Projet Infrastructure Réseau / Sécurité

Alexandre SUCHOT Henry MAISONNEUVE Adrien VAUCARD

Table des matières

1. Introduction

1.1. Présentation

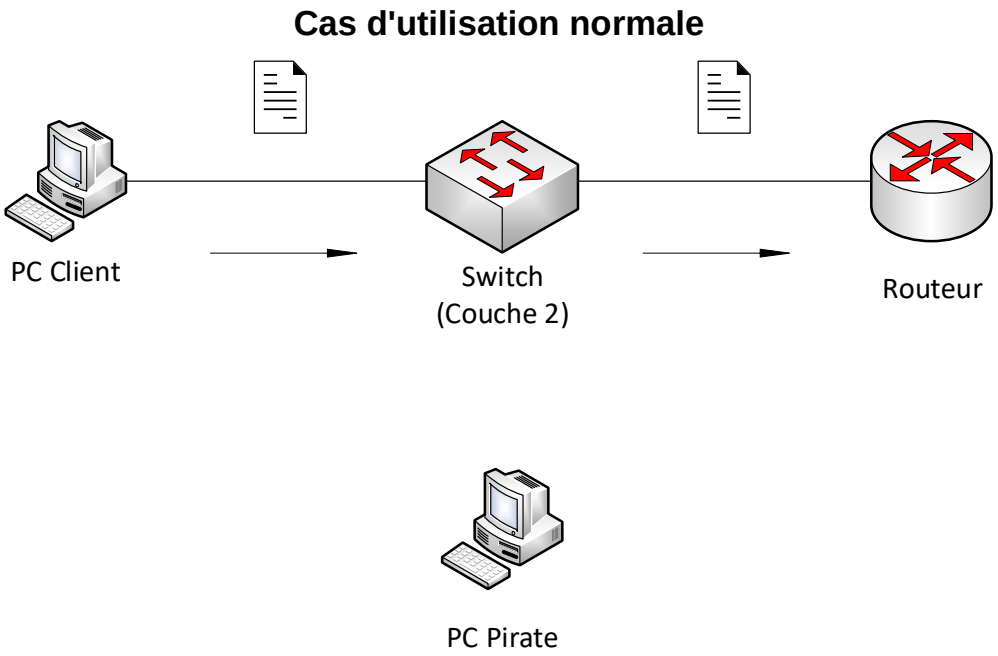
Dans le cadre de nos études, il nous a été donné d'effectuer un projet de Sécurité qui mêlait plusieurs aspects autour du réseau.

Ce projet consiste en l'établissement d'une attaque de type ***Man In The Middle*** dans un réseau local, dans le but d'intercepter des informations personnelles à travers le trafic.

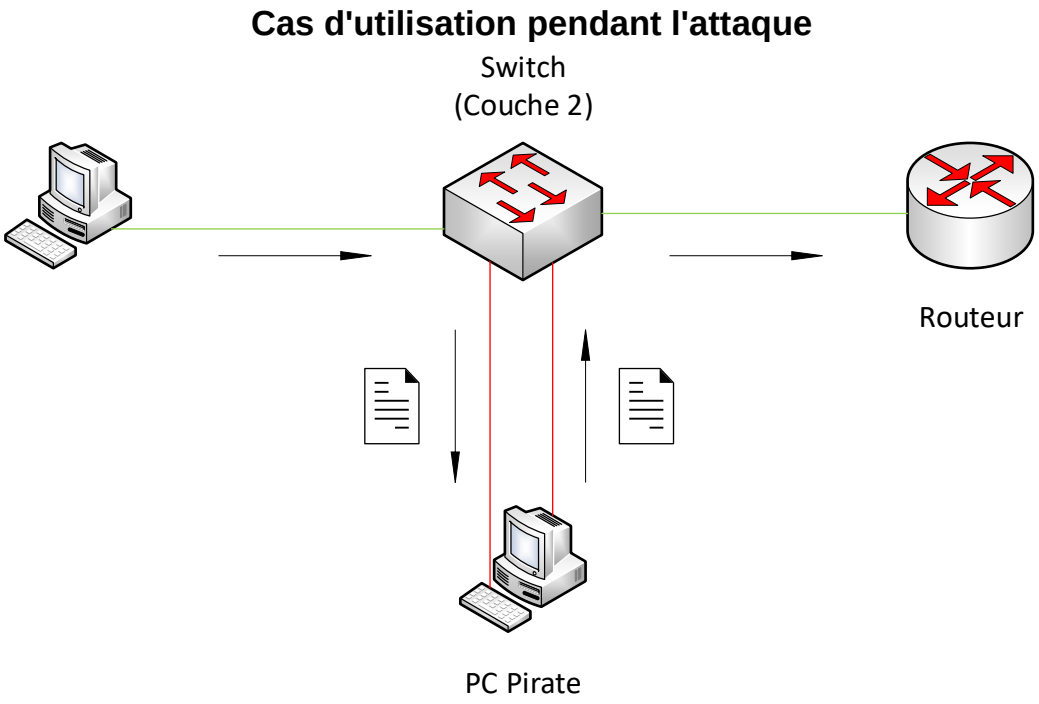
Celui-ci a pour but de nous sensibiliser sur la sécurité des réseaux Wifi publiques et sur les potentielles attaques que l'on pourrait subir dessus.

1.2. Man In The Middle

Une attaque ***Man In The Middle*** vise à intercepter le trafic d'une machine vers une autre (D'un client vers le routeur en général) en se faisant passer pour cette dernière.



Dans un cas d'utilisation courante, le client envoie ses paquets au routeur pour accéder à des sites, services en ligne, etc…



Lors d'une attaque ***Man In The Middle***, le pirate usurpe l'identité du routeur auprès du client et celle du client auprès du routeur. De ce fait, toutes les informations que s'échangent les 2 sont interceptées par la machine malicieuse.
Une fois le trafic dévié, le pirate peut très bien altérer les informations, dérober des mots de passes, etc…

Cela peut être fait, par exemple, avec la méthode d'***ARP Poisoning***.

1.3. ARP Poisoning

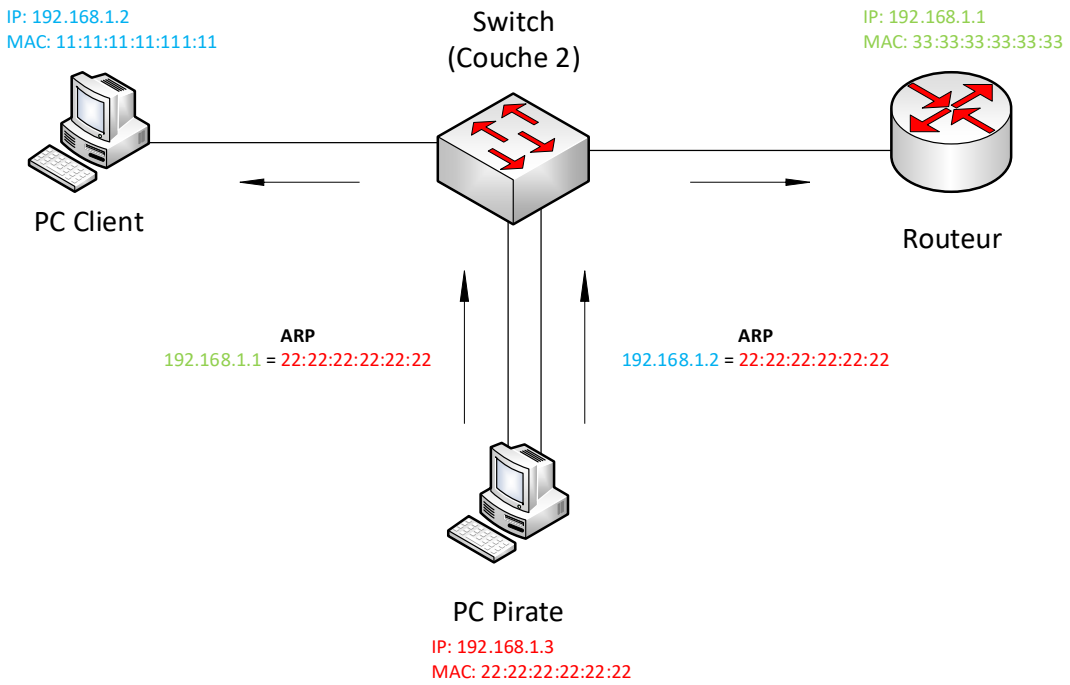
Il existe plusieurs manières de réaliser une attaque ***MITM***, mais dans ce cas-ci, il nous a été demandé de le faire via la méthode nommée ***ARP Poisoning***.

