




Objectif

L'objectif de cette SAE est de comprendre le principe de quelques attaques, comment les analyser et mettre en place des solutions pour les bloquer. La SAE est organisée sur quatre parties principales à savoir deux parties en utilisant des VMs et deux autres sur équipements Cisco.

Avant-propos

Lisez cet énoncé en entier afin de ne pas vous lancer dans des manipulations et configurations inutiles, regardez notamment combien de temps est consacré pour chaque partie. Notant que durant la séance, vous serez constamment évalué.

0 – Les manipulations à exécuter sont indiquées de cette manière. 

0 – Les questions auxquelles il vous est demandé de répondre sont indiquées de cette manière. 

Déroulement de la SAE

Il est important de noter que les parties sur VM sont à réaliser **individuellement**, tandis que les parties pratiques doivent être accomplies en **Binômes**.

TRAORE Koba Groupe A

Partie 1 : Configuration Réseau Firewall (5 Points)



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 *Marionette*.



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

Le principe d'un pare-feu est de gérer les flux entrants et sortant d'un réseau. Ils fonctionnent sur la base de règles avec plusieurs paramètres qui permettent à des flux de passer et d'autres non.



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 ?

Le noyau Linux qui permet une telle opération est le NAT (Network Access Translation).



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



En vous basant de la commande *man iptables*, citez deux tables en décrivant le type de chaînes employées ?

☐ Table filter :



- Type : La table par défaut pour filtrer les paquets.
- Chaînes :
 - INPUT : Pour gérer le trafic entrant vers la machine locale.
 - OUTPUT : Pour gérer le trafic sortant généré localement.
 - FORWARD : Pour gérer le trafic traversant la machine (par exemple, routé).

☐ Table nat :

- **Type :** Utilisée pour la modification d'adresses (NAT, DNAT, SNAT).
- **Chaînes :**
 - **PREROUTING :** Pour modifier les paquets entrants avant le routage.
 - **POSTROUTING :** Pour modifier les paquets sortants après le routage.
 - **OUTPUT :** Pour modifier les paquets générés localement avant l'envoi.

En vous servant de la commande remplissez les deux tableaux ci-dessous :

- 👉 Dézipper le fichier zip donner durant la séance et récupérer le fichier (**firewell.mar**).
- 👉 Après le lancement de Marionnette, lancer le projet à partir du fichier. mar. Vous observez une architecture réseau comportant une machine **www**, un **switch** et un équipement **G1** constitué d'un réseau public ayant l'adresse **195.83.80.0/24** voir la Figure 1.
- 👉 En vous servant de l'interface graphique de Marionnette, 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**.
- ✎ En vous servant de la commande **ip route**, configuré les chemins dans la machine de chaque client et vérifier la table de routage. Ensuite, vérifié les pings entre les machines ? N'oubliez pas d'activer le forward.

Les opérations servant à gérer les chaines entières.

Commande	Syntaxe	Rôle
<i>iptables -N test</i>	-N	Crée une nouvelle chaîne utilisateur nommée test.
<i>iptables -X test</i>	-X	Supprime la chaine test.
<i>iptables -P FORWARD DROP</i>	-P	Politique par default sur FORWARD
<i>iptables -L INPUT</i> <i>iptables -L</i> <i>iptables -nL</i>	-L -nL	Liste les règles de la chaîne INPUT. Affiche toutes les règles en vigueur Affichage très détaillé
<i>iptables -F INPUT</i> <i>iptables -F</i>	-F	Supprime toutes les règles de la chaîne INPUT. Supprime toutes les règles de partout

Manipulé les règles à l'intérieur d'une chaine.

Commande	Syntaxe	Rôle
<i>iptables -A INPUT -s 0/0 -j DENY</i>	-A	Ajoute une règle en fin de chaîne pour bloquer tout le trafic.
<i>iptables -I INPUT 1 -s 127.0.0.1</i>	-I	Met une règle en première position dans la chaîne INPUT.
<i>iptables -R INPUT 1 -s 192.168.0.1</i>	-R	Remplace la règle numéro 1 de la chaîne INPUT.
<i>iptables -D INPUT 1</i>	-D	Supprime la règle numéro 1 de la chaîne INPUT
<i>iptables -D INPUT -s 127.0.0.1 -p icmp -j DENY</i>	-D	Supprime une règle spécifique de la chaîne INPUT.

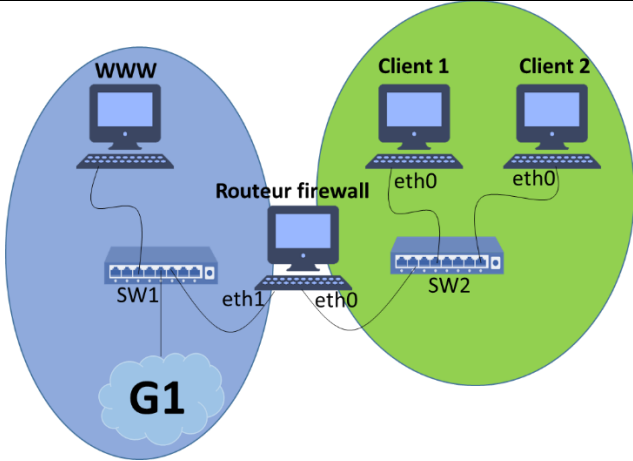


Figure1. Architecture réseau étudiée.

