

Licence Professionnel ASRALL

Projet tuteuré.



Ufwi

Groupe :
– Cyril PIERRÉ
– Maxime ROBIN
– Valentin FROLICH
– Simon BAROTTE

Sommaire

1	Introduction	4
1.1	Historique	4
1.2	Présentation	4
2	Liste des solutions de pare-feu par identification	6
3	Externalisation des logs dans une BD MySQL	8
4	Fonctionnement d’ufwi	10
4.1	Algorithme	10
5	Daemon ufwi-authd	12
5.1	Introduction	12
5.2	Intallation	13
5.3	Sécuriser l’installation	13
5.3.1	Vérification des certificats	13
5.3.2	Côté client	14
5.4	Quelques commandes liées au daemon	14
5.5	Fichier de configuration	14
5.6	Installation des certificats	15
5.7	Configuration basique du daemon	15
5.8	Procédure de test	16
5.8.1	Paramétrage Iptables	16
5.8.2	Test du système d’authentification	16
5.8.3	Déroulement de l’authentification	16
5.8.4	Utilisation du module LDAP	17
5.9	Suivi des connexions avec SQL	17
5.10	Authentification par fichier locale	18
6	Daemon ufwi-filterd	20
6.1	Introduction	20
6.2	Intallation	20
6.3	Compilation	21
6.4	Commandes	21
6.5	Fichier de configuration	21

7	Daemon ufwi-rpcd	22
7.1	Introduction	22
7.2	Installation	22
7.2.1	Pré-requis	22
7.3	Utilisation	23
7.3.1	Démarrage	23
7.3.2	ufwi_rpcd_client (nucentral_client)	23
8	Client Ufwi/nufw	25
9	Difficulté rencontré et avis personnel	28
9.1	Daemon d'authentification	28
9.1.1	Problèmes rencontrés	28
9.1.2	Avis personnel	28
10	Planning	29

1

Introduction

1.1 Historique

Il faut savoir que le projet Ufwi descend du projet NuFW. La première version publique de NuFW est sortie le 01 septembre 2003. L'idée avait germé en 2001 mais ne s'était concrétisée que le 01 septembre de l'année 2003. La société croît rapidement, créant une vingtaine d'emplois en région parisienne, et acquérant une réputation dans le milieu de la sécurité informatique. En 2009, l'entreprise décide d'abandonner le pôle service pour se concentrer sur l'édition d'un pare-feu clef en main, EdenWall, couplant NuFW avec une interface d'administration performante, cette action entraîna des difficultés financières lourdes à la société. Le 18 août 2011, soit sept ans après sa création, le tribunal du commerce de Paris a prononcé la liquidation de la société, de ce fait la fin du projet NuFW.

Cependant le projet fut repris sous le nom de Ufwi, offrant une reprise optimale du projet en opérant un rassemblement du code, notamment éparpillé chez les clients de la société, qui évite les mois de travail nécessaires à la récupération des modifications ayant eu lieu depuis cette date.

1.2 Présentation

Ufwi est une solution de pare-feu par authentification, cela signifie qu'il effectue une authentification de chaque connexion qui le traverse. L'authentification se fait de façon transparente, en requérant les informations d'identification via l'annuaire des utilisateurs d'un réseau (LDAP ou autre). Donc avec Ufwi chaque utilisateur doit décliner son identité à chaque initialisation de connexion. Une fois que l'utilisateur est identifié, le paquet se voit filtré, selon les droits associés à l'utilisateur. Une fois que le premier paquet est accepté, les paquets suivants appartenant à la même connexion sont gérés par le système de suivi d'état (seul le paquet d'initialisation d'une connexion est authentifié). Ce principe de fonctionnement requiert la présence d'un client sur le poste utilisateur, il existe des clients compatibles pour Windows et GNU/Linux. Ufwi dispose aussi d'un mode sans client, pour se faire Ufwi utilise un module "ufwi-authd" (utilisé dans le cadre de notre projet).

Comme vu précédemment Ufwi, chaque connexion est associée à un utilisateur. Ce qui veut dire que plusieurs utilisateurs peuvent travailler sur un même poste génèrent simultanément des flux à partir d'une même adresse IP source.

Donc Ufw permet le filtrage par utilisateur mais peut aussi apporter d'autre fonctionnalité intéressantes.