Le principe de celle-ci est d'inonder les tables ARP des postes clients pour "forcer" l'association de l'adresse IP du routeur avec l'adresse MAC du pirate (et forcer l'association de l'IP du client avec la MAC du pirate pour le routeur).

En effet, dans un réseau local avec un switch de couche 2, les machines communiquent par adresse MAC (Un Switch de couche 3 incorporerait une table IP et l'attaque serait inefficace).
Les machines présentes sur le réseau (***end-devices***) disposent donc d'une table ARP dynamique qui s'actualisent régulièrement grâce à des ***requêtes ARP*** effectuées en ***Broadcast***.

Celles-ci permettent de demander "*à quelle adresse MAC correspond telle adresse IP ?*".
Lors de la réception de la réponse, les adresses correspondantes sont stockées dans la table ARP.

En inondant la table ARP d'un poste, on le "force" à associer une adresse MAC avec une adresse IP et tant que les requêtes frauduleuses sont envoyées, les tables sont changées à notre avantage.



Comme il s'agit d'un switch de couche 2, les paquets sont envoyés, encapsulés dans une ***trame Ethernet*** qui se base sur les adresses MAC. Lorsque le paquet que souhaite envoyer le poste client se construira, l'adresse MAC de destination ne sera pas celle du routeur, mais bien celle de notre poste pirate.

1.4. Description Technique

En ce qui concerne le côté technique de ce que l'on utilise pour ce projet :

- 4 machines virtuelles sous **Debian 8** (Clients et pirate)
- 2 machines virtuelles sous **Debian 8** + **Apache** (Serveurs)
- 1 machine virtuelle sous **PfSense** (Routeur + Pare-feu)
- 1 Switch 16 ports TP-Link modèle TL-SG1016D

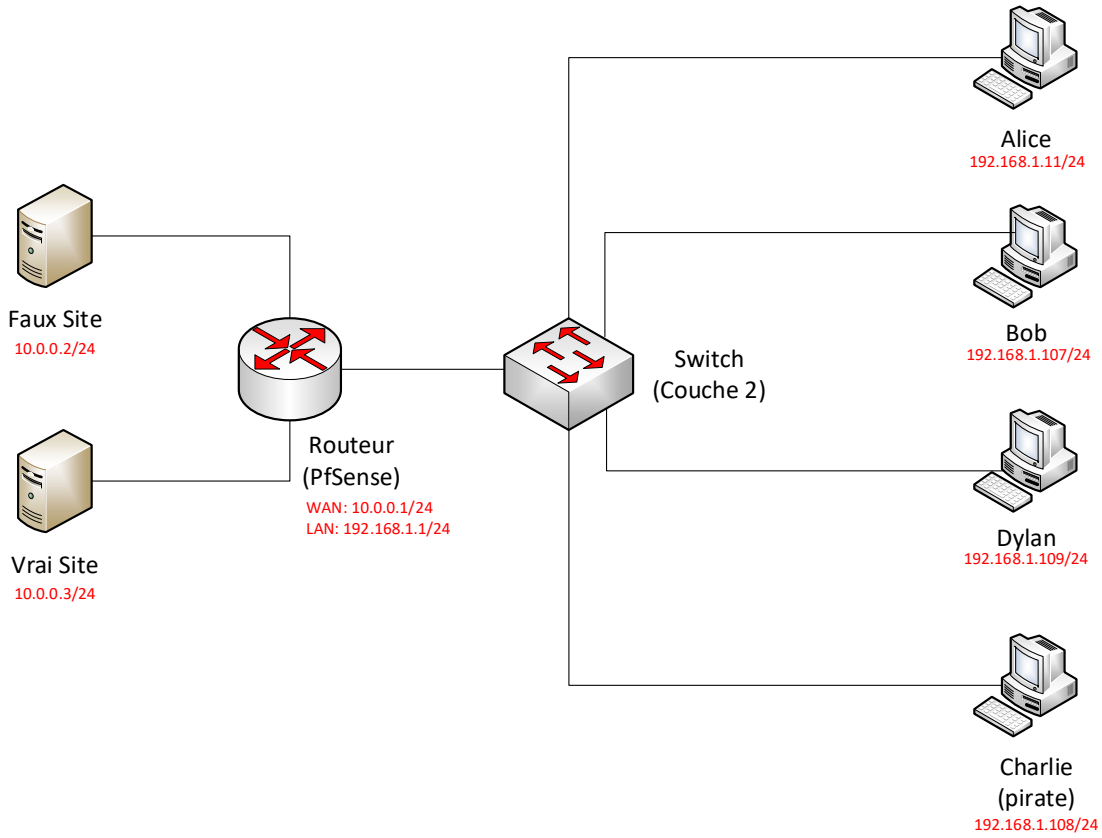
- **Python 3.4**
- **Scapy 2.4.2**
- **Phpmyadmin** (Base de données du vrai site)
- **Virtualbox 6.0** (Virtualisation)

2. Mise en œuvre

2.1. Architecture Réseau

Pour que nous puissions commencer à manipuler, il nous faut créer une architecture de réseau local. En effet, les actions que nous nous apprêtons à effectuer sont juridiquement prohibées sur des réseaux qui ne nous appartiennent pas (ici, le réseau d'Ynov).

Pour cela, nous avons réfléchi à la meilleure disposition possible et nous sommes mis d'accord dessus.



Pour des raisons pratiques et pour diminuer le matériel nécessaire, toutes les machines finales (***end-devices***) ainsi que le routeur/pare-feu ***PfSense***, seront sous forme de machine virtuelle (créée à l'aide de VirtualBox).

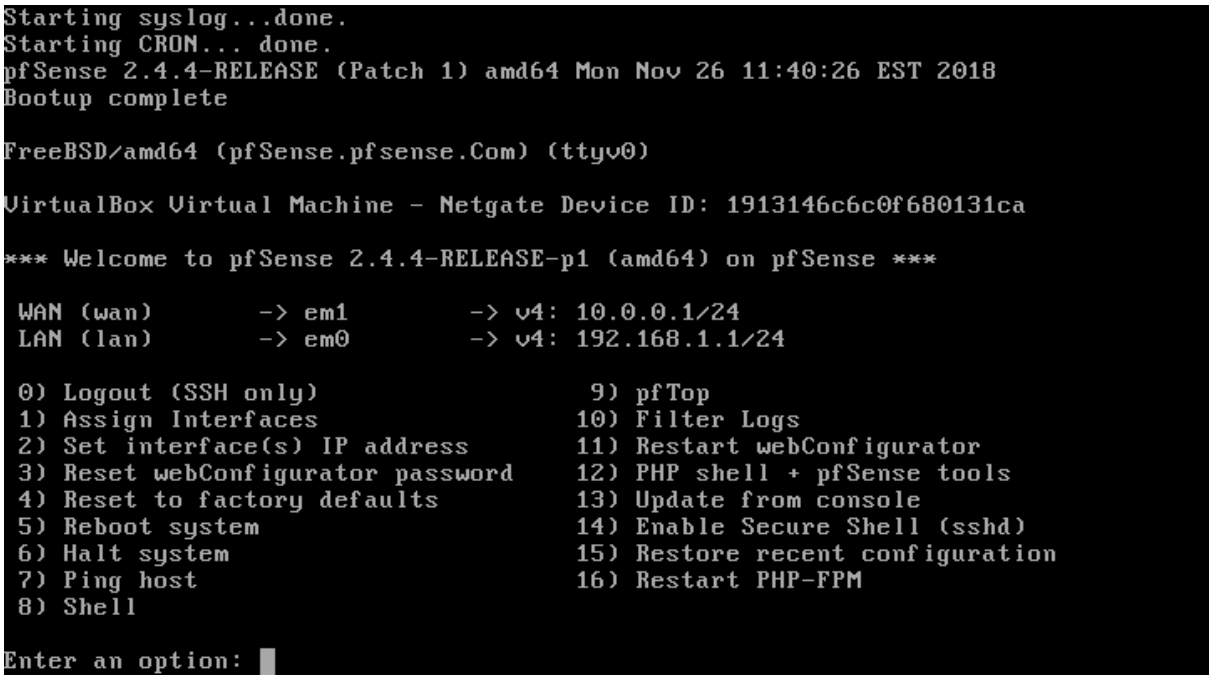
2.2. Installation

Premièrement, nous avons configuré ***PfSense***, qui joue le rôle de routeur et de pare-feu.