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

```
echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
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 ?

✎ **Déjà fait avant iptables -t nat -A POSTROUTING -s 192.168.20.0/24 -o eth1 -j MASQUERADE**
Vérifier que le camouflage a été bien activé ?

👍 Appliquer la commande suivante *iptables -A FORWARD -s 192.168.20.0/24 -j ACCEPT*.

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

✎ **Le rôle de cette commande est d'accepter de laisser passer tous les flux en provenance du réseau des clients.**

Pour vérifier le bon fonctionnement, essayez de joindre l'adresse publique 195.83.80.10. Ensuite, lancer le navigateur en utilisant la commande *epiphany http://195.83.80.10/whatismyip.php*

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

Après avoir entré cette commande sur un des clients ici le 2 l'adresse IP que me ressort epiphany et l'adresse IP de l'eth1 du firewall car ici le réseau privé à pour NAT l'interface qui est tourné vers l'extérieur du réseaux. Toutes les adresse du réseau privé deviennent l'eth1 du firewall pour pouvoir naviguez sur internet.

👍 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 a de perte dans l'interface de bouclage du routeur. **Commande : ping -c 5 127.0.0.1**

✎ En utilisant les deux tableaux précédents, crée une nouvelle chaine appelée **LOG_DROP** pour à la fois rejeter les paquets et enregistrer dans les Logs ?

✎ Appliquer un filtre sur la chaine d'entrer INPUT. *iptables -A INPUT -p icmp -s 127.0.0.1 -j LOG_DROP*

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

👍 Sur un deuxième terminal taper la commande *tail -f /var/log/messages*.

✎ Sur le premier terminal relancer le *ping* et observer les messages qui s'affichent sur le terminal 2. Expliquez ?
Les pings sont bloqués directement ceci est du à la règle précédente qui drop tout protocole ICMP

✎ Supprimer la règle numéro 1 sur la chaine INPUT ?

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

👍 Nous désirons à présent autorisé toutes les connexions sortantes sauf celle de **http depuis votre réseau privé vers le réseau public**. Editer le fichier */etc/resolv.conf* des deux machines clients. Supprimer toute ligne en gardant uniquement *nameserver 195.83.80.3*.

✎ A partir des postes clients, essayer de vous connecter avec un navigateur *epiphany* sur **http://miashs-www.u-ga.fr/~adamj**. Vérifier l'état des différentes chaines sur le routeur firewall.

✎ Trouver des commandes pour **interdire en TCP** les paquets venant et vers le **port 80** (réponse et requête http)

Partie 2: Man in the Middle et ARP Spoofing.

La partie 2 de la SAE traitera les attaques ARP Spoofing et man in the middle. Une plateforme basée sur virtuel Box est mise à disposition pour les étudiants, dont l'objectif de simuler et visualiser les attaques précédentes. La plateforme appelé **MI-LXC** représente un ensemble de réseau interconnecté entre eux développé par nos collègues à l'université de Lyon. Un Grand merci au professeur **François Lesueur** pour le partage du programme.

Une fois le fichier **OVA** téléchargé, ouvrez **Virtual Box**, puis dans le menu **Fichier, sélectionnez Importer un appareil virtuel**. Choisissez le fichier **OVA** et patientez quelques minutes le temps de la création de la machine

virtuelle. Avant de lancer la VM, il peut être nécessaire de diminuer la RAM allouée. Par défaut, la VM a 3GO : si vous avez 4GO sur votre machine physique, il vaut mieux diminuer à 2GO, voire 1.5GO pour la VM (la VM devrait fonctionner de manière correcte toujours).

👉 L'infrastructure déployée simule plusieurs postes dont un SI d'entreprise (firewall, DMZ, intranet, authentification centralisée, serveur de fichiers, quelques postes de travail interne de l'entreprise Target), une machine d'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 SAE.

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

👉 MI-LXC est déjà installé et l'infrastructure déployée, il faut avec un terminal aller dans le dossier **/root/mi-lxc**. Pour démarrer l'infrastructure, tapez **./mi-lxc.py start**. Une fois l'environnement démarré, la seule machine à utiliser est évidemment celle du hacker, affichable avec la commande **./mi-lxc.py display isp-a-hacker**.

NB. Si la souris reste bloquée dans cette fenêtre, appuyez sur CTRL+SHIFT pour la libérer. Un tuto vidéo de démarrage est proposé ici.

Ci-joint un tableau de commande qui vous sera utile pour la réalisation de votre simulation. Je vous recommande notamment de visualiser la topologie réseau qui sera donnée en Figure 2.

Commande	Description	Utilisation
print	Génère la cartographie du réseau	<code>./mi-lxc.py print</code>
attach	Permet d'avoir un shell sur une machine	<code>./mi-lxc.py attach root@target-commercial</code>
display	Lance un serveur X sur la machine cible	<code>./mi-lxc.py display target-commercial</code>

👉 *NB. Vous devez être dans mi-lxc pour exécuter les commandes.*

👉 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.

Machine	Adresses IPV4	Adresses MAC	Remarque
Target-router	100.80.0.1/16	5a :05 :b4 :9f :6c :50	Routeur principal
Target-admin	100.80.0.4/16	D6 :7E :44 :5D :1B :32	Poste admin
Target-dmz	100.80.1.2/16	B2 :19 :48 :da :e6 :19	Serveur DMZ
Target-commercial	100.80.0.2/16	8e :39 :c9 :7a :ba :9f	Réseau commercial
Target-dev	100.80.0.3/16	02 :00 :c2 :f6 :df :68	Poste développeurs
Target-intranet	100.80.0.5/16	E2 :6e :2b :3e :1c :14	Intranet interne
Target-ldap	100.80.0.10/16	Fa :94 :b0 :0e :89 :e7	Serveur LDAP
Target-filer	100.80.0.6/16	7e :96 :37 :70 :92 :14	Serveur de fichiers

J'ai utilisé la commande `./mi-lxc.py attach root@target-<différente machine>`

👉 Nous allons à présent tenter de mettre en œuvre une attaque dite ARP Spoofing. Rappelons tout d'abord ce qu'est une attaque ARP Spoofing. APR pour "ARP Poison Routing" selon le document APR décrit une attaque de reniflage (sniffing) qui se déroule en deux moment : empoisonnement de la table ARP qui livre les paquets et routage des paquets vers la bonne destination. Vous avez vu durant votre parcours que la capture de paquet locale sur un environnement commuté fournissait des résultats uniquement pour le trafic livré à l'interface elle-même soit le trafic unicast à destination de la machine elle-même, le trafic Broadcast et multicast transférés d'emblée par les commutateurs à travers tous ses ports. Dans un environnement LAN commuté, le commutateur transfère directement le trafic en fonction de l'adresse MAC de destination encodée dans les trames Ethernet. Ce sont les hôtes d'origine et de destination qui encodent ces adresses sur base d'un processus ARP. Une attaque APR (ARP Poison Routing) est une attaque d'interception (MiTM) du trafic qui consiste pour le pirate à empoisonner le cache

