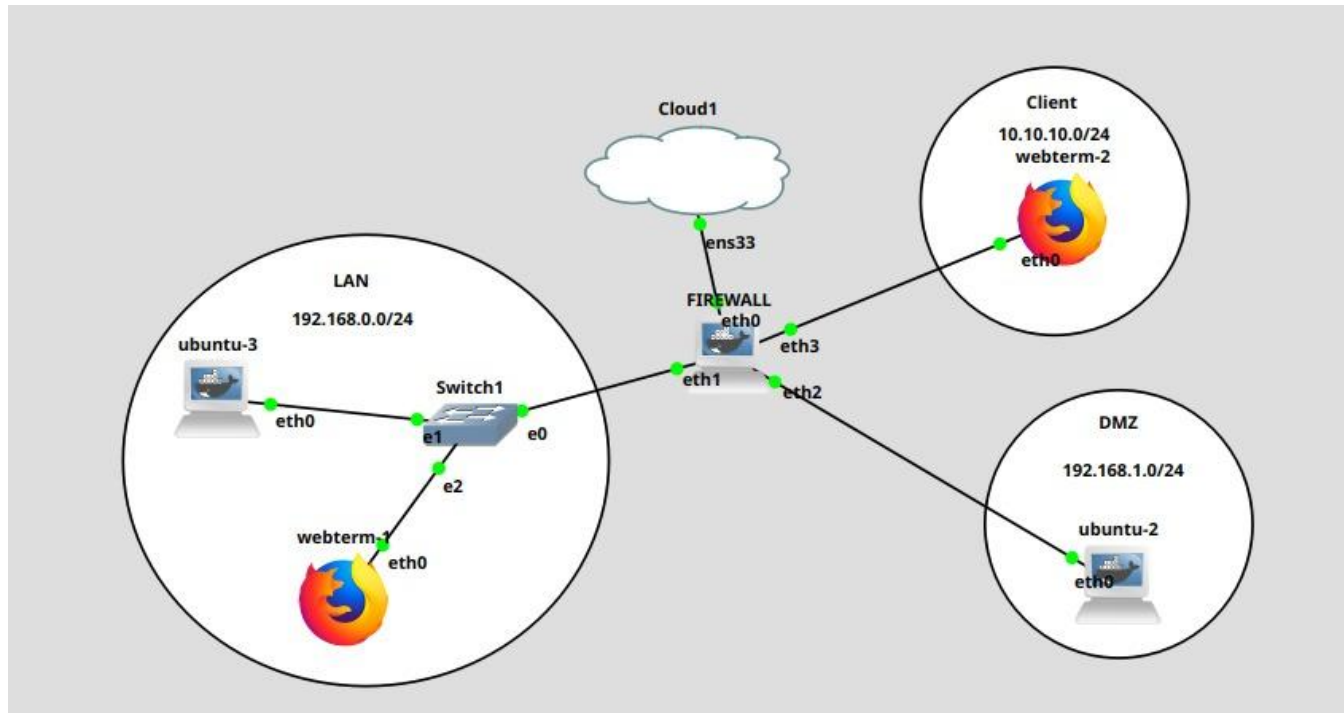


Rapport de Projet : Mise en place d'une DMZ sécurisée avec nftable

Étape 1 : Configuration des interfaces réseau

❖ Architecture :



Dans cette architecture, le **pare-feu Linux** joue un rôle central pour sécuriser les échanges entre trois réseaux distincts : le réseau Client, la DMZ, et le LAN. Il dispose de **l'interface du cloud eth0** et de **trois interfaces réseau physiques** qui correspondent à chacune de ces zones :

Interface	Réseau connecté	Plage IP	Rôle principal
eth3	Réseau Client	10.10.10.0/24	Accès des utilisateurs finaux
eth2	Réseau DMZ	192.168.1.0/24	Hébergement des services web
eth1	Réseau LAN	192.168.0.0/24	Ressources internes sensibles

❖ Configuration des interfaces sous Linux

- réseau du pare-feu Linux (/etc/network/interfaces)

```
# source /etc/network/interfaces.d/*

# eth0 : Accès Internet
auto eth0
iface eth0 inet dhcp

# eth1 : LAN
auto eth1
iface eth1 inet static
    address 192.168.0.1
    netmask 255.255.255.0

# eth2 : DMZ
auto eth2
iface eth2 inet static
    address 192.168.1.1
    netmask 255.255.255.0

# eth3 : client-internet
auto eth3
iface eth3 inet static
    address 10.10.10.1
    netmask 255.255.255.0
```

Le fichier de configuration `/etc/network/interfaces` permet de définir les paramètres IP des interfaces réseau du pare-feu. Voici la configuration utilisée :