Nous avons attribué 2 interfaces réseau à cette machine :

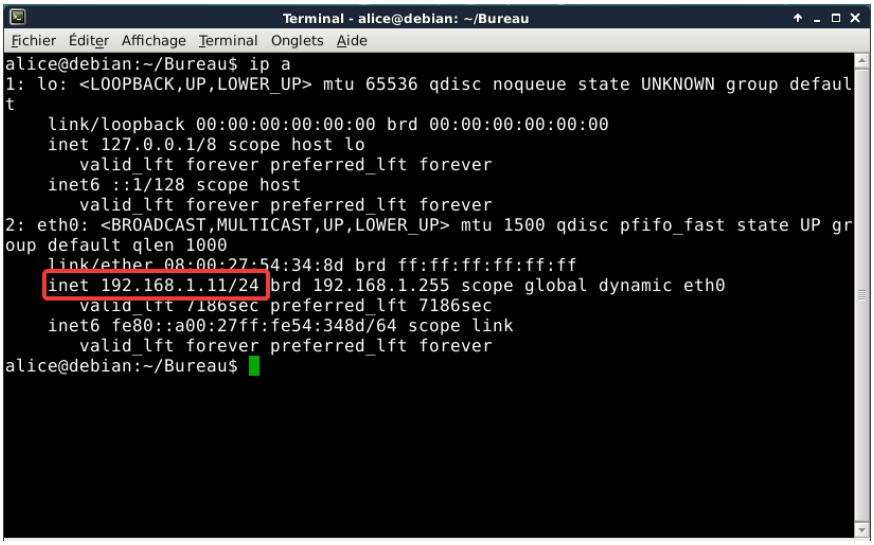
- **1 en Bridge**, reliée à un câble Ethernet pour accéder au Switch de couche 2 (Matériel physique). Cela correspond à la partie **LAN**.
- **1 en Réseau Interne**, pour simuler un réseau externe sur notre machine hôte. Cela correspond à la partie **WAN**.

Après avoir configuré et assigné les interfaces, nous avons défini notre ***PfSense*** en tant que ***serveur DHCP*** du côté du **LAN**, pour que les postes clients et pirates aient une IP attribuée automatiquement.



Ensuite, nous nous sommes occupés des postes clients et du poste pirate.

Ces machines ne disposent que d'une **interface en Bridge**, ce qui leur permet d'être reliées au Switch et de faire partie du réseau **LAN**.
Le routeur ***PfSense*** étant configuré en tant que **DHCP** du côté **LAN**, une adresse IP est automatiquement attribuée au client.

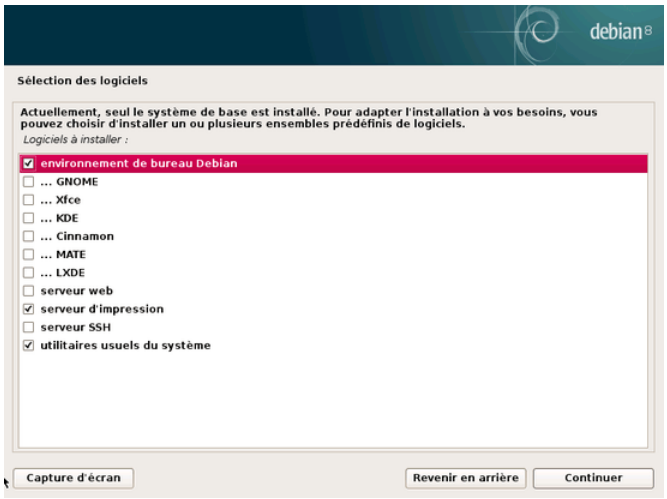


Par la suite, il nous a fallu configurer aussi les 2 serveurs (sous ***Apache***).

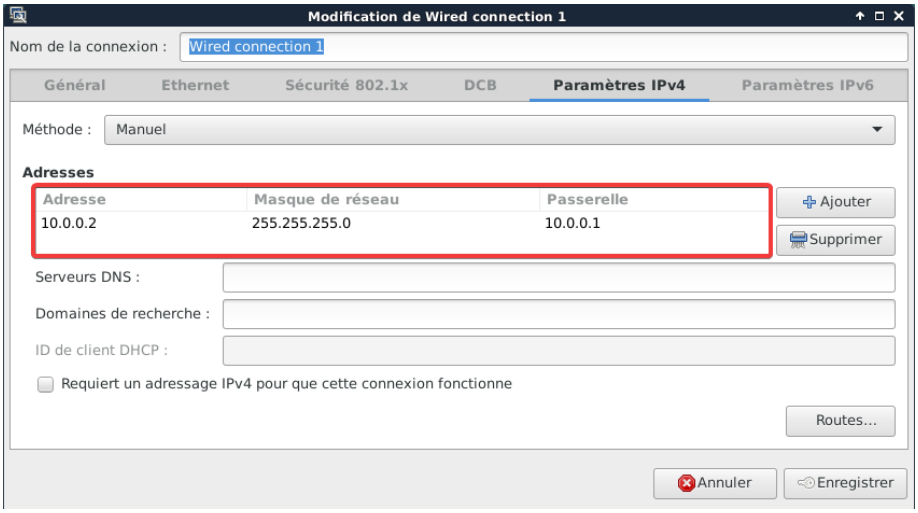
Pour ce qui est des interfaces réseau, nous avons :

- **1 interface en Réseau Interne**, qui simule un réseau externe à notre **LAN** et qui permettra d'accéder au pare-feu.
- **(Temporairement) 1 interface en NAT**, pour avoir un accès Internet et installer les paquets nécessaires au bon fonctionnement de notre machine (enlevée par la suite)

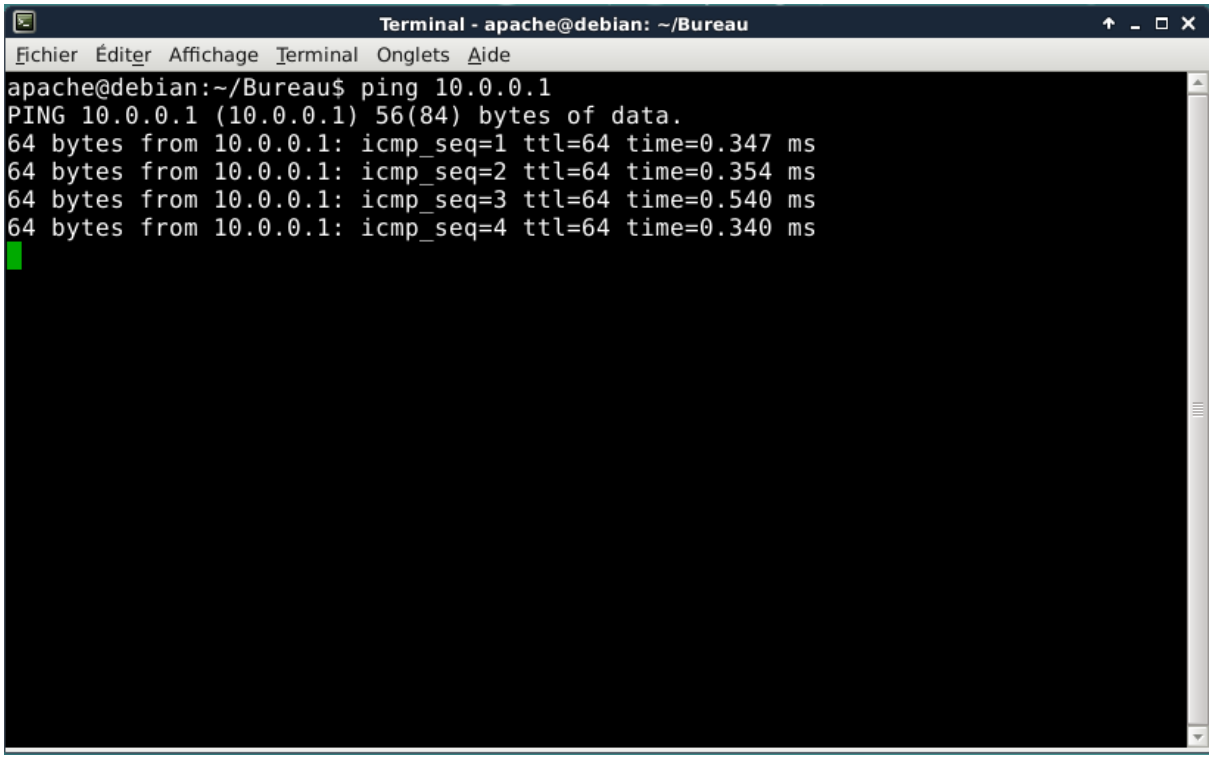
Plutôt que d'installer apache avec le gestionnaire de paquets ***Debian, APT***, nous avons préféré cocher la case "*Serveur Web*" à l'installation de notre Système d'exploitation.
De ce fait, notre serveur était opérationnel dès son lancement (Sans oublier d'installer les dépendances nécessaires au fonctionnement de notre site par la suite, telles que PHP et MySQL)



Puis, une fois apache installé, nous avons déterminé une adresse IP pour la machine.