ARP des victimes avec sa propre adresse MAC comme adresse physique de livraison pour les adresses IP des attaquées (voir Figure3) . On peut aussi classer l'attaque dans la catégorie des attaques par usurpation (spoofing).

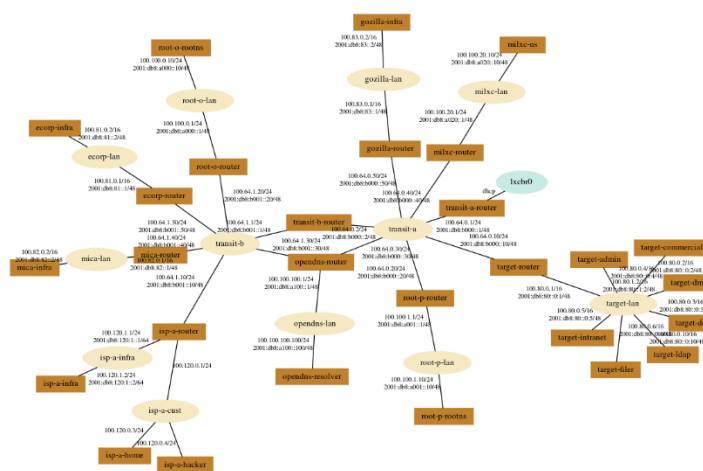


Figure 2 : Topologie réseau sur MI-LXC.

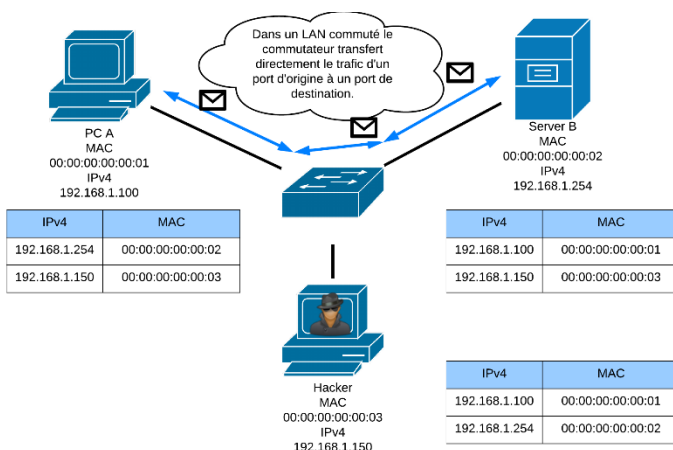


Figure 3 : Principe d'attaque ARP Spoofing.

👉 Nous allons considérer que pour une raison inconnue, le **développeur de l'entreprise** souhaite s'attaquer à l'**administrateur système** afin de lui dérober ses **identifiants Intranet**. Le but de l'attaque sera donc de faire en sorte que l'administrateur système entre ses identifiants sur un serveur web qui n'est pas l'intranet.

👉 Pour réaliser l'attaque, ouvrez 2 terminaux et connectez-vous sur l'ordinateur du développeur et de l'administrateur système. Lancez wireshark dans les deux VMs.

✎ Vérifier que les **caches arp** des deux VMs sont vides. **Ping les deux machines** ensuite revérifier les caches arp.

👉 Vider les **caches arp** une autre fois dans les deux machines et compris l'intranet. Pour réaliser l'attaque, nous allons utiliser l'outil **arp spoof** 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'administrateur 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 **arp spoof** avec les bons arguments afin de tromper l'ordinateur du admin pour qu'il pense que l'intranet est la machine du développeur.

✎ Voyez-vous la table ARP changer ? A votre avis, pourquoi ne change-t-elle pas ?

✎ Elle ne change pas car il a peut-être une protection Arp car admin avait plus de table Arp parce que j'ai tout supprimé donc n'avait aucun moyen d'ignorer.

👉 Stoppez **arp spoof** à l'aide de **CTRL+C** (ça peut mettre un peu de temps, l'outil essaie de réannoncer la véritable MAC avant de s'éteindre). Depuis la machine du sysadmin, faites un **curl** vers le serveur de

✎ L'intranet (<http://www.target.milxc>) puis relancez le **watch**.

✎ Une nouvelle entrée doit apparaître dans la table ARP. Laquelle ? Pourquoi ?

✎ La nouvelle entrée dans la table Arp est le dmz car en utilisant la commande curl pour rejoindre un site web le dmz fait office d'intermédiaire pour les requêtes http.

✎ Relancez **arp spoof** sur la machine du développeur. Est-ce que la table ARP sur la machine du sysadmin a changé ? Expliquez ? Vérifier que l'ordinateur du développeur reçoit les échanges ICMP provenant du serveur intranet ?

👉 A présent, nous allons réaliser une attaque dite de l'homme du milieu et tenter d'intercepter et d'altérer le trafic sur le réseau. Relancez l'**attaque d'ARP Spoofing** comme vu précédemment. En complément, sur la machine du développeur, lancez l'outil **urllibsnarf** qui vous affichera les **requêtes HTTP** entrantes.

✎ Depuis la machine du sysadmin, faites un **curl** sur l'intranet. Obtenez-vous le résultat escompté ? Comment le savez-vous ?



Je ne reçois rien sur la commande `urlsnarf` en effectuant la commande `curl` sur `target-admin`. Je le sais car rien ne s'affiche.

Afin de déboguer et comprendre l'origine du problème, nous allons capturer le trafic avec **tcpdump**. Sur la machine du développeur, lancez la commande suivante permettant de capturer uniquement **les paquets sur le port HTTP (TCP/80)** : **`tcpdump 'port http'`**. Puis relancez le **curl**. Enfin, faites **CTRL+C** sur le **tcpdump**.



Analysez le résultat du **tcpdump**. Est-ce que des paquets HTTP ont été interceptés sur notre machine ? Quelle est l'IP de destination ? Quelle est l'IP de notre machine ?

Par défaut, le noyau Linux ignore les paquets qui ne lui sont pas adressés. Peut-être que cela pourrait expliquer en partie pourquoi nous n'avons pas le comportement espéré. A l'aide **d'ifconfig**, ajoutez une nouvelle interface réseau afin de résoudre le problème.

Les paquets http ont bien été intercepter depuis notre machine avec pour machine dst : 10.80.0.5 et l'IP de notre machine est 10.80.0.3.

NB: pour ajouter une nouvelle interface réseau avec **ifconfig**, vous pouvez utiliser la commande suivante: **`ifconfig eth0:bad <ip> netmask <mask> up`** en > remplaçant les champs ip et mask par les bonnes valeurs.



Vérifiez à l'aide de **ifconfig** que votre interface est correctement configurée. Relancez **curl**. Que se passe-t-il ? Tentez d'expliquer pourquoi, malgré l'ARP spoofing, vous aviez quand même une réponse lors de la question 8.

Après avoir relancer curl on retrouve exactement les mêmes résultats que précédemment. Je pense que nous avons eu une réponse précédemment car les machines se trouve dans le même réseau.



Comme vu en cours, **le protocole HTTPS** a plusieurs buts. Le premier est le chiffrement, ce qui fait qu'une attaque de type Man in the Middle est bien plus difficile puisque le contenu entre le client et le serveur est chiffré. Le second, que nous allons étudier dans ce TP, est la capacité d'attester que le site web visité est le bon.



Depuis la machine "**isp-a-home**", ouvrez un navigateur pour vous connecter à **`http://www.target.milxc`** (**`./mi-lxc.py display isp-a-home`**). Vous accédez à une page **Dokuwiki**, qui est bien la page attendue.



Imaginons que votre résolveur DNS ait été compromis et que l'entrée DNS pour **`www.target.milxc`** ait été modifiée pour vous envoyer **sur un site frauduleux** ! Nous allons simuler cette action en ayant pour objectif que le navigateur, lorsqu'il souhaite se connecter à l'URL **`http://www.target.milxc`**, arrive en fait sur la machine "**ecorp-infra**".



Pour simuler la compromission, vous allez altérer l'enregistrement DNS pour **`target.milxc`** dans la zone du **TLD `.milxc`**. Sur la machine "**milxc-ns**" :

- Altération de **`/etc/nsd/milxc.zone`** pour diriger les requêtes DNS pour **`target.milxc`** vers **`100.81.0.2`** (appartenant à ecorp). Faites attention à IPv6, je vous conseille de **supprimer les enregistrements IPv6 pour être tranquilles**.
- Puis **`service nsd restart`** (le DNS de ecorp est déjà configuré pour répondre aux requêtes pour **`target.milxc`**)



Tentez à nouveau de vous rendre sur **`www.target.milxc`** à l'aide votre navigateur. Que se passe-t-il ? Est-ce transparent ?

Quand on se rend a nouveau sur `www.target.milxc` on tombe sur Never gonna give you up qui est aussi le même rickrolled



Nous constatons ainsi le cas d'attaque que nous souhaiterions détecter : un utilisateur sur "**isp-a-home**" qui, en tapant l'URL **`www.target.milxc`**, arrive en fait sur un autre service que celui attendu. Remettez le système en bon ordre de marche pour continuer (remettre la bonne IP **`100.80.1.2`** dans la zone DNS).

Screen de la Parie 1:

Screen 1 client1 → client2:

```
Client1 (debian-wheezy-08367)
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.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
[0 root@Client1 ~]$ nano /etc/network/interfaces
[0 root@Client1 ~]$ ping 192.168.20,10
connect: Network is unreachable
[2 root@Client1 ~]$ ifup eth0
[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=0.859 ms
64 bytes from 192.168.20.10: icmp_req=2 ttl=64 time=0.524 ms
64 bytes from 192.168.20.10: icmp_req=3 ttl=64 time=0.475 ms
64 bytes from 192.168.20.10: icmp_req=4 ttl=64 time=0.532 ms
64 bytes from 192.168.20.10: icmp_req=5 ttl=64 time=0.539 ms
^C
--- 192.168.20.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4029ms
rtt min/avg/max/mdev = 0.475/0.585/0.859/0.141 ms
[0 root@Client1 ~]$
```

Screen 2 client 2 → client1:

```
Client2 (debian-wheezy-08367)
Last login: Wed Dec  4 17:13:01 CET 2024 on tty0

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.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
[0 root@Client2 ~]$ nano /etc/network/interfaces
[0 root@Client2 ~]$ ifup eth0
[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=0.841 ms
64 bytes from 192.168.20.1: icmp_req=2 ttl=64 time=0.469 ms
64 bytes from 192.168.20.1: icmp_req=3 ttl=64 time=0.480 ms
64 bytes from 192.168.20.1: icmp_req=4 ttl=64 time=0.457 ms
64 bytes from 192.168.20.1: icmp_req=5 ttl=64 time=1.03 ms
64 bytes from 192.168.20.1: icmp_req=6 ttl=64 time=0.411 ms
^C
--- 192.168.20.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5028ms
rtt min/avg/max/mdev = 0.411/0.616/1.038/0.236 ms
[0 root@Client2 ~]$
```

Screen 3 client1 → firewall eth0 :

```
Client1 (debian-wheezy-08367)
64 bytes from 192.168.20.10: icmp_req=4 ttl=64 time=0.532 ms
64 bytes from 192.168.20.10: icmp_req=5 ttl=64 time=0.539 ms
^C
--- 192.168.20.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4029ms
rtt min/avg/max/mdev = 0.475/0.585/0.859/0.141 ms
[0 root@Client1 ~]$ ping 192.168.20.30
PING 192.168.20.30 (192.168.20.30) 56(84) bytes of data.
64 bytes from 192.168.20.30: icmp_req=1 ttl=64 time=1.01 ms
64 bytes from 192.168.20.30: icmp_req=2 ttl=64 time=0.478 ms
64 bytes from 192.168.20.30: icmp_req=3 ttl=64 time=0.523 ms
64 bytes from 192.168.20.30: icmp_req=4 ttl=64 time=0.520 ms
64 bytes from 192.168.20.30: icmp_req=5 ttl=64 time=0.478 ms
64 bytes from 192.168.20.30: icmp_req=6 ttl=64 time=0.243 ms
64 bytes from 192.168.20.30: icmp_req=7 ttl=64 time=0.700 ms
64 bytes from 192.168.20.30: icmp_req=8 ttl=64 time=0.485 ms
64 bytes from 192.168.20.30: icmp_req=9 ttl=64 time=0.525 ms
64 bytes from 192.168.20.30: icmp_req=10 ttl=64 time=0.592 ms
^C
--- 192.168.20.30 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9035ms
rtt min/avg/max/mdev = 0.243/0.555/1.015/0.190 ms
[0 root@Client1 ~]$
```

Screen 4 client2 → firewall eth0 :

```
Client2 (debian-wheezy-08367)
64 bytes from 192.168.20.1: icmp_req=4 ttl=64 time=0.457 ms
64 bytes from 192.168.20.1: icmp_req=5 ttl=64 time=1.03 ms
64 bytes from 192.168.20.1: icmp_req=6 ttl=64 time=0.411 ms
^C
--- 192.168.20.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5028ms
rtt min/avg/max/mdev = 0.411/0.616/1.038/0.236 ms
[0 root@Client2 ~]$ ping 192.168.20.30
PING 192.168.20.30 (192.168.20.30) 56(84) bytes of data.
64 bytes from 192.168.20.30: icmp_req=1 ttl=64 time=1.39 ms
64 bytes from 192.168.20.30: icmp_req=2 ttl=64 time=0.786 ms
64 bytes from 192.168.20.30: icmp_req=3 ttl=64 time=0.529 ms
64 bytes from 192.168.20.30: icmp_req=4 ttl=64 time=0.580 ms
64 bytes from 192.168.20.30: icmp_req=5 ttl=64 time=0.538 ms
64 bytes from 192.168.20.30: icmp_req=6 ttl=64 time=0.578 ms
64 bytes from 192.168.20.30: icmp_req=7 ttl=64 time=0.521 ms
64 bytes from 192.168.20.30: icmp_req=8 ttl=64 time=0.475 ms
64 bytes from 192.168.20.30: icmp_req=9 ttl=64 time=0.569 ms
64 bytes from 192.168.20.30: icmp_req=10 ttl=64 time=0.506 ms
^C
--- 192.168.20.30 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9044ms
rtt min/avg/max/mdev = 0.475/0.647/1.394/0.262 ms
[0 root@Client2 ~]$
```

Screen 5 client2 → firewall eth1 :