- Ufw permet de contribuer de manière très pointue à la surveillance de l'activité réseau des serveurs. En règle générale dans les installations modernes on attribue des utilisateurs distincts à chaque service que fait tourner le serveur, de ce fait avec Ufw on peut attribuer une politique réseau différente pour chaque utilisateur système.
- Ufw peut marquer chaque paquet d'une connexion avec les informations de son utilisateur (identifiant...) ce qui permet d'appliquer une politique de qualité correspondant à chaque utilisateur. Cela permet donc de distribuer la bande passante entre chaque utilisateur, ainsi attribuer une bande passante plus élevée à l'utilisateur qui utilise des applications plus gourmandes en termes de connexion.
- Ufw comprend un module de surveillance qui journalise les événements principaux se produisant sur le réseau en indiquant les utilisateurs à l'origine de ses flux (ouverture, fermeture de connexion, paquet bloqué, etc...). Ces logs peuvent être gérés au choix par Syslog, PostgreSQL ou MySQL. Ufw conserve chaque connexion ouverte (ou tentée) même si l'utilisateur en question a changé d'adresse IP ou de machine.
- Ufw permet d'identifier les utilisateurs des applications réseau grâce à une authentification unique (Single sign on). Grâce au système de journalisation SQL une table de connexion est maintenue en temps réel.

2

Liste des solutions de pare-feu par identification

Les pare-feu par identification les plus connus :

- AuthPF : Fonctionne sous OpenBSD et qui se repose sur SSH pour l'identification des utilisateurs : <http://www.openbsd.org/faq/pf/authpf.html>
- NuFW : projet ayant donné naissance à UFWI suite à la liquidation de l'éditeur "Eden-Wall Technologies"
- Cyberoam : pare-feu entièrement basé sur l'identification, en utilisant une corrélation entre adresse MAC et utilisateur : <http://www.cyberoam.com/fr/firewall.html>
- CheckPoint (NAC Blade) : utilisation des règles de filtrage en fonction d'une authentification basée sur Kerberos, l'identité de son poste et du niveau de sécurité du poste (mise à jour de sécurité / antivirus) : <http://www.cyberoam.com/fr/firewall.html>

Solution parefeu	Avantage	Inconvénient
Ufwi	<ul style="list-style-type: none">– Projet devenu libre.– Appliquer des règles horaires strictes.– composé de deux démons qui peuvent être mis en place sur des systèmes différents et le démon principal est massivement multithreadé.– Contrôle d'accès sont réalisées grâce à des greffons (des modules system, ldap, dbm, plaintext).– Journalisation de l'activité des utilisateurs peut être faite par syslog, mysql, postgresql.– Multi plate-forme.– Génération des acl grace a un module (nuaclgen)	<ul style="list-style-type: none">– Tres peu de documentation voir aucune (obligé de cf. à nufw).– Nécessite une partie cliente spécifique sur les postes clients.– Payante sous windows (Nu-WinC).– Projet peu soutenu (temps de reponse 20-30 jours).

AuthPF	<ul style="list-style-type: none">– règles par système et/ou par utilisateur.– Connection via SSH.– chaque utilisateur peut disposer de ses propres règles, présentes dans HOME/.authpf/authpf.rules.– Simple a metre en place.– C'est un Firewall de type filtre de paquet (Couches réseau et transport).– Documentation et suport en français et assez simple.	<ul style="list-style-type: none">– l'utilisateur se déconnecte, le pare-feu lui retire les règles ajoutées.– quelqu'un d'autre peut toujours spoofer cette IP et passer.– Necessite la creation d'un home pour chaque utilisateur.– Necessite Openbsd et des connaissance PF.
Cyberoam	<ul style="list-style-type: none">– Sécurise les connexions IP dynamiques (Wi-Fi) et les postes de travail partagés.– Les politiques basées sur l'identité de l'utilisateur permettent d'éviter les erreurs rencontrées avec les politiques basées sur les adresses IP.– Prend en charge la création de groupes en fonction du profil professionnel sur l'ensemble des sites distribués.	<ul style="list-style-type: none">––
CheckPoint	<ul style="list-style-type: none">– Permet définitions de politique granulaires par utilisateur et de groupe.– L'intégration transparente avec Active Directory.– Idéal pour la protection des environnements avec les médias sociaux et des applications Internet.	<ul style="list-style-type: none">––

3

Externalisation des logs dans une BD MySQL

Configuration du serveur BD

Installation des paquets :

```
apt-get install apache2 php5 mysql-server nolog
```

Configuration de la passerelle :

Configuration IP :

```
ifconfig eth0 192.168.1.137/24 ifconfig eth1 172.20.8.1/24
```

Installation des paquets :

```
apt-get install ulogd ulogd-mysql
```

Correction d'un bug : ajout d'une ligne dans le script de démarrage qui va charger un module

```
nano /etc/init.d/ulogd export LD_PRELOAD = /usr/lib/libmysqlclient.so.16
```

