

# Cyber security & basic administration field manual

Future Factory – ZZPP0920



This manual is meant for introducing students into basics of server-side cyber security. This is not a comprehensive or a complete guide but rather few tips and tricks how to get into securing your server. This guide does not follow the most secure industry standards but rather the bare minimum and “Nice to have” topics.

If you are interested in building the most secure server possible, I would suggest you to look into **Zero-trust architecture**.

# Contents

Updates .....	3
Which software to use for connecting? .....	4
SSH – Port 22 .....	5
Fail2Ban .....	7
Whowatch .....	8
Btop .....	10
Setting up postgres database backup on remote server .....	11
Troubleshooting: .....	12
Setting up backup & transfer: .....	13
Comprehensive security audit .....	16

# Updates

It's generally recommended to run **apt-get update** before running **apt-get upgrade**.

The **apt-get update** command updates the package list from the repositories, which ensures that your system has the latest information about available packages and versions.

The **apt-get upgrade** command upgrades installed packages to their latest available versions based on the updated package list obtained by apt-get update.

Running **apt-get update** first ensures that the latest package information is available, which helps to prevent issues and conflicts during the upgrade process.

```
ubuntu@farinemill:~/backup$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:2 http://nova.clouds.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://nova.clouds.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://nova.clouds.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:5 http://nova.clouds.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2436 kB]
Get:6 http://nova.clouds.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1039 kB]
Fetched 3811 kB in 1s (2551 kB/s)
Reading package lists ... Done
ubuntu@farinemill:~/backup$ sudo apt-get upgrade
Reading package lists ... Done
Building dependency tree
Reading state information ... Done
Calculating upgrade ... Done
The following packages will be upgraded:
  bind9-dnsutils bind9-host bind9-libs cloud-init libunwind8 update-notifier-common
6 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 2041 kB of archives.
After this operation, 37.9 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

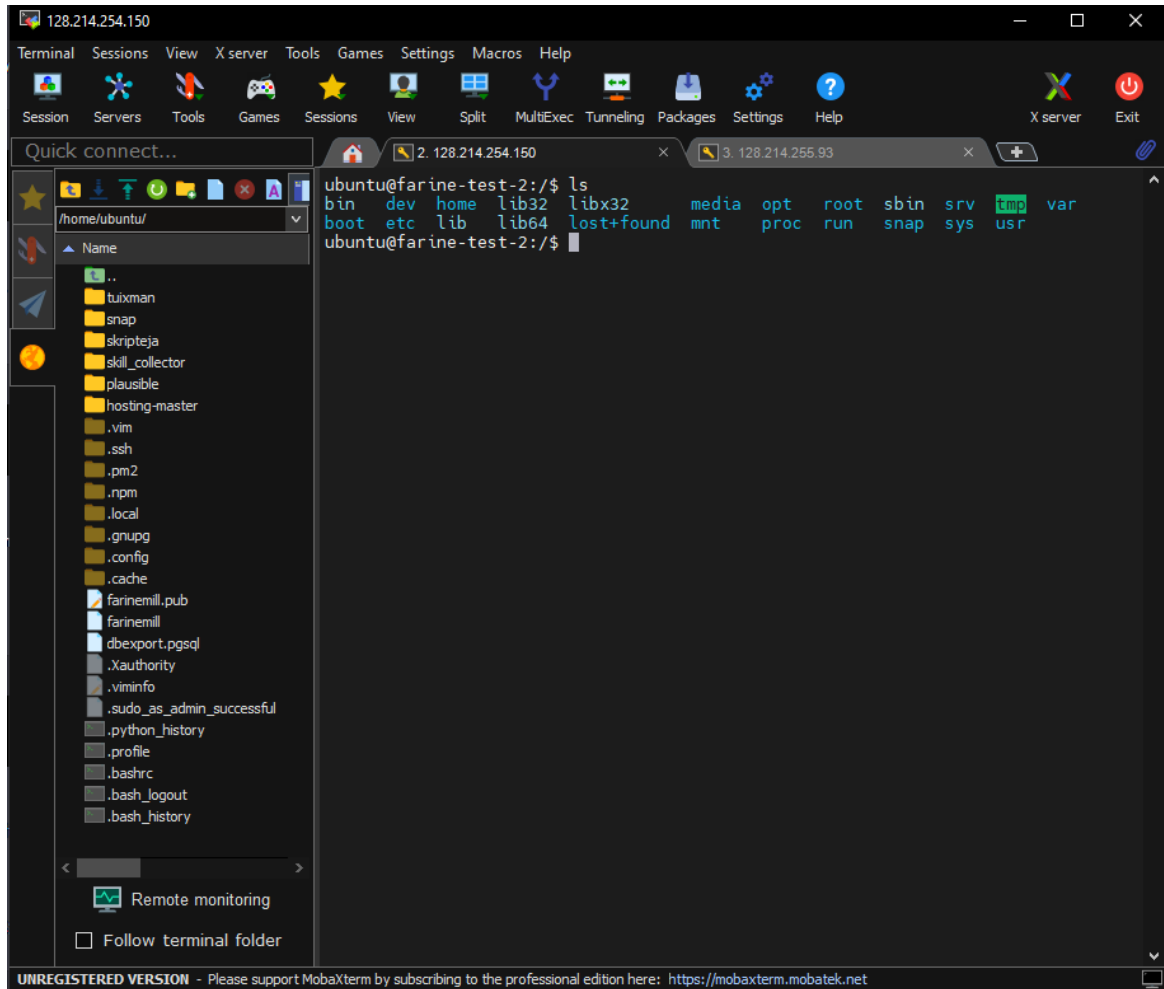
Press **Y** to continue.

Remember to make sure that the date and time in your server is also correct. Otherwise logs and events are not accurate.