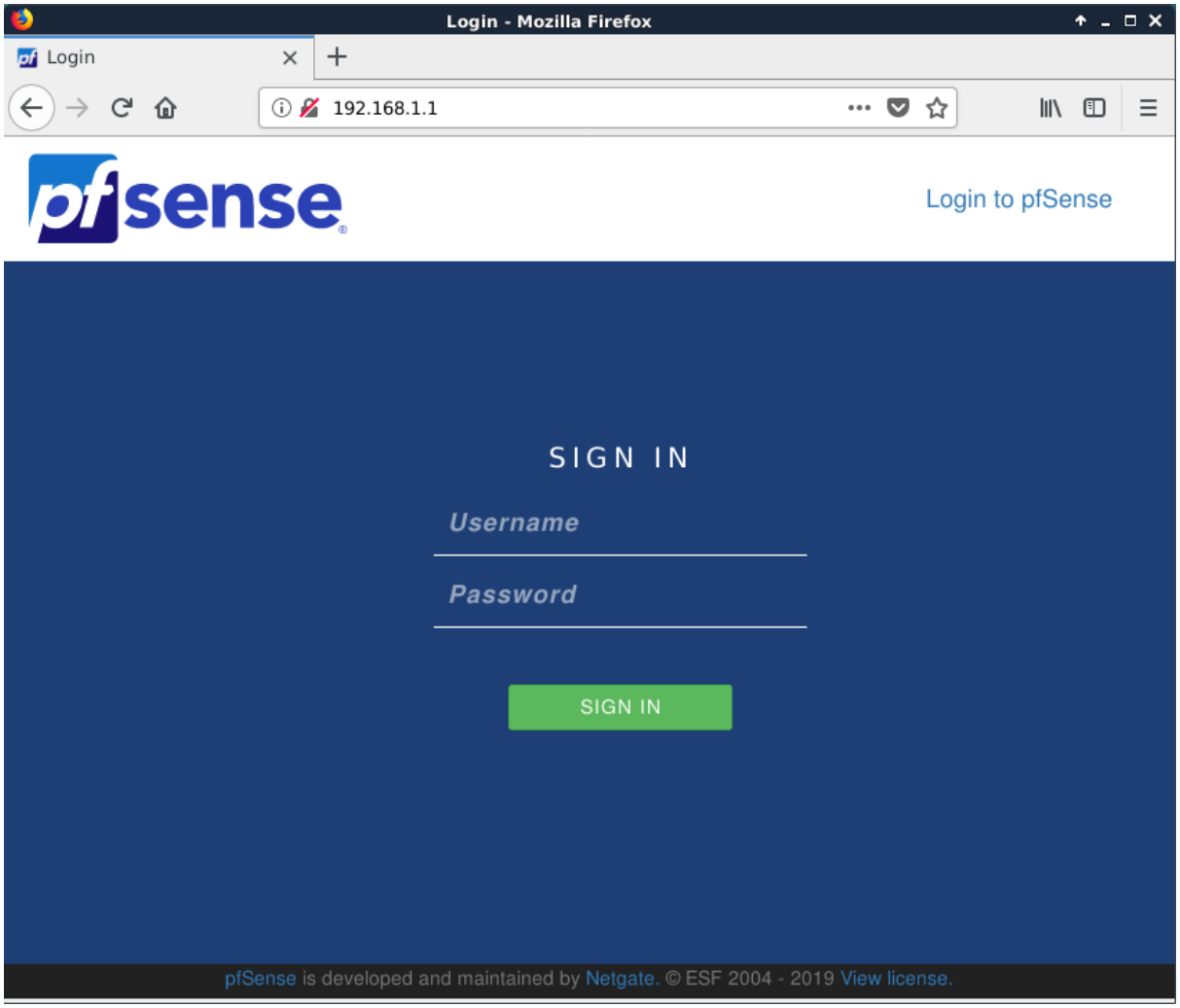


Pour vérifier si la configuration réseau s'est bien passée, nous pouvons essayer de ping notre interface **WAN** du routeur.



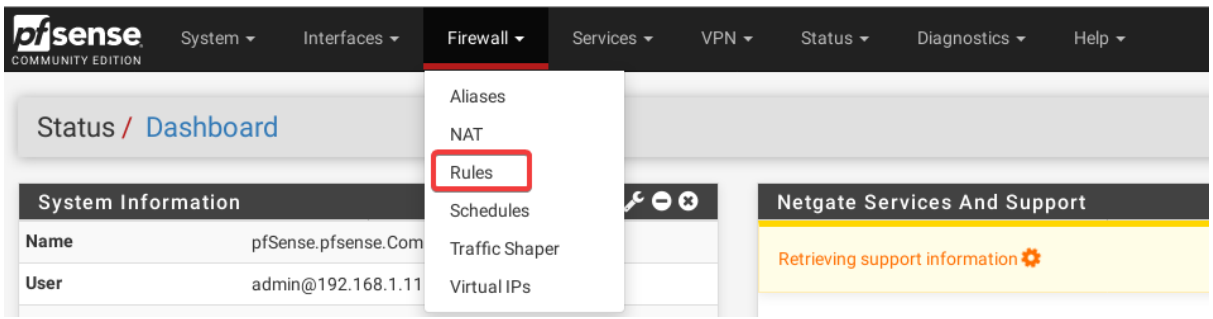
Maintenant que toutes les machines sont installées, il faut paramétrer notre pare-feu.

PfSense ne laisse passer aucun protocole, par défaut. Il nous faut autoriser seulement ceux qui sont nécessaires, pour limiter les failles de sécurité.
Pour ce faire, nous nous rendons à l'IP de notre routeur dans un navigateur web à partir d'un poste client (ou serveur).

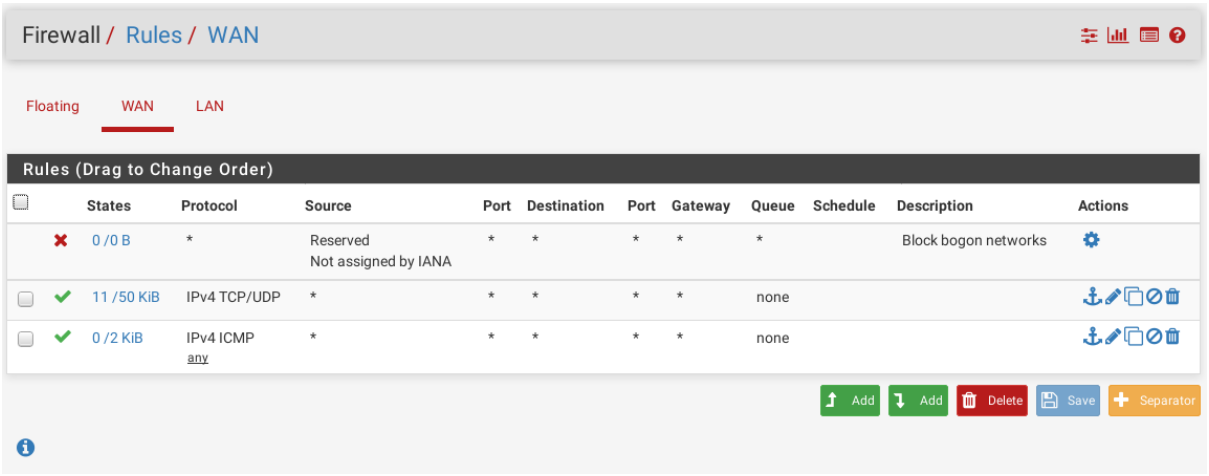


(Les identifiants de base sont ***admin*** et ***pfsense***)