```
Client2 (debian-wheezy-08367)
PING 192.83.80.253 (192.83.80.253) 56(84) bytes of data.
From 192.168.20.30 icmp_seq=1 Destination Net Unreachable
From 192.168.20.30 icmp_seq=2 Destination Net Unreachable
^C
--- 192.83.80.253 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1000ms

[1 root@Client2 ~]$ ping 192.83.80.254
PING 192.83.80.254 (192.83.80.254) 56(84) bytes of data.
64 bytes from 192.83.80.254: icmp_req=1 ttl=64 time=0.872 ms
64 bytes from 192.83.80.254: icmp_req=2 ttl=64 time=0.451 ms
64 bytes from 192.83.80.254: icmp_req=3 ttl=64 time=0.518 ms
64 bytes from 192.83.80.254: icmp_req=4 ttl=64 time=0.610 ms
^C
--- 192.83.80.254 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.451/0.612/0.872/0.162 ms
```

Screen 6 client1 → firewall eth1 :


```
Client1 (debian-wheezy-08367)
^C
--- 192.168.20.30 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9035ms
rtt min/avg/max/mdev = 0.243/0.555/1.015/0.190 ms
[0 root@Client1 ~]$ route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
default          192.168.20.30  0.0.0.0         UG    0      0      0 eth0
192.168.20.0     *              255.255.255.0   U     0      0      0 eth0
[0 root@Client1 ~]$ nano /etc/network/interfaces
[0 root@Client1 ~]$ ping 192.83.80.254
PING 192.83.80.254 (192.83.80.254) 56(84) bytes of data:
64 bytes from 192.83.80.254: icmp_req=1 ttl=64 time=0.857 ms
64 bytes from 192.83.80.254: icmp_req=2 ttl=64 time=0.468 ms
64 bytes from 192.83.80.254: icmp_req=3 ttl=64 time=0.455 ms
64 bytes from 192.83.80.254: icmp_req=4 ttl=64 time=0.451 ms
64 bytes from 192.83.80.254: icmp_req=5 ttl=64 time=0.473 ms
64 bytes from 192.83.80.254: icmp_req=6 ttl=64 time=0.402 ms
^C
--- 192.83.80.254 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5021ms
rtt min/avg/max/mdev = 0.402/0.517/0.857/0.155 ms
[0 root@Client1 ~]$
```

FW Règles :

```
RFW (debian-wheezy-08367)
[0 root@RFW ~]$ iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
    0    0 ACCEPT    all  --  any    any     195.83.80.0/24  192.168.20.0/24
    0    0 ACCEPT    all  --  any    any     192.168.20.0/24  195.83.80.0/24

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
[0 root@RFW ~]$ iptables -t nat -A POSTROUTING -s 192.168.20.0/24 -o eth1 -j MASQUERADE
[0 root@RFW ~]$ iptables -L -v
Chain INPUT (policy ACCEPT 16 packets, 1352 bytes)
 pkts bytes target    prot opt in     out     source         destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
    5   420 ACCEPT    all  --  any    any     195.83.80.0/24  192.168.20.0/24
   147 12348 ACCEPT    all  --  any    any     192.168.20.0/24  195.83.80.0/24

Chain OUTPUT (policy ACCEPT 16 packets, 1352 bytes)
 pkts bytes target    prot opt in     out     source         destination
[0 root@RFW ~]$
```

Client1 → www :

```

Client1 (debian-wheezy-08367)
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out    source            destination

Chain OUTPUT (policy ACCEPT 958 packets, 80076 bytes)
pkts bytes target      prot opt in     out    source            destination

[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 ---
5 packets transmitted, 0 received, 100% packet loss, time 4007ms

[1 root@Client1 ~]$ ping 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=1.21 ms
64 bytes from 195.83.80.10: icmp_req=2 ttl=63 time=0.801 ms
64 bytes from 195.83.80.10: icmp_req=3 ttl=63 time=0.858 ms
^C
--- 195.83.80.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 0.801/0.959/1.219/0.186 ms
[0 root@Client1 ~]$

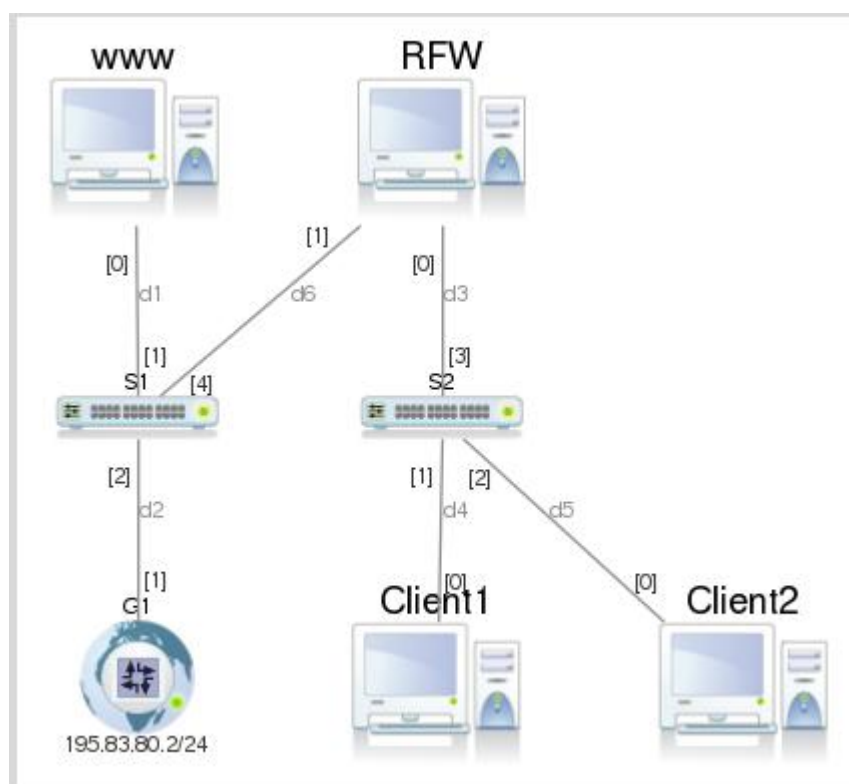
```

Client 2 → www :

```

[1 root@Client2 ~]$ ping 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=1.26 ms
64 bytes from 195.83.80.10: icmp_req=2 ttl=63 time=0.926 ms
^C
--- 195.83.80.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.926/1.096/1.266/0.170 ms
[0 root@Client2 ~]$ ping 195.83.80.10

```



Desactive icmp all redirect :