```
sudo timedatectl set-timezone Europe/Helsinki
```

# Which software to use for connecting?

Gang De Farine in general used MobaXTerm free portable version. The reason for this is mostly because it has very user-friendly interface and it makes the process much more enjoyable.



## Key points:

- Easy to download or upload anything.
- It's fast to swap between server windows.
- It looks nice.
- MobaXTerm shows you the filesystem on left.
- Has split screen if you want to use it.

You can configure MobaXTerm to use your desired username, password or ssh key so all you need to do is just press your saved IP address once and you're in.

## SSH – Port 22

**Disable** password login through SSH and only allow private key authentication.

```
ubuntu@farine-test-2:/$ sudo nano /etc/ssh/sshd_config
```

Look for the line that says **PasswordAuthentication** and change the value to no:

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
```

**Warning:** You will need to enable password authentication for a moment when you create a new ssh file and send the ssh public key to your backup server. Once the key has been sent you can disable password authentication again.

For setting up taking backups from database and sending them off to another server, which we will discuss later, let's also make sure these configurations match:

```
PubkeyAuthentication yes
```

```
# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no
```

```
sudo systemctl restart sshd
```

The reason you should disable password authentication is because once your server has a public IP address it will immediately get attacked by constant stream of bots. You are able to follow it by typing the following command:

**Sudo tail -f /var/log/auth.log**

```
ubuntu@farine-test-2:/$ sudo tail -f /var/log/auth.log
Mar 24 19:03:29 farine-test-2 sshd[195752]: Invalid user test1 from 186.156.178.35 port 38014
Mar 24 19:03:30 farine-test-2 sshd[195752]: pam_unix(sshd:auth): check pass; user unknown
Mar 24 19:03:30 farine-test-2 sshd[195752]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=186.156.178.35
Mar 24 19:03:32 farine-test-2 sshd[195752]: Failed password for invalid user test1 from 186.156.178.35 port 38014 ssh2
Mar 24 19:03:34 farine-test-2 sshd[195752]: Received disconnect from 186.156.178.35 port 38014:11: Bye Bye [preauth]
Mar 24 19:03:34 farine-test-2 sshd[195752]: Disconnected from invalid user test1 186.156.178.35 port 38014 [preauth]
```

With the same tail command you'll be able to follow other logs too in case you have services running in public.

Ideal way of distributing SSH keys within your team is that each member has their own username and key pair associated to it and on top of that during the login process they are required to type a password for their key to work. While this is ideal, depending on individual team skill levels it may cause unnecessary administration work.

The reason behind this is that if you have one key and it leaks then the attacker basically owns the server after that but then again, this is a 5 member IT student group, not a strictly controlled development environment.

You may also distribute the copies of the SSH key without setting passwords provided by CSC Pouta for speeding things up. This is not the correct way but it'll work.