- **eth0** : Cette interface est connectée au **Cloud1** ou à une passerelle Internet, et elle est configurée pour obtenir automatiquement une adresse IP via DHCP. Cela permet au pare-feu d'avoir un accès Internet sans avoir à fixer manuellement l'adresse IP
- **eth1** : L'interface eth1 est dédiée au **réseau LAN** (192.168.0.0/24), qui héberge des ressources internes sensibles. Le pare-feu prend l'adresse **192.168.0.1**, servant de passerelle pour les machines du LAN.
- **eth2** : L'interface eth2 est connectée à la **DMZ** (zone démilitarisée), qui contient des services accessibles depuis l'extérieur, comme le serveur web ubuntu-2. Le pare-feu est configuré avec l'adresse **192.168.1.1**, qui servira de passerelle au serveur.
- **eth3** : Cette interface eth3 est utilisée pour le **réseau client-internet**, représenté par webterm-2. Ce réseau est utilisé par les clients finaux qui peuvent naviguer vers Internet ou accéder aux services hébergés dans la DMZ. Le pare-feu est configuré en **10.10.10.1**.

```
#
# This is a sample network config, please uncomment lines to configure the network
#

# Uncomment this line to load custom interface files
# source /etc/network/interfaces.d/*

# Static config for eth0
auto eth0
iface eth0 inet static
    address 192.168.1.2
    netmask 255.255.255.0
    gateway 192.168.1.1
    up echo nameserver 192.168.0.1 > /etc/resolv.conf
```

Cette configuration concerne la machine ubuntu2 dans la **DMZ** . Elle utilise une IP statique (**192.168.1.2**) avec comme **passerelle** le pare-feu (**192.168.1.1**). La commande `up echo nameserver 192.168.0.1 > /etc/resolv.conf` définit manuellement le DNS, pointant vers une machine du **LAN** (peut-être un serveur DNS interne).

```
#
# This is a sample network config, please uncomment lines to configure the network
#

# Uncomment this line to load custom interface files
# source /etc/network/interfaces.d/*

# Static config for eth0
auto eth0
iface eth0 inet static
    address 192.168.0.3
    netmask 255.255.255.0
    gateway 192.168.0.1
    up echo nameserver 192.168.0.1 > /etc/resolv.conf
```

Cette configuration est celle de la machine ubuntu-3 dans le **réseau LAN**. Elle reçoit une **adresse IP statique 192.168.0.3**, avec pour **passerelle** le pare-feu (**192.168.0.1**)

```
#
# This is a sample network config, please uncomment lines to configure the network
#

# Uncomment this line to load custom interface files
# source /etc/network/interfaces.d/*

# Static config for eth0
auto eth0
iface eth0 inet static
    address 10.10.10.2
    netmask 255.255.255.0
    gateway 10.10.10.1
    up echo nameserver 10.10.10.1 > /etc/resolv.conf
```

Cette configuration correspond à la machine située dans le **réseau client-internet** webterm-2 . Elle utilise une **adresse IP statique 10.10.10.2**, avec comme **passerelle** le pare-feu (**10.10.10.1**). Le DNS est défini manuellement avec `up echo nameserver 10.10.10.1 > /etc/resolv.conf`, ce qui permet de résoudre les noms via le pare-feu .

❖ Étape 2 : Configuration de nftables :

Dans cette étape, nous mettons en place les **règles de pare-feu avec nftables** afin d'assurer la sécurité et le contrôle du trafic réseau entre les différentes zones :

Internet, DMZ, LAN et client.

L'objectif est de :

- **Limiter l'accès** aux ressources sensibles,
- **Autoriser uniquement le trafic légitime** (ex. HTTP/HTTPS vers le serveur web en DMZ),
- **Empêcher les connexions non sollicitées** (ex. Internet vers LAN),
- **NATer le trafic sortant** pour que les hôtes internes accèdent à Internet via le pare-feu,
- Et **journaliser** les tentatives d'accès bloquées pour faciliter l'audit de sécurité.

Voici les règles que notre pare-feu doit respecter.

Source	Destination	Port/service	Autoriser
Internet	DMZ (WEB)	80/443	<u>oui</u>
Internet	LAN	Tous	<u>non</u>
LAN	Internet	Tous	<u>oui</u>
LAN	DMZ	Tous	<u>oui</u>
DMZ	LAN	Tous	non

Installation de nftables :

```
/ # apt install nftables -y
```

Configuration de nftables – Règles de sécurité et NAT

Lien de téléchargement du fichier de configuration :