Configuration de ulogd : modification de son fichier de configuration

```
nano /etc/ulogd.conf
```

Décommenter la ligne 46 (pour charger un module supplémentaire)

Renseigner les informations de connexion à la base de données :

```
paragraphe «[MYSQL]» ligne 59 : table="ulog" pass="passulog" user="ulog" db="ulog" host="172.20.8.2"
```

Configuration du serveur de BD :

Configuration IP :

```
ifconfig eth0 172.20.8.2/24
```

Lister tous les fichiers installés à l'installation de nolog :

```
dpkg -L nolog | more
```

Ouvrir le fichier suivant (démarche à suivre pour créer les tables de la base de données)

```
nano /usr/share/doc/nolog/README.Debian
```

Connexion à la base de données et création de l'utilisateur (les deux programmes vont se connecter avec ce compte) :

```
mysql -u root -p create database ulog; create user 'ulog'@'%' identified by 'passulog'; grant all privileges on ulog.* to ulog; exit
```

Commandes de création de la base :

```
cd /usr/share/doc/nolog/scripts gunzip ipv4.sql.gz cat ipv4.sql | mysql -uulog -p ulog
```

Modification du fichier de configuration de mysql

```
nano /etc/mysql/my.cnf ligne 47
```

Il faut qu'il écoute sur l'interface 172.20.8.2

```
bind address= '172.20.8.2'
```


Renommer les fichiers de configuration :

```
cd /etc/nulog cp default.core.conf core.conf cp default.nulog.conf nulog.conf cp default.wrapper.conf wrapper.conf
```

Renseigner les informations de connexion à la base de données :

```
nano core.conf host=localhost db=ulog user=ulog password=passulog table=ulog
```

Prise en compte des changements : redémarrage de services Sur la passerelle :

```
/etc/init.d/ulogd restart
```