```

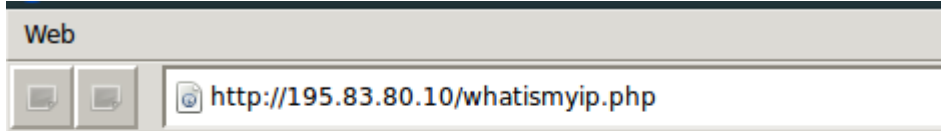
RFW (debian-wheezy-08367)
GNU nano 2.2.6      File: /etc/sysctl.conf      Modified

# Do not send ICMP redirects (we are not a router)
net.ipv4.conf.all.send_redirects = 0

```

```
[1 root@RFW ~]$ sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.all.send_redirects = 0
[0 root@RFW ~]$
```

Sur un des clients (2) :



Votre adresse : 195.83.80.254

Sur firewall ping localhost 127.0.0.1 :

```

Chain OUTPUT (policy ACCEPT 16 packets, 1352 bytes)
 pkts bytes target    prot opt in     out     source                   destination
[0 root@RFW ~]$ echo 0 > proc/sys/net/ipv4/conf/all/send_redirects
-bash: proc/sys/net/ipv4/conf/all/send: No such file or directory
[1 root@RFW ~]$ echo 0 > proc/sys/net/ipv4/conf/all/send_redirects
-bash: proc/sys/net/ipv4/conf/all/send: No such file or directory
[1 root@RFW ~]$ nano /etc/sysctl.conf
[0 root@RFW ~]$ echo 0 > proc/sys/net/ipv4/conf/all/send_redirects
-bash: proc/sys/net/ipv4/conf/all/send: No such file or directory
[1 root@RFW ~]$ sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.all.send_redirects = 0
[0 root@RFW ~]$ 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=0.353 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_req=4 ttl=64 time=0.038 ms
^C
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 301ms
rtt min/avg/max/mdev = 0.038/0.117/0.353/0.136 ms
[0 root@RFW ~]$
```

NOUVELLE CHAÎNE LOG8DROP /

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

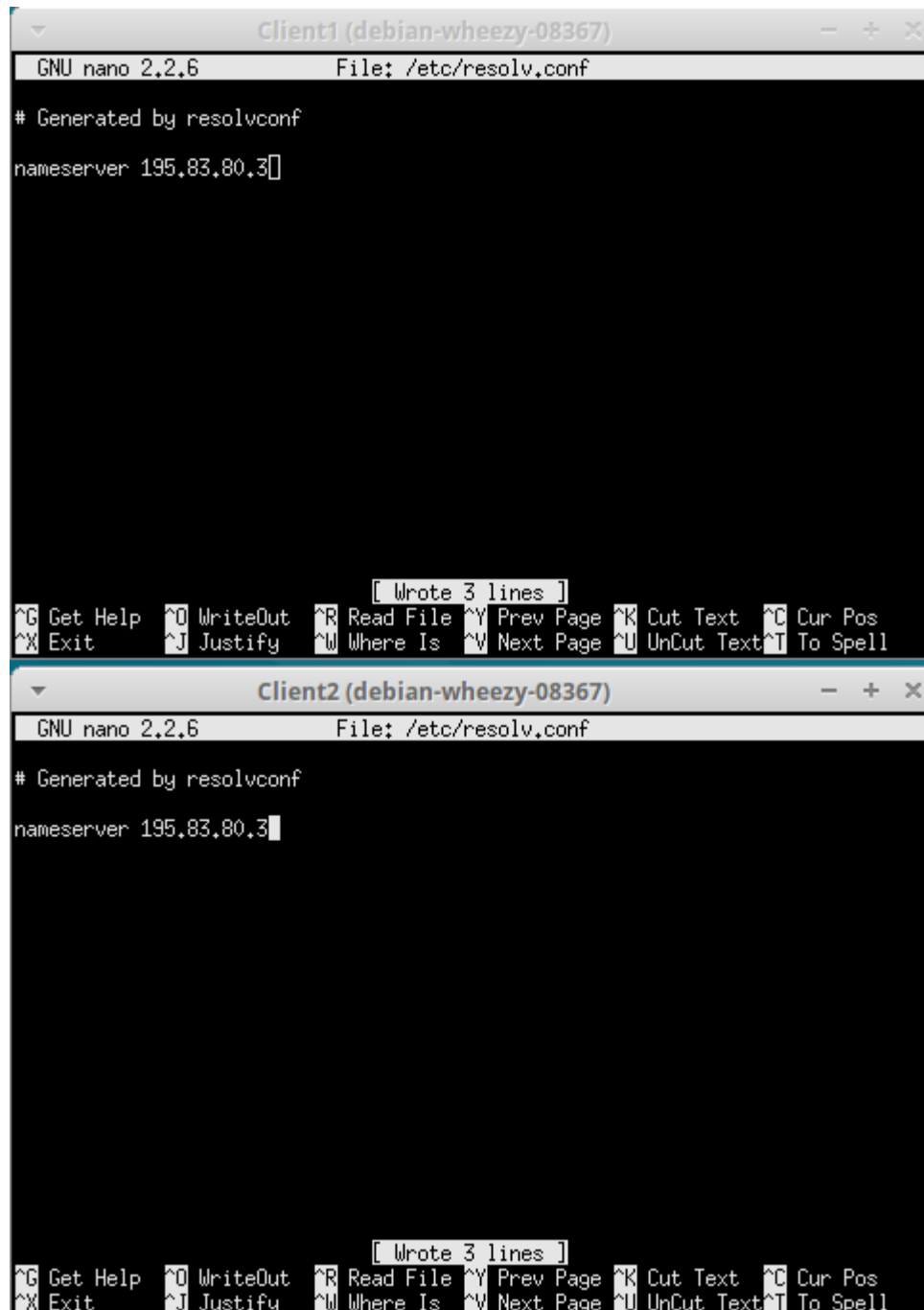
Filtre entré INPUT :

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

Afficher la table INPUT avec les numéros d'arriver :

```
[0 root@RFW ~]$ iptables -L INPUT --line-numbers
Chain INPUT (policy ACCEPT)
num target      prot opt source      destination
1 LOG_DROP      icmp -- localhost  anywhere
[0 root@RFW ~]$
```


Nameserver sur etc/resolv.conf :



The image shows two terminal windows side-by-side, both titled 'Client1 (debian-wheezy-08367)' and 'Client2 (debian-wheezy-08367)'. Each window displays the nano 2.2.6 editor editing the file /etc/resolv.conf. The content of the file is: # Generated by resolvconf, nameserver 195.83.80.3. A status bar at the bottom of each window indicates '[Wrote 3 lines]' and provides keyboard shortcuts for various nano editor functions.