[https://drive.google.com/file/d/1px_U4m9ryjaPR1pWL - U3S1diAk0s71F/view?usp=drive_link](https://drive.google.com/file/d/1px_U4m9ryjaPR1pWL-U3S1diAk0s71F/view?usp=drive_link)

```
# Supprimer toutes les règles précédentes pour éviter les conflits
flush ruleset

#####
# Table de filtrage des paquets (filter)
#####
table inet filter {

    # === Chaîne INPUT ===
    # Gère les paquets destinés au pare-feu lui-même
    chain input {
        type filter hook input priority 0;
        policy drop;

        # Autoriser le trafic local (loopback)
        iif "lo" accept

        # Autoriser les connexions déjà établies ou connexes
        ct state established,related accept

        # Autoriser ICMP (ping) en IPv4 et IPv6
        ip protocol icmp accept
        ip6 nexthdr icmpv6 accept

        # Autoriser SSH vers le pare-feu depuis le LAN (interface eth1)
        iif "eth1" tcp dport 22 accept

        # Autoriser SSH vers le pare-feu depuis le LAN (interface eth1)
        iif "eth1" tcp dport 22 accept

        # Autorise le LAN au serveur DNS
        iif "eth1" udp dport 53 accept
        iif "eth1" tcp dport 53 accept
        iif "eth2" udp dport 53 accept
        iif "eth2" tcp dport 53 accept

        # Journaliser et bloquer les paquets venant d'Internet vers le LAN
        iif "eth0" oif "eth1" log prefix "DROP-INTERNET-LAN: " flags all counter drop
    }

    # === Chaîne FORWARD ===
    # Gère les paquets qui traversent le pare-feu (transit entre interfaces)
    chain forward {
        type filter hook forward priority 0;
        policy drop;

        # Autoriser les connexions déjà établies ou connexes
        ct state established,related accept

        # Autoriser le trafic du LAN vers Internet
        iif "eth1" oif "eth0" accept

        # Autoriser le trafic du LAN vers la DMZ
        iif "eth1" oif "eth2" accept
    }
}
```



```
# DMZ vers Internet : autorisé
iif "eth2" oif "eth0" accept

# Autoriser trafic de eth3 (LAN 10.10.10.0/24) vers DMZ
iif "eth3" oif "eth2" accept

# Interdire le trafic de la DMZ vers le LAN (politique par défaut : drop)

# Autoriser l'accès HTTP/HTTPS depuis Internet vers les serveurs web en DMZ
iif "eth0" oif "eth2" tcp dport { 80, 443 } accept

# Journaliser tout le reste venant d'Internet non autorisé explicitement
iif "eth0" log prefix "DROP-FWD-INTERNET: " flags all counter drop
}

# === Chaîne OUTPUT ===
# Gère les paquets émis par le pare-feu lui-même
chain output {
    type filter hook output priority 0;
    policy drop;

    # Autoriser les connexions sortantes du pare-feu
    oif "eth0" accept

    # Autoriser les connexions établies/connexes
    ct state established,related accept
}

#####
# Table de traduction d'adresses (NAT)
#####
table ip nat {

    # === Chaîne PREROUTING ===
    # S'applique aux paquets entrants AVANT le routage
    chain prerouting {
        type nat hook prerouting priority -100;
        iif "eth0" tcp dport 80 dnat to 192.168.1.2:80
        iif "eth0" tcp dport 443 dnat to 192.168.1.2:443
    }

    # === Chaîne POSTROUTING ===
    # S'applique aux paquets sortants APRÈS le routage
    chain postrouting {
        type nat hook postrouting priority 100;

        # Activer le masquering pour permettre aux hôtes du LAN et de la DMZ d'accéder à Internet via le pare-feu

        # NAT pour les hôtes du LAN
        oifname "eth0" ip saddr 192.168.0.0/24 masquerade

        # NAT pour les hôtes de la DMZ
        oifname "eth0" ip saddr 192.168.1.0/24 masquerade
    }
}
```

Dans cette section, nous avons défini les règles principales de notre pare-feu à l'aide de nftables. Le fichier de configuration se compose de deux grandes parties :

- **Table filter (filtrage) :**
Elle contient les règles de sécurité pour contrôler le trafic :
 - Tout trafic non explicitement autorisé est bloqué par défaut (politique drop).
 - Le trafic local (loopback), les connexions déjà établies, le ping, le SSH depuis le LAN, et l'accès DNS sont autorisés.

- Les règles de la chaîne forward permettent au LAN et à la DMZ d'accéder à Internet, mais bloquent les flux non autorisés, notamment de la DMZ vers le LAN.
- Les accès web (HTTP/HTTPS) de l'extérieur vers la DMZ sont autorisés.
- Les paquets rejetés sont **journalisés** pour permettre une surveillance réseau.
- **Table nat (traduction d'adresse) :**
Elle permet l'accès à Internet depuis le LAN et la DMZ via le pare-feu.
 - La technique de **masquering** est utilisée pour masquer les adresses internes lors de la sortie vers Internet.

Ce fichier constitue la **base de la politique de sécurité** appliquée par notre pare-feu.

Application de la configuration :

```
/ # nft -f firewall_conf.nft
```

Cette commande permet de **charger et appliquer toutes les règles** définies dans le fichier firewall_conf.nft

❖ Étape 3 – Déploiement d'un service web dans la DMZ

Cette étape a pour objectif de **mettre en place un serveur web sécurisé** dans la DMZ (Zone Démilitarisée) du réseau. L'idée est d'héberger un service (site ou application web) **accessible uniquement depuis Internet**, tout en empêchant tout accès direct depuis ou vers le réseau interne (LAN), conformément aux bonnes pratiques de sécurité.

Objectifs :

- Installer un serveur **Apache** ou **Nginx** sur la machine située dans la DMZ (adresse IP 192.168.1.2, par exemple).
- S'assurer que le service est **accessible uniquement depuis Internet**, via les **ports 80 (HTTP)** et **443 (HTTPS)**.
- Vérifier que les requêtes provenant d'autres zones non autorisées sont bloquées.