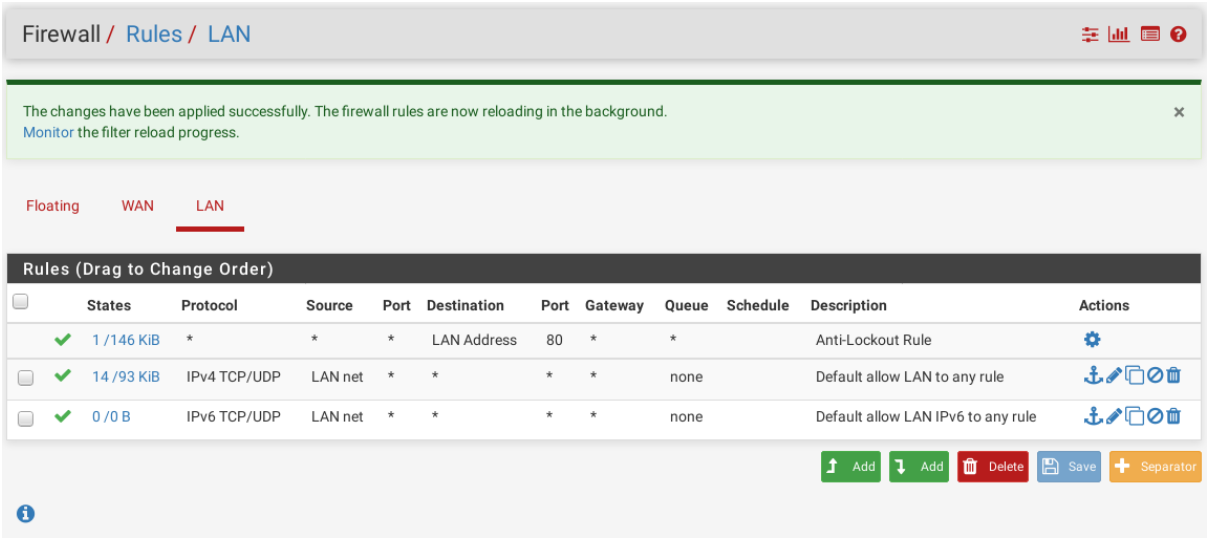
Une fois que nous nous sommes connecté et effectués les dernières configurations, nous pouvons nous rendre dans l'onglet **Firewall -> Rules**.



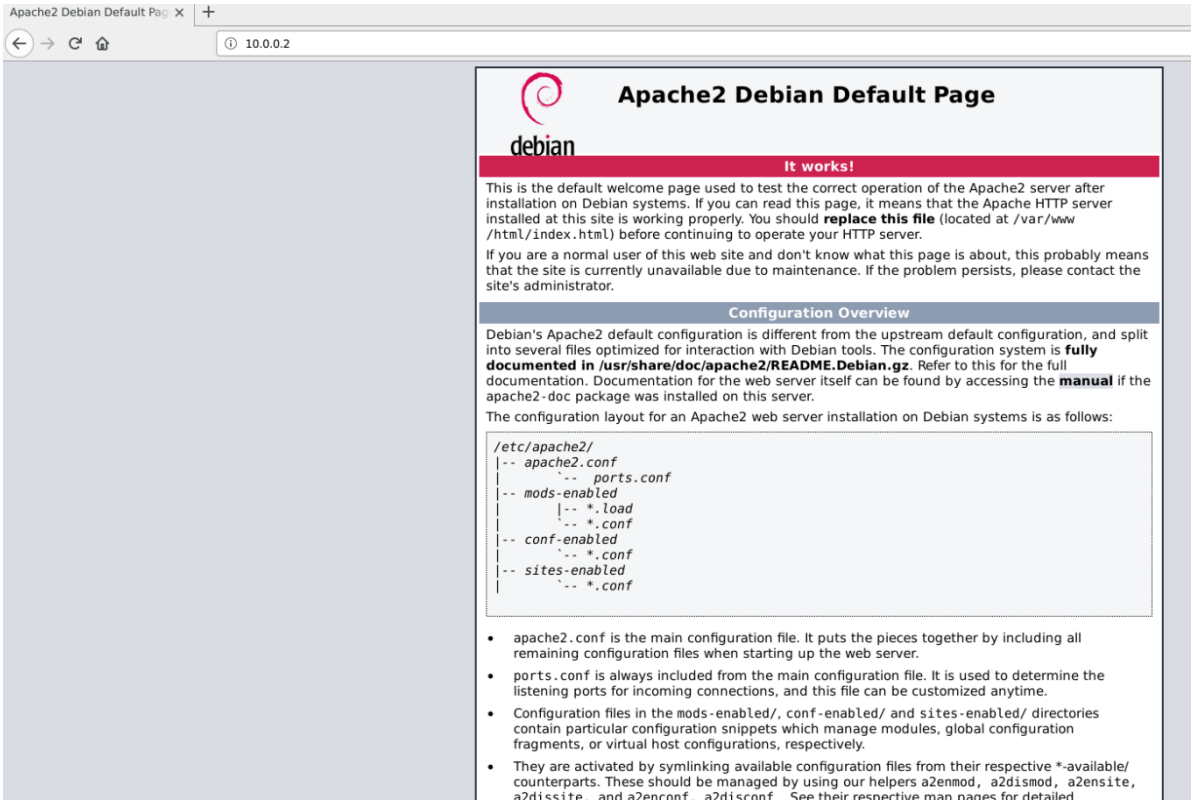
Pour autoriser les connexions aux sites du côté du **WAN**, nous autorisons les protocoles **TCP/UDP**, ainsi que le protocole **ICMP** pour autoriser le ping (pour les tests, cette règle sera supprimée par la suite).



Ensuite, pour le **LAN**, nous autorisons la même chose.



On vérifie ensuite que l'on arrive bien à accéder au serveur depuis un poste client.



Pour finir, on définit notre ***PfSense*** en tant que serveur ***DNS*** pour associer nos noms de domaines avec les IP de nos serveurs.

Host Overrides

Host	Parent domain of host	IP to return for host	Description	Actions
	ouiche.com	10.0.0.3		
	oulche.com	10.0.0.2		

Voilà, notre architecture est prête, nous pouvons dès à présent commencer à manipuler.

3. Solutions Logicielles

3.1. Contexte

Maintenant que notre installation réseau est prête, nous pouvons commencer à réfléchir aux solutions que nous allons mettre en place pour arriver à nos fins.

Pour rappel, les étapes que nous devons appliquer pour répondre au problème :

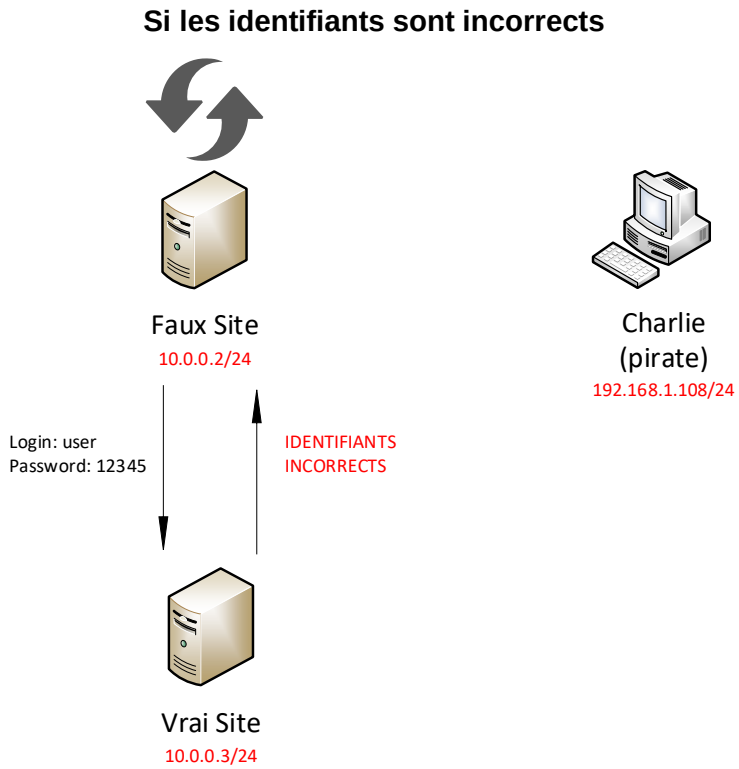
1- Intercepter le trafic entre une machine client et le routeur
2- Lors d'une requête vers le vrai site, rediriger vers le faux site
3- Tester et récupérer les identifiants rentrés
4- Les stocker dans un fichier.