**Tip 1: Remember to ask your customer what kind of user management and security settings he wants for the end product!**

**Tip 2: In case everything is fine, you may still want to change fresh SSH keys for the customer regardless. Otherwise, you are delivering a product that is already upside down when it comes to credential security 😊**

# Fail2Ban

Fail2ban is a software that helps protect servers from brute-force attacks and other types of malicious traffic by monitoring logs and automatically blocking IP addresses that show suspicious activity. It works by analyzing log files.

```
ubuntu@farine-test-2:/$ sudo apt-get install fail2ban
```

Once Fail2ban is installed, you can configure it by editing its configuration file located at `/etc/fail2ban/jail.conf` or `/etc/fail2ban/jail.local`. The `jail.local` file takes precedence over `jail.conf`, so it's recommended to use `jail.local` for your customizations.

Here's a basic configuration example for Fail2ban:

First, create a backup copy of the original `jail.local` configuration file:

```
ubuntu@farine-test-2:/$ sudo cp /etc/fail2ban/jail.local /etc/fail2ban/jail.local.bak
```

Edit the `jail.local` configuration file and add the following lines to the end:

```
[ssh]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 5
bantime = 600
```

This configuration will monitor the SSH logs and ban IP addresses that fail to authenticate more than 5 times within 10 minutes (600 seconds).

Restart Fail2Ban by typing:

```
sudo systemctl restart fail2ban
```

To remove a banned IP address from a fail2ban jail:

```
sudo fail2ban-client set <jail-name> unbanip <IP-address>
```

# Whowatch

## Highly recommended, very useful tool.

Whowatch is a command-line utility for monitoring and displaying the processes running on a Linux system in real-time. It displays the list of users logged in, the processes they are running, and the system resources being used by each process.

```
sudo apt-get install whowatch
```

Once the installation is complete, you can run whowatch by typing whowatch in a terminal window. The utility will display a list of logged-in users, the processes they are running.

Whowatch is very useful for detecting weird user activity or reverse shells in case something is going on. You will be also able to follow what each user in your server is doing and how did they end up doing it.

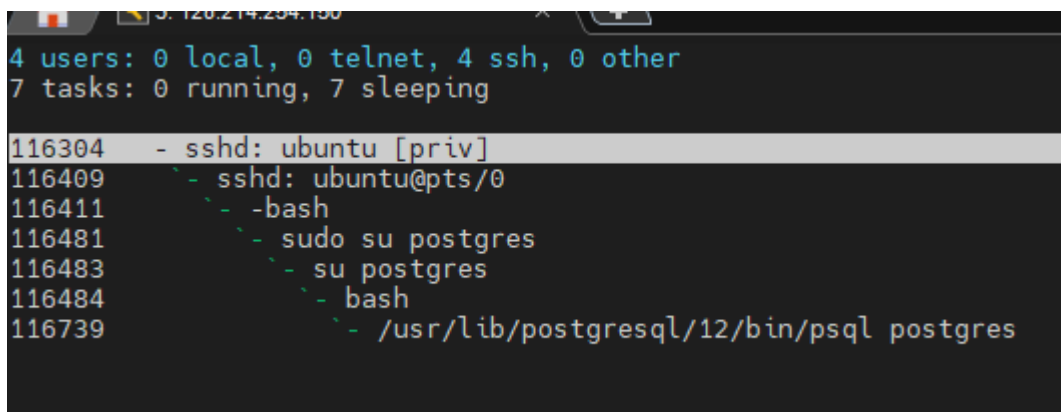
Example:

There are 4 users logged into the test server and each of them are using Ubuntu as their user.

```
4 users: 0 local, 0 telnet, 4 ssh, 0 other
(sshd)   ubuntu pts/0 95.175.104.20 -
(sshd)   ubuntu pts/2 95.175.104.20 -
(sshd)   ubuntu pts/3 88.115.192.208 -
(sshd)   ubuntu pts/4 85.202.81.140 -
```