Installation et configuration du serveur web dans la DMZ :

Après avoir fait une mise à jour :

```
/ # apt update
```

On a installé apache2

```
/ # apt install apache2 -y
```

Ensuite, une **page web HTML simple** a été créée dans le fichier /var/www/html/index.html à l'aide d'un script bash

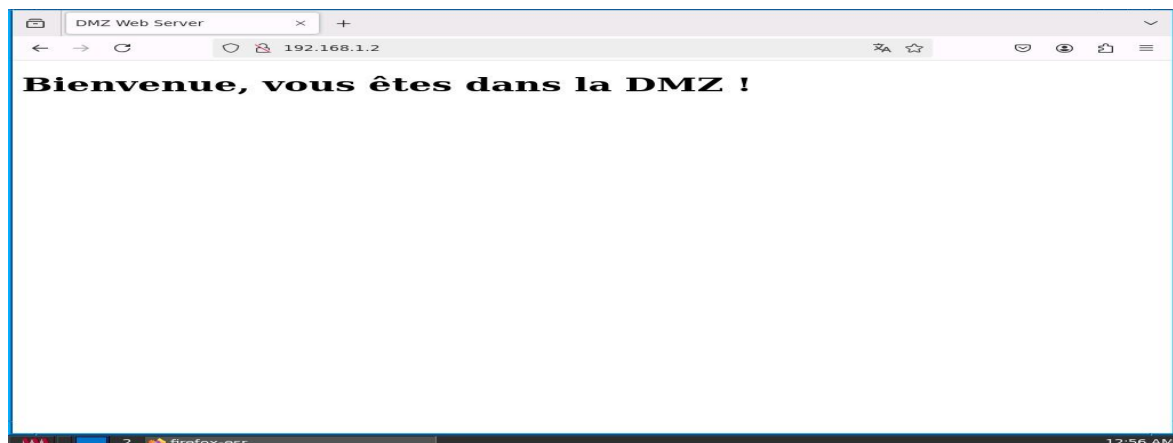
```
/ # bash -c 'cat > /var/www/html/index.html <<EOF
> <!DOCTYPE html>
> <html lang="fr">
> <head>
>     <meta charset="UTF-8">
>     <title>DMZ Web Server</title>
> </head>
> <body>
>     <h1>Bienvenue, vous êtes dans la DMZ !</h1>
> </body>
> </html>
> EOF'
/ #
```

Cette page contient un message de bienvenue indiquant que l'utilisateur accède au serveur situé dans la DMZ.

```
# Autoriser l'accès HTTP/HTTPS depuis Internet vers les serveurs web en DMZ
iif "eth0" oif "eth2" tcp dport { 80, 443 } accept
```

Ce service est désormais accessible depuis **Internet uniquement**, via le **port 80 et 443**, conformément aux règles de **nftables** définies dans l'étape précédente.

SUR LE CLIENT :

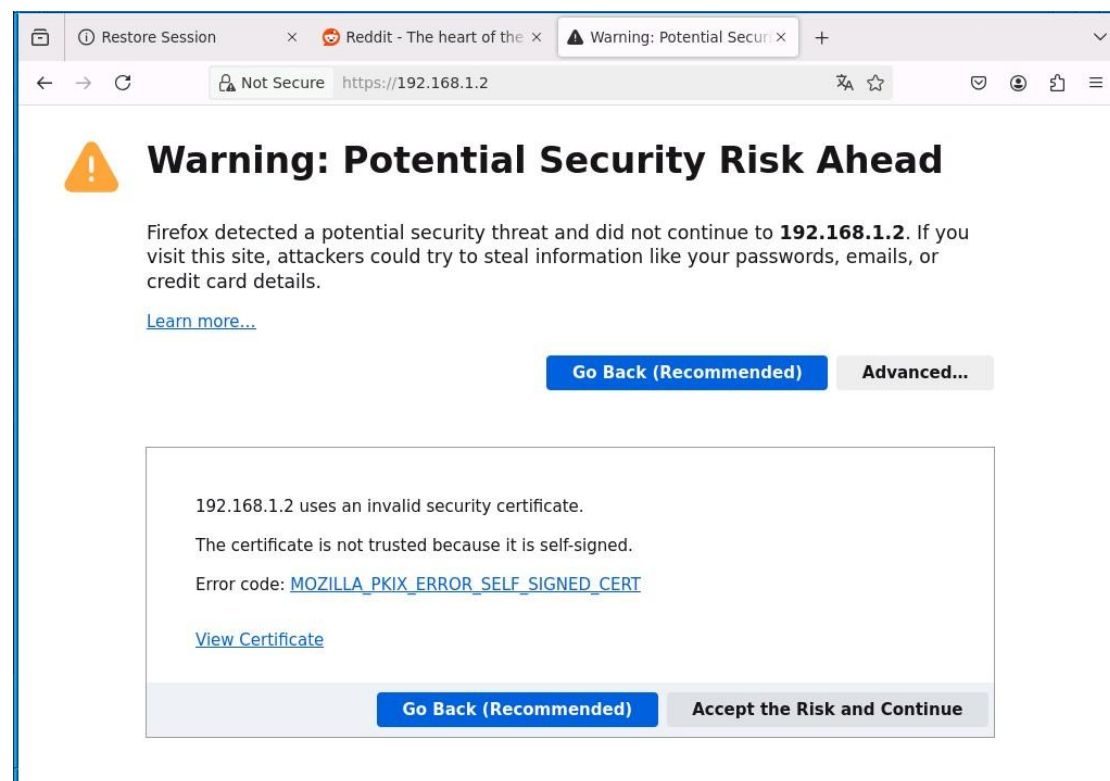



```
/ # a2enmod ssl
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create s
elf-signed certificates.
To activate the new configuration, you need to run:
    service apache2 restart
```