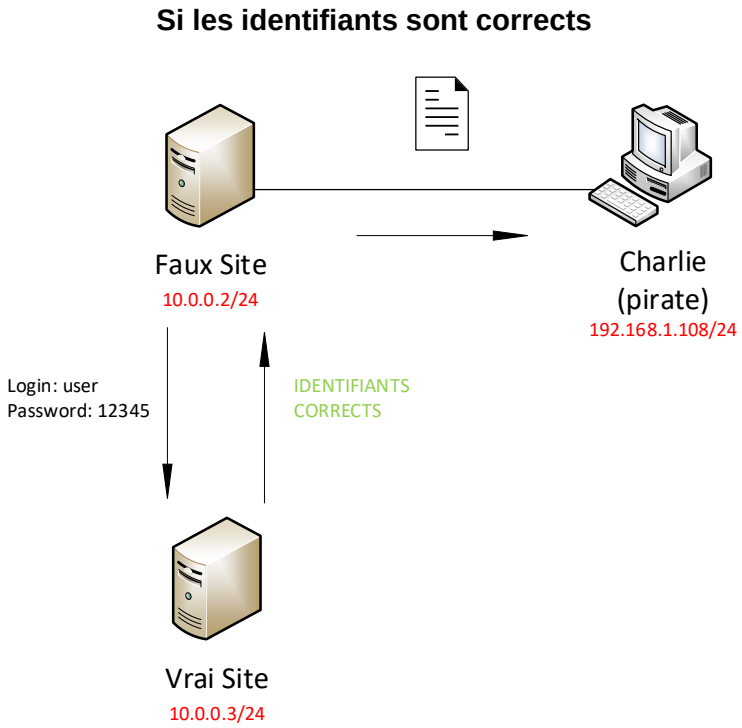
Et tout ça…sans que le client ne se doute de quelque chose.

Selon le scénario, le pirate a main mise sur un faux site, qui ressemble au vrai, via lequel il pourra récupérer les informations envoyées par le client qui a, auparavant, été redirigé dessus.

Le faux site, quant à lui, offre une page de connexion, très similaire à celle du vrai site qu'il usurpe, qui permet, une fois les identifiants rentrés, de tester ceux-ci à travers une **API** que proposera le vrai site.

L'outil qui nous sert à récupérer ces informations doit être conçu en ***Python*** avec la bibliothèque ***Scapy***, et doit disposer d'une interface graphique.





3.2. Développement des sites

Pour des raisons légales, nous ne pouvions pas avoir un accès à Internet pour *spoofer* un site existant. Il nous fallait donc créer un Vrai et un Faux site fictif.

Le premier (Vrai site) met à disposition une page de connexion qui vérifie, une fois entrés, les identifiants, qui sont stockés dans une base de données MySQL. Celui-ci redirige ensuite vers la page d'accueil.

Le deuxième site (Faux) offre une page de connexion très similaire à la première mais vient ici tester les identifiants à travers une API du vrai site. En effet, dans une situation réelle, la site pirate n'a pas d'accès direct à la base de données du site cible. Nous voulions reproduire cette situation de la manière la plus fidèle possible.
Cependant, par manque de temps, nous n'avons pas pu réaliser cette API comme convenu.

Pour le développement de ces sites, nous avons utilisé les technologies web courantes, HTML / CSS / PHP. Nous ne voulions pas consacrer du temps à la formation sur un nouveau langage ou une nouvelle technologie que ce n'était pas le but principal de ce projet.

3.3. Développement de l'outil

Le script que nous créons doit pouvoir effectuer certaines actions :

- "***Spoofer***" l'adresse MAC du routeur/client (se faire passer pour…)
- Rediriger vers le faux site (en modifiant les requêtes envoyées)
- Récupérer les identifiants de connexion et les stocker dans un fichier

Nous allons donc procéder par étapes.

Premièrement, pour la première partie, nous devons faire en sorte de s'interposer entre la source et la cible des informations (le client et le routeur).

Pour cela, nous effectuons une attaque de type ***ARP Poisoning*** (ou ***ARP Spoofing***).
Nous envoyons donc des fausses requêtes ARP à notre cible pour fausser sa table ARP.

Mais pour que les paquets atteignent notre machine, il nous faut activer l'***ip forwarding***.
Sans cela, les paquets seraient bloqués et ne passeraient pas notre machine.
(Pour que cela soit possible, il faut exécuter le script en *root*).

```
#Activate IP Forwarding
print("\n[*] Enabling IP Forwarding...\n")
os.system("echo 1 > /proc/sys/net/ipv4/ip_forward")
```

Il faut aussi penser à faire une fonction pour remettre en état les tables ARP. Dans le cas contraire, la victime se rendra compte que quelque chose ne va pas.
Ce script envoie des requêtes ARP en boucle, jusqu'à ce qu'on le stoppe. Dans ce cas, il remet en état les tables ARP victimes.

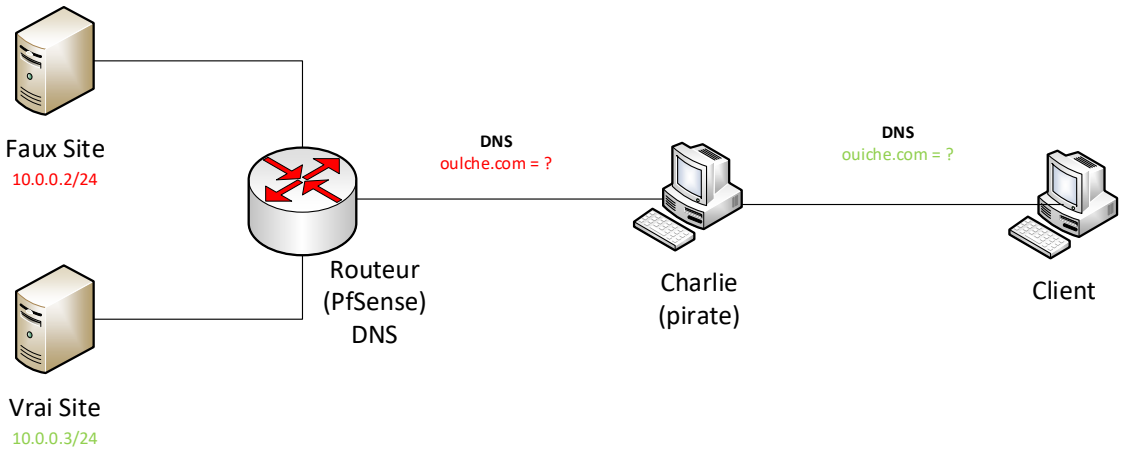
(c.f : *mitm-scapy.py*)

Maintenant, passons à la redirection.

Maintenant que nous pouvons avoir accès à tout le trafic passant entre la machine client et le routeur, nous pouvons exercer des actions sur ces paquets, comme les archiver ou encore les modifier, et c'est ce dernier point qui va nous intéresser.

Nous allons donc créer un script qui nous permet de modifier ces paquets, mais plus précisément, les requêtes DNS. En effet, cela était plus abordable que les redirections **HTTP** car, ce protocole repose sur le principe de ***Handshake*** qui nécessite un échange de 3 requêtes avant la connexion.
Modifier une simple requête DNS était donc plus aisé.

Premièrement, nous avons décidé de modifier la requête DNS du client pour remplacer le vrai nom de domaine par le faux. De ce fait, la requête envoyée vers le serveur DNS demandera notre domaine "pirate" et le client recevra donc notre IP "pirate".



Le problème rencontré fut que, Scapy seul, ne pouvait pas modifier les paquets sur leur envoi ("*On the fly*"). Il peut seulement les *sniffer* et agir sur une "copie".
Les paquets, une fois passé dans la machine pirate, atteignent leur destination et il est impossible de les bloquer pour les modifier.

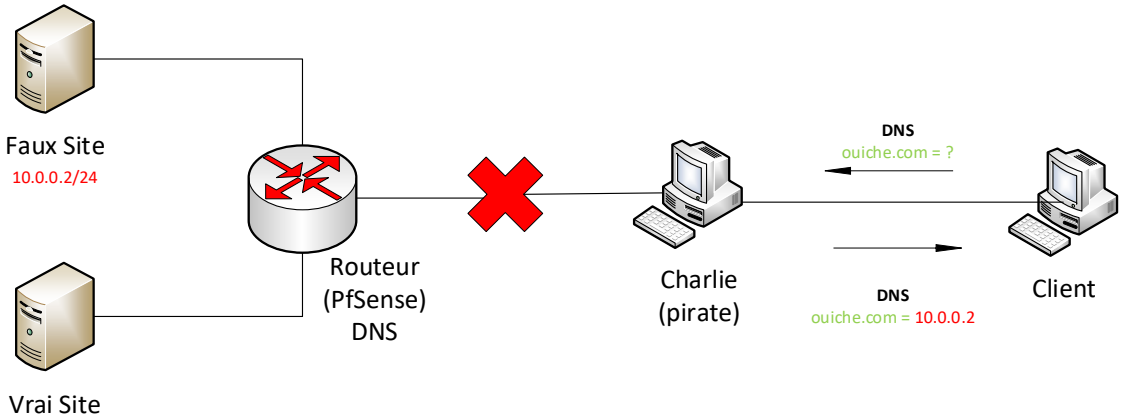
C'est pour cela que nous avons utilisé ***NetFilterQueue***, qui est à la base une extension C (qui dispose d'un wrapper en python) qui permet de mettre en file d'attente des paquets, de les "*drop*", les accepter, etc…

Mais, malgré cela, la redirection n'a pas fonctionné, les paquets étant bien modifiés, leur intégrité revérifiée et l'adresse IP et MAC de destination cohérentes, ils n'arrivaient pas à revenir vers la machine client.