```
Client1 (debian-wheezy-08367)
GNU nano 2.2.6      File: /etc/resolv.conf

# Generated by resolvconf
nameserver 195.83.80.3

[ Wrote 3 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell

Client2 (debian-wheezy-08367)
GNU nano 2.2.6      File: /etc/resolv.conf

# Generated by resolvconf
nameserver 195.83.80.3

[ Wrote 3 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

Connexion sortante http rejeté le reste accepter:



A terminal snippet showing a root user at a host named RFW executing the command: iptables -A OUTPUT -p tcp --dport 80 -j REJECT.

```
[0 root@RFW ~]$ iptables -A OUTPUT -p tcp --dport 80 -j REJECT
[0 root@RFW ~]$
```

Connection refusé depuis client2 :


```
[0 root@Client2 ~]$ epiphany sur http: //miashs-www.u-ga.fr/~adamj
** (epiphany-browser:1748): WARNING **: Could not connect: Connection refused
** (epiphany-browser:1748): WARNING **: Could not connect: Connection refused
** (epiphany-browser:1748): WARNING **: Could not connect: Connection refused
** (epiphany-browser:1748): WARNING **: Could not connect: Connection refused
** (epiphany-browser:1748): WARNING **: Could not connect: Connection refused
** (epiphany-browser:1748): WARNING **: Could not connect: Connection refused
** (epiphany-browser:1748): WARNING **: Could not connect: Connection refused
** (epiphany-browser:1748): WARNING **: Unable to start Zeroconf subsystem
** (epiphany-browser:1748): WARNING **: Could not connect: Connection refused
```

Vérification des différentes chaines :

```
[0 root@RFW ~]$ iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 195.83.80.0/24 192.168.20.0/24
2 ACCEPT all -- 192.168.20.0/24 195.83.80.0/24
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 REJECT tcp -- anywhere anywhere tcp dpt:http reject-with
icmp-port-unreachable
Chain LOG_DROP (0 references)
num target prot opt source destination
1 LOG all -- anywhere anywhere LOG level warning prefix
"DROP ICMP: "
2 DROP all -- anywhere anywhere
[0 root@RFW ~]$
```

Refus requêtes et réponses http :

```
[0 root@RFW ~]$ iptables -A INPUT -p tcp --dport 80 -j DROP
[0 root@RFW ~]$ iptables -A OUTPUT -p tcp --sport 80 -j DROP
[0 root@RFW ~]$
```



Oops! It was not possible to show this website

The website at **http://miashs-www.u-ga.fr/~adamj** seems to be unavailable. The precise error was:

Cannot connect to destination (miashs-www.u-ga.fr)

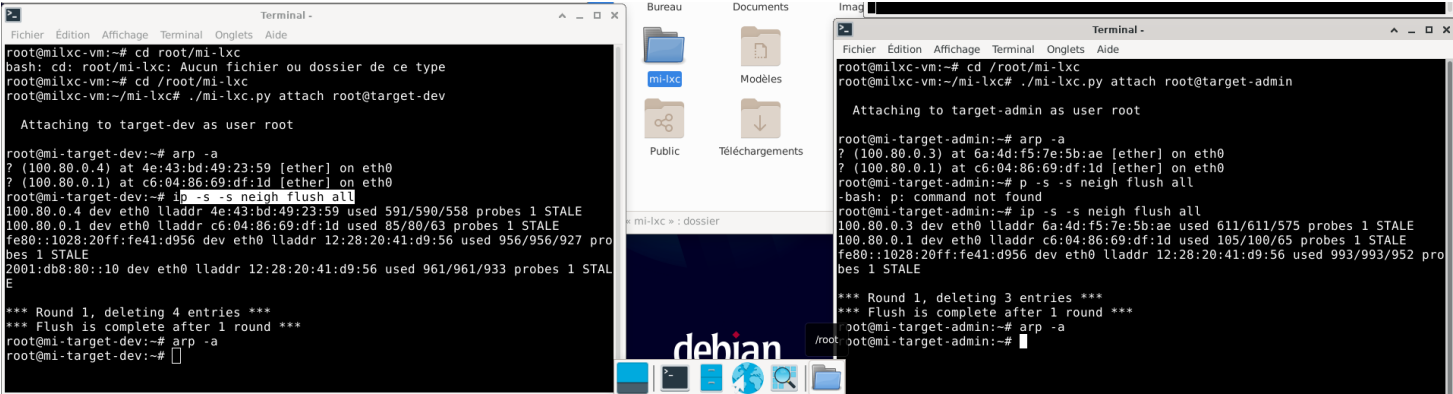
It could be temporarily switched off or moved to a new address. Don't forget to check that your internet connection is working correctly.

Try again

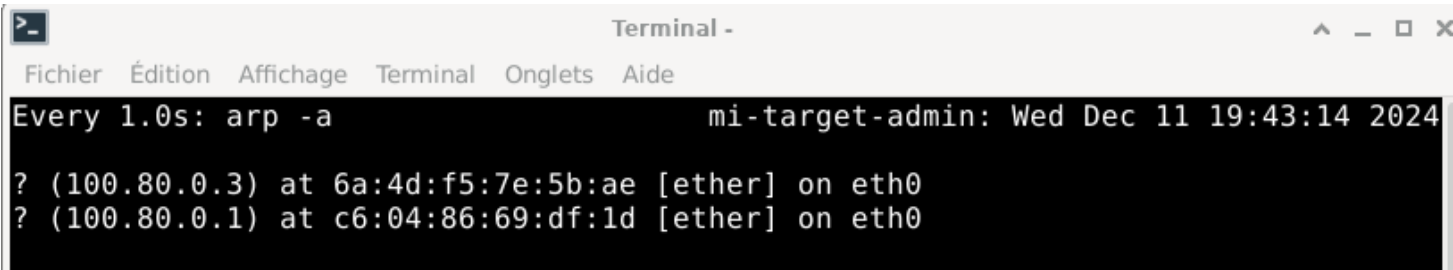
Screen de la Partie 2 :