La commande `a2enmod ssl` est utilisée sur les systèmes basés sur **Apache2** (comme Debian, Ubuntu, etc.) pour **activer le module SSL** dans Apache, afin de permettre les connexions HTTPS.

```
/ # a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
    service apache2 reload
/ # service apache2 reload
* Reloading Apache httpd web server apache2
*
```

`a2ensite default-ssl` active le site Apache configuré pour HTTPS (port 443) en liant le fichier `default-ssl.conf` dans les sites activés. Cela permet à Apache de servir des pages via une connexion sécurisée SSL/TLS.





Nous avons activé le succès la connexion via HTTPS

Ce site est désormais disponible via l'adresse de la machine server dmz

❖ Étape 4 : Tests et validations :

Cette étape permet de vérifier que la configuration du pare-feu, du routage et des services respecte bien les règles de sécurité définies dans l'architecture. Les tests suivants ont été réalisés pour valider le fonctionnement du réseau et l'isolation des zones :

1. Accès au serveur web depuis internet :

No.	Time	Source	Destination	Protocol	Length	Info
16	0.000000	10.10.10.2	192.168.1.2	TCP	74	57372 → 80 [SYN] Seq=0 Win=64256 Len=0 MSS=1460 SACK_PERM TSval=2501887261 TSecr=0 WS=128
20	0.000091	192.168.1.2	10.10.10.2	TCP	74	80 → 57372 [ACK] Seq=0 Ack=1 Win=65536 Len=0 MSS=1460 SACK_PERM TSval=2410543969 TSecr=2501887261 WS=128
30	0.001338	10.10.10.2	192.168.1.2	TCP	66	57372 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2501887262 TSecr=2410543969
40	0.002113	10.10.10.2	192.168.1.2	HTTP	487	GET / HTTP/1.1
50	0.003446	192.168.1.2	10.10.10.2	TCP	66	80 → 57372 [ACK] Seq=1 Ack=422 Win=64768 Len=0 TSval=2410543973 TSecr=2501887262
60	0.003935	192.168.1.2	10.10.10.2	HTTP	565	HTTP/1.1 200 OK (text/html)
70	0.006341	10.10.10.2	192.168.1.2	TCP	66	57372 → 80 [ACK] Seq=422 Ack=500 Win=84128 Len=0 TSval=2501887266 TSecr=2410543973
85	0.006504	10.10.10.2	192.168.1.2	TCP	66	57372 → 80 [FIN, ACK] Seq=422 Ack=500 Win=84128 Len=0 TSval=2501887267 TSecr=2410543973
95	0.006923	192.168.1.2	10.10.10.2	TCP	66	80 → 57372 [FIN, ACK] Seq=500 Ack=423 Win=64768 Len=0 TSval=2410548975 TSecr=2501887267
105	0.006349	10.10.10.2	192.168.1.2	TCP	66	57372 → 80 [ACK] Seq=423 Ack=501 Win=84128 Len=0 TSval=2501887267 TSecr=2410548975
115	1.139357	02:42:0b:60:9c:02	02:42:0b:60:9c:02	ARP	42	Who has 192.168.1.2? Tell 192.168.1.1
125	1.139711	02:42:0b:60:9c:02	02:42:0b:60:9c:02	ARP	42	192.168.1.2 is at 02:42:0b:60:9c:02