(c.f : *alterDNSQR-scapy.py*)

Nous avons donc décidé de tenter une autre approche.

La deuxième approche que nous avons eue par la suite, fut de *spoofer* le serveur DNS. C'est-à-dire que nous interceptons la requête DNS et, au lieu de la transmettre au réel serveur, nous répondons directement à la machine client en lui associant le vrai nom de domaine demandé avec notre adresse IP "pirate".



Pour réaliser ce 2ème script, nous avons aussi eu recourt à ***NetFilterQueue***, pour altérer les paquets en provenance du client (toujours en prenant soin de recalculer le *checksum* ainsi que la *length* du packet).

Malgré cela, encore une fois, les paquets n'atteignaient pas la machine client et celle-ci en renvoyait en boucle. Nous pensons qu'une vérification du paquet par l'hôte ne se fait pas de manière naturelle et que celui-ci le rejette.

(c.f : *spoofDNS-scapy.py*)

Le principal problème qui nous doit cet échec est le manque cruel de documentation et/ou de communauté autour de Scapy.

La plupart des outils que nous utilisons regorgent de documentation, bibliothèques, etc… Cependant, sur ce projet, il nous a été quasiment impossible de trouver de l'aide de la part d'autres développeurs, que ce soit dans nos connaissances ou en ligne.

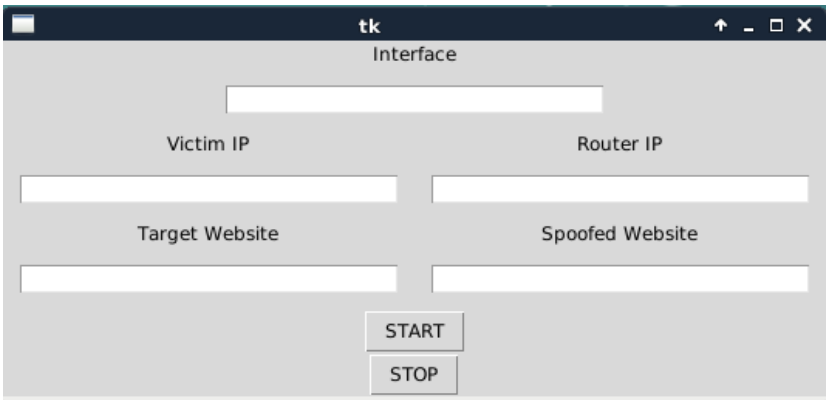
La redirection ne fonctionnant pas, nous avons décidé d'intercepter les requêtes vers le vrai site pour récupérer les identifiants.

Pour ce faire, nous avons juste eu besoin de ***Scapy*** et de sa fonction *"sniff"*.
Dans notre script, nous filtrons les requêtes POST vers le site, et extrayons l'utilisateur et le mot de passe en manipulant les chaînes de caractères pour les envoyer vers un fichier texte.
Si ceux-ci sont déjà présents dans notre liste, ils sont ignorés.

(c.f : *catchCredentials-scapy.py*)

Pour terminer, nous avons incorporé une interface graphique à notre outil, pour permettre de centraliser les actions et de simplifier les manipulations.

Pour cela, nous avons utilisé le module ***Tkinter*** qui est une bibliothèque graphique pour Python qui est assez souvent utilisé dans les projets.



Dans ce GUI (**G**raphical **U**ser **I**nterface), nous rentrons l'interface sur laquelle nous souhaitons réaliser l'attaque, l'adresse IP de la victime ainsi que l'adresse IP du routeur.

Les identifiants et mots de passes, une fois récupérés, sont enregistrés dans un fichier texte à l'aide de notre script précédent.

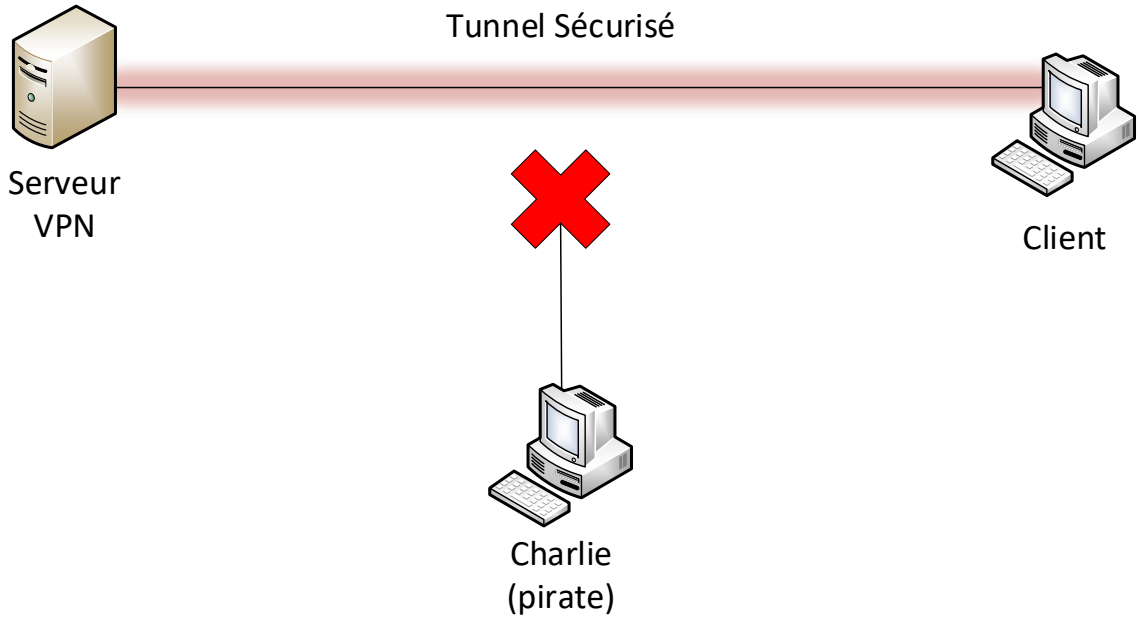
Des simples boutons START et STOP nous permettent d'aisément lancer ou arrêter cet outil.

Voilà pour ce qui est de l'outil développé. Bien entendu, celui-ci peut être amélioré et optimisé, mais nous nous sommes contentés de son bon fonctionnement pour ne pas dédier notre temps à des tâches superflues pour ce projet.

4. Prévention

4.1. Solutions numériques

Pour éviter d'être la cible de ce genre d'attaque, nous pourrions utiliser un service ***VPN*** (**V**irtual **P**rivate **N**etwork) qui fait passer toutes les informations que l'on envoie, dans un tunnel chiffré. Ces dernières seront irrécupérables pour le pirate.



Si l'on souhaite utiliser un VPN pour protéger ses informations personnelles, il faut bien regarder ce que propose les différents services disponibles.

De manière optimale, il faudrait prendre un VPN qui ne sauvegarde pas les logs de connexion, où sont stockées les informations ayant transitées via le service. Ces informations pourraient très bien être revendues à des sociétés.
C'est ce que l'on retrouve beaucoup chez les serveurs VPN à bas prix, voire gratuit.

D'autres logiciels tels que les antivirus permettent de détecter ce genre d'attaque, de vérifier les doublons d'adresse MAC ou IP dans les tables ARP et bien d'autres fonctionnalités encore.

" ***Xarp***" par exemple, détecte les attaques ***ARP Spoofing***.

Mais si nous devions nous reposer uniquement sur des solutions logicielles, nous ne parviendrions pas à assurer notre sécurité. Il faut pour cela, connaître certaines pratiques qui limitent les risques.

4.2. Bonnes Pratiques

Comme dans la plupart des systèmes d'informations, même en ayant toutes les sécurités nécessaires, le maillon le plus faible du schéma restera l'humain. Il faut donc adapter son comportement pour limiter les attaques.