Sur le serveur :

```
/etc/init.d/ulogd restart
```

On choisit ce que l'on veut loguer avec iptables

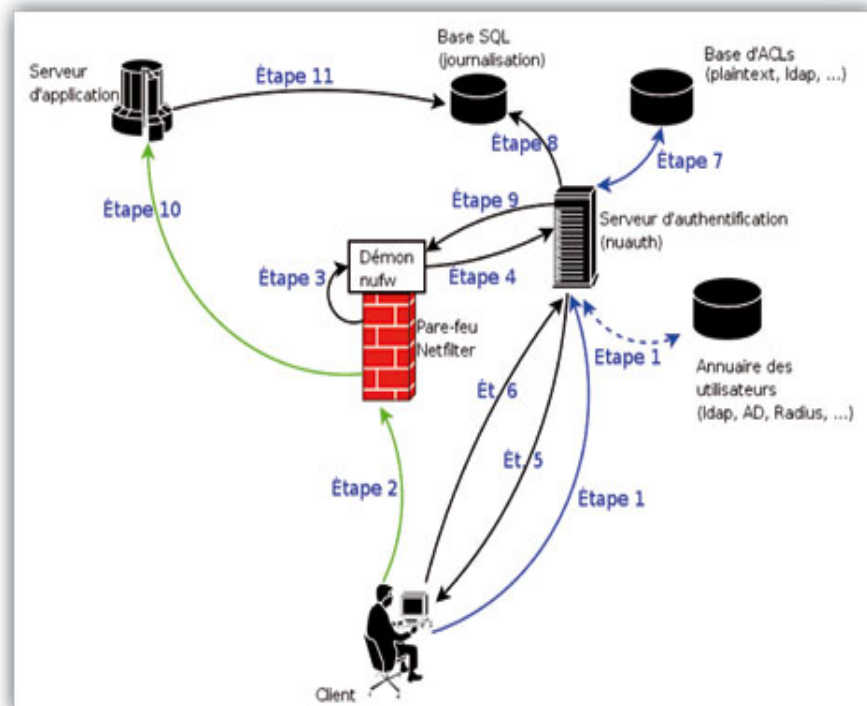
4

Fonctionnement d'ufwi

4.1 Algorithme

1. Au démarrage de sa session de travail, l'utilisateur lance un client sur son poste de travail. Le client ouvre un tunnel crypté par TLS vers le serveur nuauth qui réalise l'authentification de l'utilisateur pour ce tunnel au regard d'un annuaire référent. Ce tunnel est ensuite conservé pendant toute la durée des opérations et l'ensemble des échanges entre nuauth et le client se fait ensuite par son intermédiaire.
2. Supposons maintenant que l'utilisateur ouvre une connexion vers un serveur, par exemple une connexion web à destination du serveur d'application. Il utilise alors son navigateur favori qui envoie un paquet à destination du serveur. Le paquet envoyé est un paquet TCP standard avec notamment comme caractéristiques port destination 80 (web) et bit SYN positionné (ouverture de connexion).
3. Le pare-feu (Netfilter) envoie ce paquet au démon nufw (au moyen de la décision QUEUE ou NFQUEUE) : contrairement à un pare-feu " classique ", Netfilter ne prend pas de décision à propos de cette connexion.
4. Le démon nufw se contente de relayer le paquet reçu vers le serveur d'authentification, qui a autorité sur les décisions.
5. Le démon nuauth analyse le paquet reçu (et en particulier l'adresse IP source) et envoie une demande de mise à jour à tous les clients connectés (par l'étape 1) depuis cette adresse pour qu'ils authentifient les connexions en attente. Cette demande est envoyée au travers du (ou des) canal (canaux) crypté(s) ouvert(s) à l'étape 1.
6. Le client qui a ouvert notre connexion " témoin " stipule à nuauth qu'il l'a fait, en précisant tous les paramètres IP (IP et port destination, port source, etc.). À ce stade, le serveur nuauth connaît, de manière sûre, l'identité de l'utilisateur à la source de notre connexion.
7. nuauth réalise une requête sur la base des ACL, pour vérifier si l'utilisateur dispose des

- droits pour établir cette connexion. nauth obtient ainsi une décision.
8. (En option) nauth journalise toutes les informations relatives à cette connexion dans une base SQL, avec bien sûr l'identité de l'utilisateur.
 9. nauth envoie la décision obtenue en 7 à nufw, qui la relaie à son tour à Netfilter. La décision associée à notre utilisateur est alors appliquée par Netfilter.
 10. S'il est autorisé, le paquet poursuit sa route.
 11. (En option) Si le serveur (Apache, par exemple) désire connaître l'identité de l'utilisateur, au lieu de la lui demander directement, il peut réaliser une simple requête SQL SELECT pour récupérer l'identifiant de l'utilisateur en partant de marqueurs uniques de la connexion (la socket source, pour les connaisseurs). L'utilisateur est ainsi authentifié de manière transparente, et sans pouvoir tricher, sur le serveur : il s'agit d'une solution de Single Sign On (authentification unique) très simple, très sûre, et indépendante du protocole.
 12. Le reste du flux est géré par le suivi de connexion (Stateful inspection) de Netfilter : les paquets ne repassent pas par les démons nufw/nauth.



5

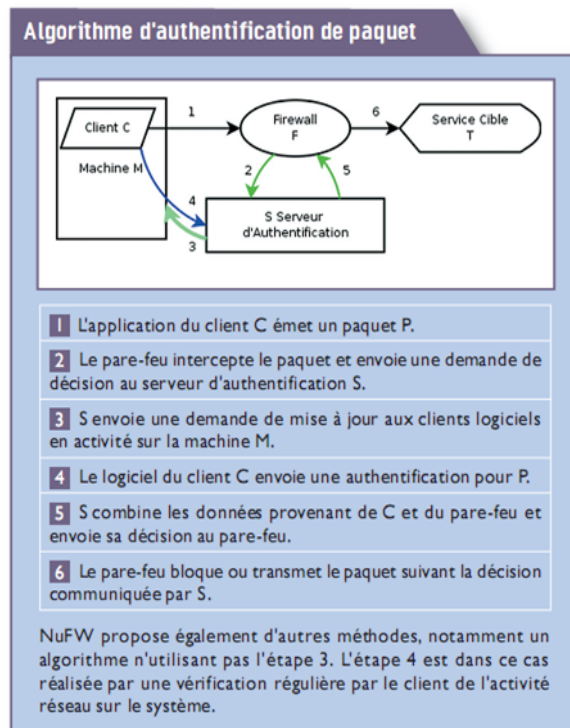
Daemon ufwi-authd

5.1 Introduction

Nuauth command est une interface qui permet de contrôler des fonctions importantes du daemon authd, comme l'obtention de la liste des utilisateurs connectés par exemple. Chaque fois qu'un client envoie un paquet(1) pour commencer une connexion à travers la passerelle, la station cliente envoie un paquet(2) d'identification au daemon authd. Le pare-feu de la passerelle met en file d'attente le paquet et envoie directement des informations au daemon authd.

Le travail du daemon va être d'analyser les deux paquets (1) et (2) et de vérifier si le client a le droit d'initialiser la connexion qu'il demande. Si ufwi-authd indique que le paquet(1) est autorisé alors la connexion est initialisée, sinon la connexion est annulée. Ufwi-authd peut aussi utiliser un serveur LDAP pour la définition des utilisateurs et groupes.