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.10.2	10.10.10.1	DNS	88	Standard query 0x2934 A contile.services.mozilla.com
2	0.000076	10.10.10.2	10.10.10.1	DNS	88	Standard query 0xea32 AAAA contile.services.mozilla.com
3	0.000273	10.10.10.2	10.10.10.1	DNS	79	Standard query 0x1117 A spocs.getpocket.com
4	0.000300	10.10.10.2	10.10.10.1	DNS	79	Standard query 0xf129 AAAA spocs.getpocket.com
5	1.264598	10.10.10.2	192.168.1.2	TCP	74	52076 → 80 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM TSval=2501787899 TSecr=0 WS=128
6	1.264783	192.168.1.2	10.10.10.2	TCP	74	80 → 52076 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=1460 SACK_PERM TSval=2501787899 TSecr=2501787899 WS=128
7	1.264869	10.10.10.2	192.168.1.2	TCP	66	52076 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2501787899 TSecr=2410444607
8	1.265021	10.10.10.2	192.168.1.2	HTTP	487	GET / HTTP/1.1
9	1.265145	192.168.1.2	10.10.10.2	HTTP	66	80 → 52076 [ACK] Seq=1 Ack=422 Win=64768 Len=0 TSval=2501787899 TSecr=2501787899
10	1.265882	192.168.1.2	10.10.10.2	HTTP	565	HTTP/1.1 200 OK (text/html)
11	1.266000	10.10.10.2	192.168.1.2	TCP	66	52076 → 80 [ACK] Seq=422 Ack=500 Win=64128 Len=0 TSval=2501787900 TSecr=2410444608
12	5.004918	10.10.10.2	10.10.10.1	DNS	88	Standard query 0x2934 A contile.services.mozilla.com
13	5.004989	10.10.10.2	10.10.10.1	DNS	88	Standard query 0xea32 AAAA contile.services.mozilla.com
14	5.007089	10.10.10.2	10.10.10.1	DNS	79	Standard query 0x1117 A spocs.getpocket.com
15	5.007977	10.10.10.2	10.10.10.1	DNS	79	Standard query 0xf129 AAAA spocs.getpocket.com
16	6.267122	10.10.10.2	192.168.1.2	TCP	66	52076 → 80 [FIN, ACK] Seq=422 Ack=500 Win=64128 Len=0 TSval=2501792901 TSecr=2410444608
17	6.268085	192.168.1.2	10.10.10.2	TCP	66	80 → 52076 [FIN, ACK] Seq=500 Ack=423 Win=64768 Len=0 TSval=2501792901 TSecr=2501792901
18	6.269817	10.10.10.2	192.168.1.2	TCP	66	52076 → 80 [ACK] Seq=423 Ack=501 Win=64128 Len=0 TSval=2501792904 TSecr=2410444610
19	6.436672	02:42:0b:0b:9c:03	02:42:d1:ea:da:00	ARP	42	Who has 10.10.10.2? Tell 10.10.10.1
20	6.436754	02:42:d1:ea:da:00	02:42:0b:0b:9c:03	ARP	42	10.10.10.2 is at 02:42:d1:ea:da:00
21	10.009982	10.10.10.2	10.10.10.1	DNS	88	Standard query 0xb72f A contile.services.mozilla.com
22	10.010012	10.10.10.2	10.10.10.1	DNS	88	Standard query 0x6531 AAAA contile.services.mozilla.com
23	10.012880	10.10.10.2	10.10.10.1	DNS	79	Standard query 0x442b A spocs.getpocket.com
24	10.012918	10.10.10.2	10.10.10.1	DNS	79	Standard query 0xf02a AAAA spocs.getpocket.com
25	15.013033	10.10.10.2	10.10.10.1	DNS	88	Standard query 0xb72f A contile.services.mozilla.com
26	15.015127	10.10.10.2	10.10.10.1	DNS	88	Standard query 0x6531 AAAA contile.services.mozilla.com
27	15.017535	10.10.10.2	10.10.10.1	DNS	79	Standard query 0x442b A spocs.getpocket.com
28	15.017606	10.10.10.2	10.10.10.1	DNS	79	Standard query 0xf02a AAAA spocs.getpocket.com

2. Aucune réponse de LAN depuis internet :

```
/ # ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2): 56 data bytes
^C
--- 192.168.0.2 ping statistics ---
50 packets transmitted, 0 packets received, 100% packet loss
```

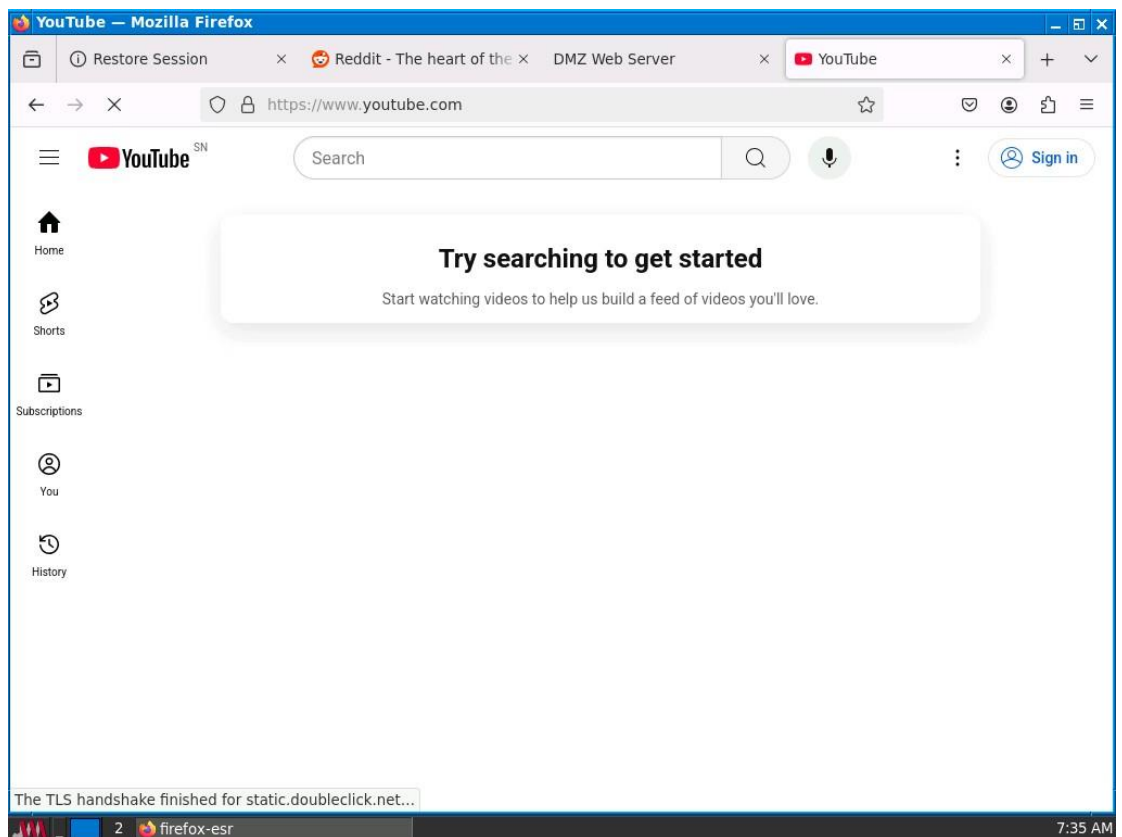
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.10.2	10.10.10.1	DNS	84	Standard query 0x81d2 A de
2	0.000200	10.10.10.2	10.10.10.1	DNS	84	Standard query 0xd7d0 AAA
3	0.505068	02:42:d1:ea:da:00	02:42:0b:6b:9c:03	ARP	42	Who has 10.10.10.1? Tell 1
4	0.505305	02:42:0b:6b:9c:03	02:42:d1:ea:da:00	ARP	42	10.10.10.1 is at 02:42:0b:
5	2.972144	10.10.10.2	192.168.0.2	ICMP	98	Echo (ping) request id=0x
6	3.972504	10.10.10.2	192.168.0.2	ICMP	98	Echo (ping) request id=0x
7	4.972605	10.10.10.2	192.168.0.2	ICMP	98	Echo (ping) request id=0x
8	5.005728	10.10.10.2	10.10.10.1	DNS	84	Standard query 0x066a A de
9	5.005855	10.10.10.2	10.10.10.1	DNS	84	Standard query 0xd36e AAA
10	5.972777	10.10.10.2	192.168.0.2	ICMP	98	Echo (ping) request id=0x
11	6.973095	10.10.10.2	192.168.0.2	ICMP	98	Echo (ping) request id=0x