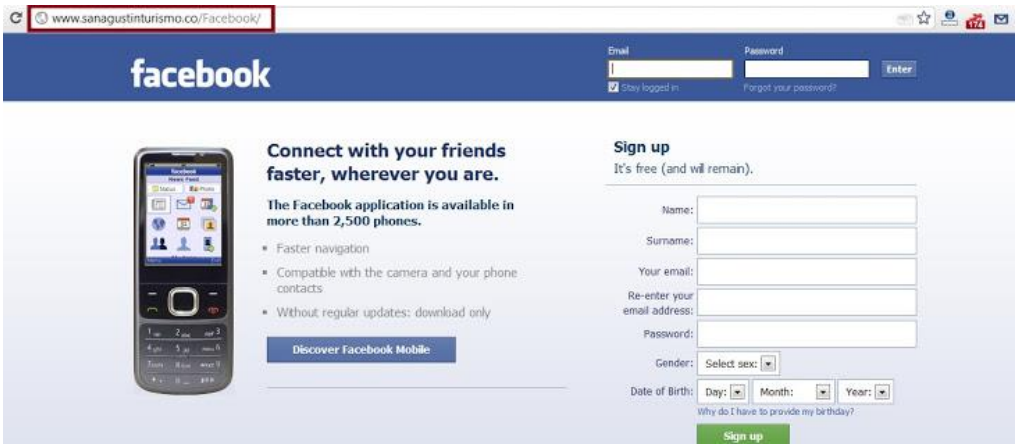
"*Un homme averti en vaut deux.*"

Le seul moyen de vraiment se protéger contre ces attaques reste de connaître la façon dont celles-ci s'exécutent et réagir en amont.

Tout d'abord, ne **jamais** visiter des sites avec des informations sensibles sur un réseau public sans protection voire même, si nous n'en disposons pas, ne pas aller sur ces sites.

Ensuite, examiner rapidement le réseau sur lequel nous nous trouvons. Plus celui-ci est grand, plus il y a de chances qu'une personne mal intentionnée tente de nous dérober des données.

Vérifier la barre d'adresse de site web reste aussi une excellente solution. Autant s'assurer que nous sommes sur le site souhaité en regarder si le nom de domaine correspond bien.



(Pour aider, certains navigateurs différencies par la couleur le nom de domaine de l'URL, pour ne pas se faire avoir).

5. Gestion de projet

5.1. Organisation

Pour mener à bien ce projet, il nous fallait une organisation bien définie car, étant assez conséquent et le temps nous étant assez limité, il ne fallait pas perdre notre temps sur des questionnements de logistiques.

Dès les premiers temps d'échanges, nous avons découpé le projet en plusieurs grands axes :

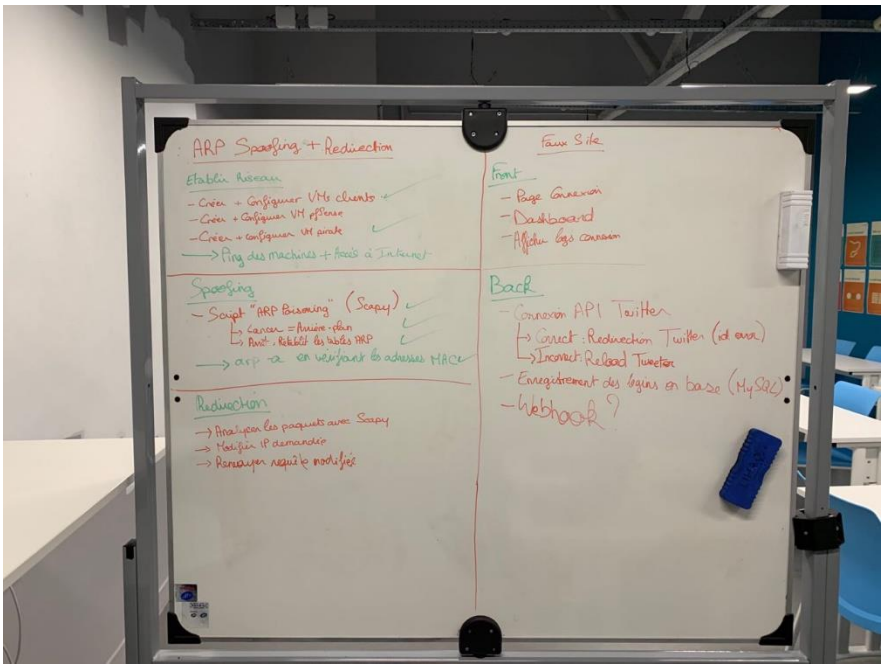
- Architecture Réseau
- Développement Web
- Développement des scripts

Chaque sujet fut ensuite abordé en *brain-storming* pour avoir un maximum d'approches différentes du problème et ainsi décuplé nos possibilités.
Puis, l'idée la plus adaptée fut retenue et découpées en étapes pour que l'on puisse avoir une vision sur l'avancement global du projet.

Pour centraliser ces informations, nous utilisions un tableau blanc dans notre salle de cours, qui nous permettait à la fois d'avoir un support commun et visible pour tous les membres du groupe, mais aussi d'avoir la qualité d'être plus simple à utiliser que n'importe quel outil web. C'était un gros atout lors de nos prises de notes, schémas, etc…

Un dépôt Git a été créé pour les sites web et un autre pour les scripts. Cela nous permettait de centraliser et sauvegarder notre travail, en plus d'avoir un historique.

Un groupe de discussion *Telegram* nous permettait aussi de communiquer depuis l'extérieur notre avancée, les idées et les nouvelles.



Ayant déjà expérimenté cette façon de travailler los d'un précédent stage de 2 membres du groupe, nous avons décidé de la remettre en place pour ce projet-ci.

Pour ce qui concerne la répartition des tâches, cela a été fait en fonction des connaissances et préférences de chacun.

- **L'architecture réseau** a été réfléchie par l'ensemble du groupe
- **Le développement web** a été assuré par Henry et Alexandre
- **Le développement des scripts** a été fait par Adrien

Bien sûr, dans chacune de nos tâches, si nous avions besoin d'une aide pour avancer, les autres membres du groupe pouvait très bien venir nous porter secours. C'est aussi un des avantages de travailler en groupe dans un même espace de travail et de parler ensemble.

Une certaine transparence a aussi été tenue, de manière que tout le monde soit au courant des avancées de chacun et des points sur lesquels il fallait accentuer les recherches/travail.

La question de la répartition du temps fut assez ardue.

En effet, ce projet était le premier projet de cette importance dans ce domaine. Aucun de nous n'avait expérimenté cela auparavant. L'évaluation de la charge de travail et de la durée de chaque tâche fut compliquée car nous n'avions aucun moyen de comparaison.

De plus, la période de travail a été entrecoupée d'un *Workshop* au sein de l'école qui nous obligeait à nous consacrer à lui pendant une durée de 2 semaines. Ce qui nous a sorti de l'esprit de ce projet

Un diagramme de Gantt aurait pu être établi pour mettre à l'écrit notre planning cependant, vu que nous ne savions pas vraiment à quoi nous attendre, celui-ci aurait été modifié constamment et l'organisation initiale n'aurait pas été respectée.

En conclusion, malgré quelques soucis, nous avons réussi à atteindre la plupart de nos objectifs durant cette période, et nous continuerons sûrement à travailler dessus par la suite car c'est un sujet assez complet et intéressant, qui pourrait très bien être utilisé dans le monde professionnel.

6. Annexes

6.1. Sites Web

- How to Build a Man-in-the-Middle Tool with Scapy and Python « Null Byte :: WonderHowTo
- How to Build a Man in the Middle Script with Python
- Comprendre les attaques via ARP spoofing (MITM, DOS) - Information Security
- Installation de Pfsense | pfSense | IT-Connect
- Configuration de pfSense (SSH, DNS, DHCP, parefeu, port forwarding) - LABRAT
- Download and Installation — Scapy 2.4.0-dev documentation
- Manipulez les paquets réseau avec Scapy - OpenClassrooms
- Scapy quick demo
- How to Build a DNS Packet Sniffer with Scapy and Python « Null Byte :: WonderHowTo

6.2. Livres

- "Les Réseaux 9ème édition" **PUJOLLE 2018**
- " CCNA Routing and Switching ICND2 200-105" **Wendell Odom 2013**

6.3. Ressources

Tous les scripts, ainsi que les fichiers sources de nos sites sont disponibles dans une archive, en pièce jointe avec ce document.