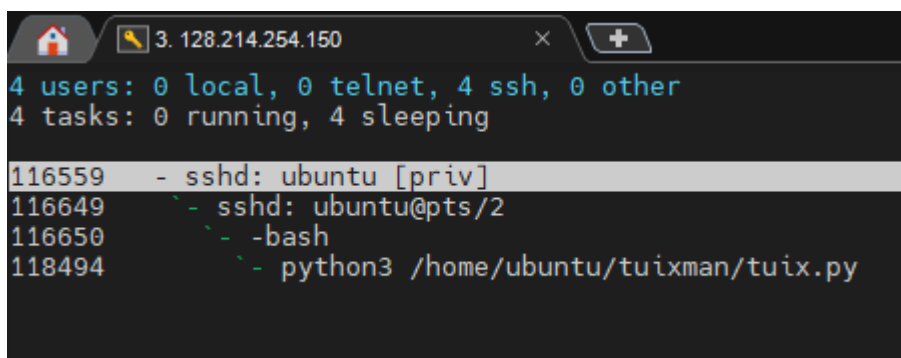
By pressing enter on one of the Ubuntu's, you will be able to see which path they have chosen to follow.



```
3. 128.214.254.150
4 users: 0 local, 0 telnet, 4 ssh, 0 other
7 tasks: 0 running, 7 sleeping
116304 - sshd: ubuntu [priv]
116409   - sshd: ubuntu@pts/0
116411     - -bash
116481     - sudo su postgres
116483     - su postgres
116484     - bash
116739     - /usr/lib/postgresql/12/bin/psql postgres
```

It appears out developer is working on database.

Meanwhile somebody else is working on Tuix.



```
3. 128.214.254.150
4 users: 0 local, 0 telnet, 4 ssh, 0 other
4 tasks: 0 running, 4 sleeping
116559 - sshd: ubuntu [priv]
116649   - sshd: ubuntu@pts/2
116650     - -bash
118494     - python3 /home/ubuntu/tuixman/tuix.py
```

If penetration testing and lateral movement is familiar concept to you, it will become seemingly clear how easy it is to catch unwanted shells or impersonation with this tool.

# Btop

Btop is a terminal-based monitoring tool that allows you to monitor the system's resources such as CPU, memory, and disk usage in real-time. It is a popular alternative to the traditional 'top' command, providing a more user-friendly and interactive interface.

```
ubuntu@farine-test-2:/$ sudo apt install btop
```

The top screenshot shows the Btop terminal interface. At the top, it displays system information: Xeon (S) 2.2 GHz, CPU usage (C0: 1%, C1: 2%, C2: 1%), and LAV: 0.00 0.01 0.02. Below this, it shows system uptime (up 4d 18:01) and resource usage statistics for memory, disks, and network. The main part of the screen is a process list with columns for PID, Program, Command, per-core, reverse, tree, cpu, and lazy. The bottom screenshot shows the Btop++ configuration menu (v1.2.13) with a 'color theme' selection screen. The menu includes options for theme background, truecolor, force tty, vim keys, presets, rounded corners, graph symbol, braille, clock format, base 10 sizes, background update, show battery, and selected battery. The background update is currently set to 'True'.

# Setting up postgres database backup on remote server

Setting up a secure backup can be a real can of worms on larger environments with multiple privileged users but in our case we are doing it as simple manner as possible. You need to launch additional server, install and configure everything above where the customer wants his backups and logs to be sent.

Both servers need to have SSH settings mentioned earlier in order for this to work. You will also have to open password authentication unless you download the public key to your local computer and then upload to your backup server which also works. If that is the case, you can keep the password login disabled.

Create a new key pair

```
ssh-keygen -t rsa
```

This will create a public and private key pair in the default location `~/.ssh/id_rsa.pub` and `~/.ssh/id_rsa`.

Copy the contents of your public key (`id_rsa.pub`) to the `authorized_keys` file of your user account on the server. You can use the following command to copy the key:

```
ssh-copy-id username@server_ip_address
```

Replace `username` with your actual username and `server_ip_address` with the IP address of your server.

To test the connection:

```
ssh username@server_ip_address
```

# Troubleshooting:

If you are not able to get it to work one way is to simply drop the SSH key into your home folder and try to connect by typing the following command:

```
sudo ssh -i farinemill ubuntu@128.214.255.93
```

Where farinemill is our private key into backup server, ubuntu is backup servers default user and the IP mentioned is the server where we want to connect.