3. Accès a internet depuis le LAN et le serveur DMZ :

```
/ # ping google.com
PING google.com (142.250.201.78): 56 data bytes
64 bytes from 142.250.201.78: seq=0 ttl=113 time=56.403 ms
64 bytes from 142.250.201.78: seq=1 ttl=113 time=54.139 ms
64 bytes from 142.250.201.78: seq=2 ttl=113 time=90.846 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 54.139/67.129/90.846 ms
/ #
```

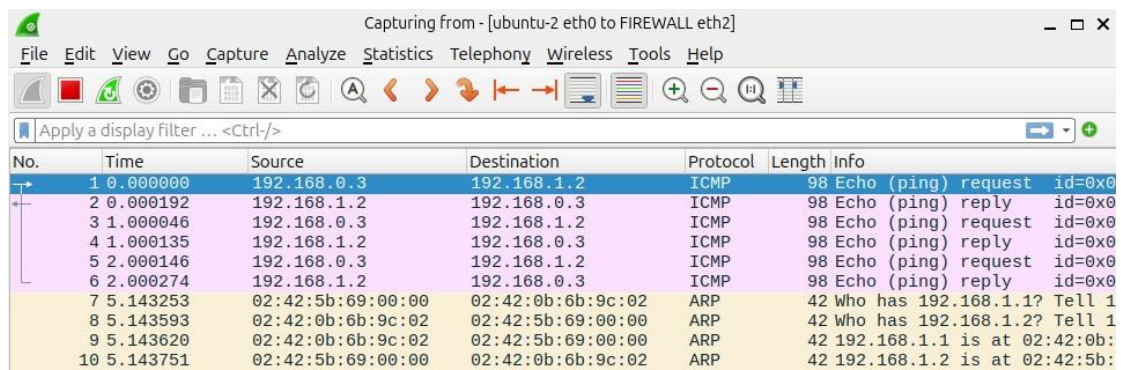
Capturing from - [ubuntu-2 eth0 to FIREWALL eth2]

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.2	192.168.1.1	DNS	70	Standard query 0x7e00 A goo
2	0.000105	192.168.1.2	192.168.1.1	DNS	70	Standard query 0x2e06 AAAA
3	0.000139	192.168.1.1	192.168.1.2	DNS	86	Standard query response 0x7
4	0.000386	192.168.1.1	192.168.1.2	DNS	98	Standard query response 0x2
5	0.009099	192.168.1.2	142.250.201.78	ICMP	98	Echo (ping) request id=0x1
6	0.065373	142.250.201.78	192.168.1.2	ICMP	98	Echo (ping) reply id=0x1
7	1.009407	192.168.1.2	142.250.201.78	ICMP	98	Echo (ping) request id=0x1
8	1.063334	142.250.201.78	192.168.1.2	ICMP	98	Echo (ping) reply id=0x1
9	2.009854	192.168.1.2	142.250.201.78	ICMP	98	Echo (ping) request id=0x1
10	2.100432	142.250.201.78	192.168.1.2	ICMP	98	Echo (ping) reply id=0x1
11	5.128433	02:42:5b:69:00:00	02:42:0b:6b:9c:02	ARP	42	Who has 192.168.1.1? Tell 1
12	5.128681	02:42:0b:6b:9c:02	02:42:5b:69:00:00	ARP	42	Who has 192.168.1.2? Tell 1
13	5.128743	02:42:0b:6b:9c:02	02:42:5b:69:00:00	ARP	42	192.168.1.1 is at 02:42:0b:
14	5.129310	02:42:5b:69:00:00	02:42:0b:6b:9c:02	ARP	42	192.168.1.2 is at 02:42:5b:



4. Communication LAN-DMZ :

```
/ # ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: seq=0 ttl=63 time=1.088 ms
64 bytes from 192.168.1.2: seq=1 ttl=63 time=0.739 ms
64 bytes from 192.168.1.2: seq=2 ttl=63 time=0.614 ms
^C
--- 192.168.1.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.614/0.813/1.088 ms
```

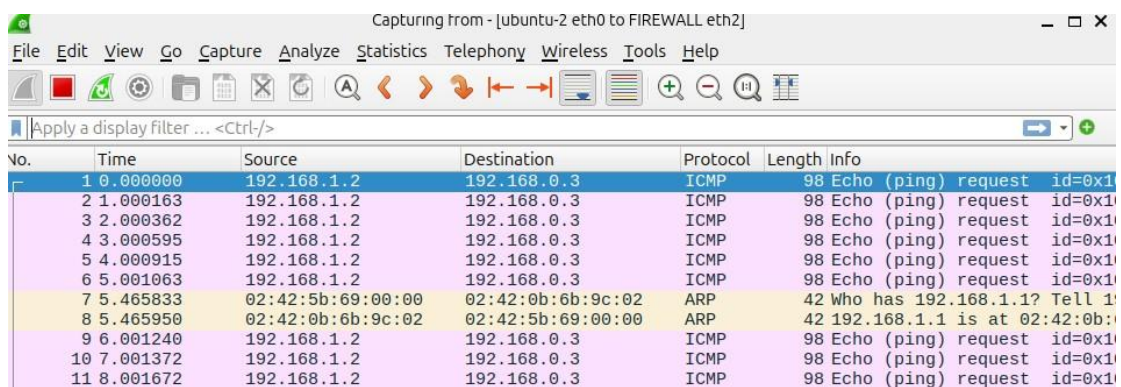



Capturing from - [ubuntu-2 eth0 to FIREWALL eth2]

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.3	192.168.1.2	ICMP	98	Echo (ping) request id=0x0
2	0.000192	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) reply id=0x0
3	1.000046	192.168.0.3	192.168.1.2	ICMP	98	Echo (ping) request id=0x0
4	1.000135	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) reply id=0x0
5	2.000146	192.168.0.3	192.168.1.2	ICMP	98	Echo (ping) request id=0x0
6	2.000274	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) reply id=0x0
7	5.143253	02:42:5b:69:00:00	02:42:0b:6b:9c:02	ARP	42	Who has 192.168.1.1? Tell 1
8	5.143593	02:42:0b:6b:9c:02	02:42:5b:69:00:00	ARP	42	Who has 192.168.1.2? Tell 1
9	5.143620	02:42:0b:6b:9c:02	02:42:5b:69:00:00	ARP	42	192.168.1.1 is at 02:42:0b:
10	5.143751	02:42:5b:69:00:00	02:42:0b:6b:9c:02	ARP	42	192.168.1.2 is at 02:42:5b:

5. pas de communication DMZ – LAN

```
/ # ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3): 56 data bytes
^C
--- 192.168.0.3 ping statistics ---
8 packets transmitted, 0 packets received, 100% packet loss
/ #
```



Capturing from - [ubuntu-2 eth0 to FIREWALL eth2]

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1
2	1.000163	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1
3	2.000362	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1
4	3.000595	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1
5	4.000915	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1
6	5.001063	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1
7	5.465833	02:42:5b:69:00:00	02:42:0b:6b:9c:02	ARP	42	Who has 192.168.1.1? Tell 1
8	5.465950	02:42:0b:6b:9c:02	02:42:5b:69:00:00	ARP	42	192.168.1.1 is at 02:42:0b:
9	6.001240	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1
10	7.001372	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1
11	8.001672	192.168.1.2	192.168.0.3	ICMP	98	Echo (ping) request id=0x1

Conclusion

Ce projet visait à mettre en place une architecture réseau sécurisée avec une DMZ, un LAN et un accès Internet, en utilisant un pare-feu Linux configuré avec nftables. La configuration des interfaces, des règles de filtrage, et du NAT a permis de contrôler strictement les communications entre les zones. Un serveur web a été déployé dans la DMZ et rendu accessible uniquement depuis Internet, tandis que les flux non autorisés (comme l'accès au LAN depuis Internet ou la DMZ) ont été bloqués. Les tests ont confirmé l'efficacité de la configuration. Ce travail a permis de mieux comprendre les concepts de sécurité réseau, de segmentation, et de filtrage, tout en offrant une base pour des infrastructures plus avancées.