Flush arp



Arspooof
admin



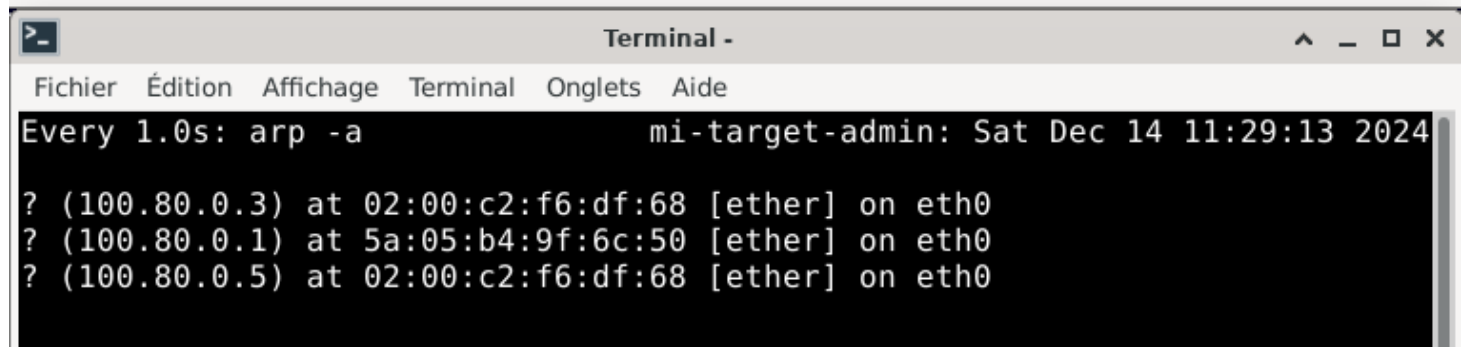
Dev

```
root@mi-target-dev:~# arpspoof -i eth0 -t 100.80.0.4 100.80.0.5
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
6a:4d:f5:7e:5b:ae 4e:43:bd:49:23:59 0806 42: arp reply 100.80.0.5 is-at 6a:4d:f5:7e:5b:ae
```

Nouvelle entrée DMZ :

```
? (100.80.1.2) at b2:19:48:da:e6:19 [ether] on eth0
```

MAC de dev pour ip intranet :



```
Every 1.0s: arp -a                                mi-target-admin: Sat Dec 14 11:29:13 2024
? (100.80.0.3) at 02:00:c2:f6:df:68 [ether] on eth0
? (100.80.0.1) at 5a:05:b4:9f:6c:50 [ether] on eth0
? (100.80.0.5) at 02:00:c2:f6:df:68 [ether] on eth0
```

Urlsnarf :

```
root@mi-target-dev:~# urlsnarf
urlsnarf: listening on eth0 [tcp port 80 or port 8080 or port 3128]
```

Curl sur intranet :

```
root@mi-target-admin:~# curl -4 http://100.80.0.5
```

Tcpdump de dev :

```

root@mi-target-dev:~# tcpdump -i eth0 port 80
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C15:42:31.397247 IP 100.80.0.4.41578 > 100.80.0.5.http: Flags [S], seq 1209985088, win 64240, options [mss 1460,sackOK,TS val 999149091 ecr 0,nop,wscale 7], length 0
15:42:31.397281 IP 100.80.0.4.41578 > 100.80.0.5.http: Flags [S], seq 1209985088, win 64240, options [mss 1460,sackOK,TS val 999149091 ecr 0,nop,wscale 7], length 0
15:42:31.397324 IP 100.80.0.4.41578 > 100.80.0.5.http: Flags [.], ack 2440474071, win 502, options [nop,nop,TS val 999149091 ecr 3856171410], length 0
15:42:31.397328 IP 100.80.0.4.41578 > 100.80.0.5.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 999149091 ecr 3856171410], length 0
15:42:31.397377 IP 100.80.0.4.41578 > 100.80.0.5.http: Flags [P.], seq 0:74, ack 1, win 502, options [nop,nop,TS val 999149091 ecr 3856171410], length 74: HTTP:

```

Eth0 BAD :

```

ifconfig eth0:bad 100.80.0.3 netmask 255.255.255.0 up

```

```

eth0:bad: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 100.80.0.3 netmask 255.255.255.0 broadcast 100.80.0.255
    ether 6a:e0:9e:81:f2:4d txqueuelen 1000 (Ethernet)

```

Curl sur intranet + eth0 : bad :

```

root@mi-target-dev:~# tcpdump -i eth0:bad port 80
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0:bad, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C16:05:51.738774 IP 100.80.0.4.41584 > 100.80.0.5.http: Flags [S], seq 3998279561, win 64240, options [mss 1460,sackOK,TS val 1000549432 ecr 0,nop,wscale 7], length 0
16:05:51.738809 IP 100.80.0.4.41584 > 100.80.0.5.http: Flags [S], seq 3998279561, win 64240, options [mss 1460,sackOK,TS val 1000549432 ecr 0,nop,wscale 7], length 0
16:05:51.738873 IP 100.80.0.4.41584 > 100.80.0.5.http: Flags [.], ack 2340817323, win 502, options [nop,nop,TS val 1000549432 ecr 3857571751], length 0
16:05:51.738878 IP 100.80.0.4.41584 > 100.80.0.5.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 1000549432 ecr 3857571751], length 0
16:05:51.738924 IP 100.80.0.4.41584 > 100.80.0.5.http: Flags [P.], seq 0:74, ack 1, win 502, options [nop,nop,TS val 1000549432 ecr 3857571751], length 74: HTTP: GET / HTTP/1.1
16:05:51.738930 IP 100.80.0.4.41584 > 100.80.0.5.http: Flags [P.], seq 0:74, ack 1, win 502, options [nop,nop,TS val 1000549432 ecr 3857571751], length 74: HTTP:

```


Welcome to Firefox x start [Target] x +

https://www.target.milxc/doku.php

Firefox automatically sends some data to Mozilla so that we can improve your experience. Choose What I Share

Target

Log In

Search

Recent Changes Media Manager Sitemap

Trace: • start

start

Hi there ! We're a great cyber-company devoted to developping bleeding-edge cyber-security solutions. Our world-renowned cyber-experts leverage AI to the next level, so we can protect you from unknown and futuristic threats.

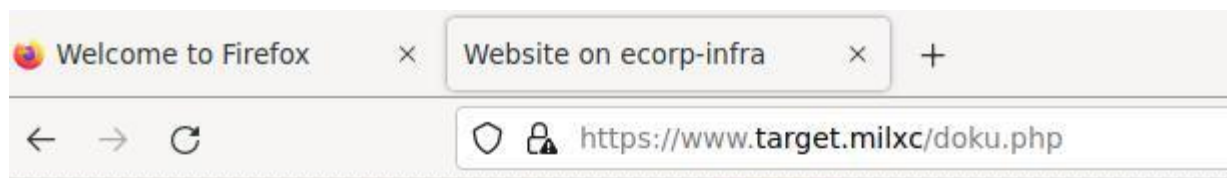
Feel free to get in touch with our sales department (commercial@target.milxc) for a free demo !

start.txt · Last modified: 2022/05/20 11:56 (external edit)

DONATE PHP POWERED W3C HTML W3C CSS CERN DOKUMENT

```
$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    100.81.0.2
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      100.81.0.2
ns.ecorp.milxc. IN AAAA    2001:db8:81::2
```

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

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