If it still does not work try commenting out the GSSAPI related options in sshd\_config file.

```
# GSSAPI options
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no
#GSSAPIEnablek5users no
```

^G Get Help    ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify    ^C Cur Pos  
^X Exit        ^R Read File    ^\ Replace     ^U Uncut Text ^T To Spell    ^\_ Go To Line

Also, make sure the UsePAM line is set to yes:

```
# WARNING: 'UsePAM no' is not supported in Fedora and may cause several
# problems.
UsePAM yes
#AllowAgentForwarding yes
```

^G Get Help    ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify    ^C Cur Pos  
^X Exit        ^R Read File    ^\ Replace     ^U Uncut Text ^T To Spell    ^\_ Go To Line

In order to updates to take effect you need to restart the service by typing: **systemctl restart sshd**

If your owner permissions are not set to read, write, and execute (**drwx-----**), use the chmod to change them: **chmod 0700 /home/ubuntu**

Go to the **.ssh** and recheck permissions by typing **ls -ld**.

This directory should also have read, write, and execute permissions for the file owner. To enforce them, use chmod again: **chmod 0700 /home/ubuntu/.ssh**

The .ssh folder contains the authorized\_keys file. Check its permissions with: **ls -ld authorized\_keys**

The file owner should have read and write permissions. To set them, use: **chmod 0600 /home/ubuntu/.ssh/authorized\_keys**

## Setting up backup & transfer:

Now that we have working SSH from server to server, let's create a new script file by typing: **nano backup.sh**

Once we have created the script type the following inside:

```
#!/bin/bash

export PGPASSWORD=<Your db user password>

pg_dump -h 127.0.0.1 -U <db username> <db name> | ssh -i
/home/ubuntu/farinemill ubuntu@128.214.255.93 "cat >
/home/ubuntu/backup/$(date +%Y-%m-%d).tar.gz"

unset PGPASSWORD
```

The next line `export PGPASSWORD=<Your db user password>` sets an environment variable called PGPASSWORD with the value being the password for the database user.

The `pg_dump` command is used to create a backup of a PostgreSQL database. It takes several arguments such as the host name (-h), username (-U), and the name of the database to be backed up.

The output of the `pg_dump` command is piped (|) to the `ssh` command, which securely transfers the backup data over the network to another server with IP address 128.214.255.93.

The ssh command uses the cat command to write the output to a file in the remote machine's directory /home/ubuntu/backup/ with a filename that includes the current date, formatted as YYYY-MM-DD, followed by the file extension .tar.gz.

Finally, the unset PGPASSWORD command is used to unset the PGPASSWORD environment variable, ensuring that the password is not accidentally leaked or misused elsewhere.

```
ubuntu@farinemill:~$ cd backup/
ubuntu@farinemill:~/backup$ ls
2023-03-24.tar.gz
ubuntu@farinemill:~/backup$
```

Now that we have working backup, let's automate it. We are going to use crontab for this job.

Users can create and edit their own crontab files using the "crontab" command. The crontab file consists of lines that define the commands or scripts to be executed, along with the schedule for when they should run. Each line specifies the minute, hour, day of the month, month, and day of the week on which the command or script should be run.

Crontab is commonly used for tasks such as backups, system maintenance, and scheduled scripts. It can also be used to run periodic commands or scripts that perform automated tasks or generate reports. The scheduling flexibility of crontab makes it a useful tool for automating many routine system administration tasks.

Switch to root and type: **crontab -e**

Inside your crontab type the following schedule:

```
# m H * * * command
0 0 * * * /home/ubuntu/skripteja/backup.sh
```

This creates a schedule where each day the server sends a full backup into the backup server at midnight.

We have a fully working backup now and we know how to schedule events. We can send the logs into the backup and logging server the same way by using crontab.

Although this is great, at some point our backup server is going to get clogged from constant logs and database backups so we need to do something about that. After all, full server is unable to receive any further logs or database backups.

fire up a root crontab in your backup server and type the following:

```
0 0 * * * find /home/ubuntu/backup -type f -name '*.tar.gz' -mtime +30 -delete
```

This will check your backup folder each midnight and **deletes** any backup that are older than 30 days (You may want to inform your client about this). This way the server always has space for new backups but at the same time has entire months' worth of dumps.

# Comprehensive security audit

For this course we are going to use LinPEAS script.

