFILE="-"' >> ~/.bashrc
./pokemon/.profile:# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
./pokemon/.profile:# see /usr/share/doc/bash/examples/startup-files for examples.
./pokemon/.profile:# the files are located in the bash-doc package.
./pokemon/.profile:# if running bash
./pokemon/.profile:    # include .bashrc if it exists
./pokemon/.profile:    if [ -f "$HOME/.bashrc" ]; then
./pokemon/.profile:        . "$HOME/.bashrc"
./pokemon/Videos/Gotta/Catch/Them/ALL/!Could_this_be_what_Im_looking_for?.cplplusplus:std::cout << "ash : pikapika"
Binary file ./pokemon/.config/dconf/user matches
./pokemon/.config/nautilus/accels: (gtk_accel_path "<Actions>/DirViewActions/Trash" "Delete")
./pokemon/.config/nautilus/accels: (gtk_accel_path "<Actions>/DirViewActions/LocationTrash" "")
./pokemon/.config/nautilus/accels: (gtk_accel_path "<Actions>/DirViewActions/LocationRestoreFromTrash" "")
./pokemon/.config/nautilus/accels: (gtk_accel_path "<Actions>/DirViewActions/Empty Trash" "")
./pokemon/.config/nautilus/accels: (gtk_accel_path "<Actions>/ShellActions/PromptLocationSlashAccel" "slash")
./pokemon/.config/nautilus/accels: (gtk_accel_path "<Actions>/CanvasViewActions/Sort by Trash Time" "")
./pokemon/.config/nautilus/accels: (gtk_accel_path "<Actions>/DirViewActions/Restore From Trash" "")
./pokemon/.config/nautilus/accels: (gtk_accel_path "<Actions>/DesktopViewActions/Empty Trash Conditional" "")
Binary file ./pokemon/.cache/compizconfig-1/unityshell.pb matches
Binary file ./pokemon/.cache/gstreamer-1.0/registry.x86_64.bin matches
```

```
pokemon@root:/home$ su ash
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

bash: /home/ash/.bashrc: Permission denied
ash@root:/home$ ls
ash  pokemon  roots-pokemon.txt
```

We are then able to read the `roots-pokemon.txt` and get our final flag:

```
ash@root:/home$ ls
ash  pokemon  roots-pokemon.txt
ash@root:/home$ cat roots-pokemo.txt
cat: roots-pokemo.txt: No such file or directory
ash@root:/home$ cat roots-pokemon.txt
Pikachu!ash@root:/home$
```

Who is Root's Favorite Pokemon?:

Pikachu!