LinPEAS is a script designed to automate privilege escalation and system enumeration on Linux systems. It does not contain any new vulnerabilities itself, but rather it attempts to discover and exploit known vulnerabilities and misconfigurations in order to escalate privileges on the system.

LinPEAS will scan the system and analyze various system files, directories, and services to identify any potential vulnerabilities. It will then attempt to exploit these vulnerabilities in order to gain additional privileges on the system. Additionally, it will search for sensitive information such as passwords, SSH keys, and other credentials that may have been left exposed on the system.

LinPEAS is a very useful tool for auditing Linux systems and identifying potential security issues. However, it should only be used on systems where you have permission to run such tests, as using it on systems without proper authorization can be illegal and may result in serious consequences.

It is generally not recommended to run LinPEAS as the root user because LinPEAS is designed to identify and exploit potential vulnerabilities in the system. If LinPEAS is run as the root user, it will have full access to the system and could potentially cause damage or disruption to the system.

Running LinPEAS as a non-root user is generally safer because it will only have limited access to the system and will be unable to perform any actions that require elevated privileges. This can help to reduce the risk of unintended consequences, such as accidentally modifying or deleting important system files.

Additionally, running LinPEAS as a non-root user can help to ensure that the results of the scan are accurate and relevant to the specific user's permissions and privileges. If LinPEAS is run as the root user, it may identify potential vulnerabilities that are not relevant to a regular user's permissions or may not be exploitable without root access.

Overall, it is recommended to run LinPEAS as a non-root user whenever possible to minimize the risk of unintended consequences and ensure accurate and relevant results.



You can run LinPEAS by typing the following command:

```
curl -L https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh | sh
```

Alternatively, you could save it to a script file by typing nano linpeas.sh and typing inside the command above.

```

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 Privsec Checklist: https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist
LEGEND:
██████████ : 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 ...
17 808k 17 142k 0 0 175k 0 0:00:04 --:-- 0:00:04 175k
-----
Basic information
  
```

LinPEAS also contains exploit suggerester along with full system audit which might be useful information.

```

Searching Signature verification failed in dmesg
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#dmesg-signature-verification-failed
dmesg Not Found

Executing Linux Exploit Suggester
https://github.com/mzet-/linux-exploit-suggester
[+] [CVE-2022-2586] nft_object UAF

Details: https://www.openwall.com/lists/oss-security/2022/08/29/5
Exposure: probable
Tags: [ ubuntu=(20.04) ]{kernel;5.12.13}
Download URL: https://www.openwall.com/lists/oss-security/2022/08/29/5/1
Comments: kernel.unprivileged_usersns_clone=1 required (to obtain CAP_NET_ADMIN)

[+] [CVE-2021-4034] PwnKit

Details: https://www.qualys.com/2022/01/25/cve-2021-4034/pwnkit.txt
Exposure: probable
Tags: [ ubuntu=10|11|12|13|14|15|16|17|18|19|20|21 ],debian=7|8|9|10|11,fedora,manjaro
Download URL: https://codeload.github.com/berdav/CVE-2021-4034/zip/main

[+] [CVE-2021-3156] sudo Baron Samedit

Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
Exposure: probable
Tags: mint=19,[ ubuntu=18|20 ], debian=10
Download URL: https://codeload.github.com/blasty/CVE-2021-3156/zip/main

[+] [CVE-2021-3156] sudo Baron Samedit 2

Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
Exposure: probable
Tags: centos=6|7|8,[ ubuntu=14|16|17|18|19|20 ], debian=9|10
Download URL: https://codeload.github.com/worawit/CVE-2021-3156/zip/main
  
```



Lastly, for advanced administrators I would suggest getting into Snort.

Snort is a popular open-source Intrusion Detection System (IDS) that is used to monitor and analyze network traffic in real-time. Snort can be used to detect and prevent a wide variety of network-based attacks, including malware infections, Denial of Service (DoS) attacks, and intrusion attempts.

Snort can be configured to send email notifications when it detects a security event that matches a specific rule. This can be a useful feature for network administrators who need to be notified of potential security threats as soon as possible.

On top of everything, snort can also be configured to alert you inside your console every time your chosen event occurs. It is a complex tool to configure but at the same time, it is one of the more powerful tools out there.

Good Luck out there !

Guide written by: Aarne Salmi – AB8596



**ZZPP0920-IT 2023 T-03 Gang De Farine**