Ci-dessous le un schéma montrant le processus d'authentification utilisé par NuFW, resté inchangé avec UFWI :



5.2 Intallation

Pré-requis :

Script autogen.sh :

- version automake1.7

Compilation Nufw :

- GNU libtool
- GNU make
- libpam-dev
- glib 2.4+
- libipq (iptables-dev pour debian) ou libnetfilter queue
- libldap
- libsasl2
- libgnutls
- libgcrypt

Noyau :

Il est recommandé d'utiliser un noyau récent afin de bénéficier de toutes les dernières nouveautés implémenter dans ce dernier. Une version de noyau supérieur à 2.6.18 est un bon choix. Le patch dump-connection-mark.diff (disponible dans patches/) peut être appliqué au noyau afin d'améliorer les performances de ce dernier lorsque nous utiliserons le log de session.

Compilation :

La compilation du daemon est relativement simple, elle se déroule en quatre étapes :

- Lancement du script ./autogen.sh
- Exécution de ./configure
- make
- Et pour finir make install

Lors de la première installation, il faut penser à copier le fichier de configuration avec la commande suivante :

```
cp ./conf/nuauth.conf /usr/local/etc/nuauth.conf
```

5.3 Sécuriser l'installation

5.3.1 Vérification des certificats

Il est particulièrement recommandé de placer nuauth dans un endroit protégé afin de garantir la sécurité des communications entre nufw et nuauth¹. Dans la mesure où la décision du pare-feu dépend de la réponse de nuauth, il est important de pouvoir valider l'identité du serveur nuauth. Pour cela, on peut demander à nufw de vérifier le certificat présenté par nuauth lors de l'établissement du tunnel TLS. Ceci peut être mis en place grâce à l'option -a suivie du nom du fichier contenant le certificat d'autorité racine. Cette option est ajoutée à la ligne de commande démarrant nufw. Ce faisant, nufw vérifiera la validité du certificat présenté par nauth.

1. Même si tous les paquets sont chiffrés par TLS

5.3.2 Côté client

Du côté du client, le système doit être intègre pour que les informations concernant les applications et le système d'exploitation soient pertinentes. On doit toujours garder à l'esprit que seul l'agent installé côté client est capable d'obtenir ces renseignements. En cas d'attaque, il est évident que ces informations PEUVENT et SERONT faussées par l'installation d'un agent NuFW modifié.

On doit tenir compte de cet avertissement et ne surtout pas oublier que cette fonctionnalité permet de sécuriser des flux qui auraient dû être ouvert sans vérification sur un système basique.

La pertinence du filtrage d'application et/ou de système d'exploitation dépend de la confiance que l'on place dans le système qui réalisera l'authentification. Elle est "relativement bonne" sur un système sécurisé sur lequel les utilisateurs ne peuvent installer de logiciels

5.4 Quelques commandes liées au daemon

Commandes principales :

- quit : déconnexion
- refresh cache : rafraichit tous les caches
- reload : recharge la configuration du daemon d'authentification

Information :

- help : affiche la liste des commandes utilisables
- version : affiche la version du daemon
- uptime : affiche depuis combien de temp tourne le daemon

Gestion des utilisateurs :

- users : affiche els utilisateurs connectés
- disconnect all : déconnecte tous les utilisateurs
- disconnect ID : déconnecte un utilisateur grâce à son identifiant (ID)

5.5 Fichier de configuration

Le fichier authd.conf est le fichier principal de configuration pour le daemon ufwi-authd. C'est dans ce fichier que seront indiqué l'adresse du daemon ufwi-filterd par exemple ou encore le niveau de debug, le nombre de connexion qu'un utilisateur peut lancer. Dans ce fichier seront aussi renseigné les différents paramètres qui guide le comportement du daemon, mais aussi les paramètres système, et pour finir les chemins absolus des autres fichiers de configuration.

Il existe aussi d'autres fichiers de configurations liés à ufwi-authd :

- modules/nuauth-tls.conf qui contiendra les paramètres TLS
- modules/nuauth-krb5.conf configuration authentification Kerberos 5
- modules/nuauth-ldap.conf authentification ldap
- modules/nuauth-mysql.conf configuration de la base de donnée pour les logs utilisateurs (mysql)
- modules/nuauth-pgsql.conf configuration de la base de donnée pour les logs utilisateurs (postgres)

5.6 Installation des certificats

Le daemon d'authentification et le client utilise des certificats afin de communiquer ensemble. Lors de l'installation de la suite ufwi des certificats sont installés par défaut dans le répertoire `/certs`. On peut utiliser ces certificats par défaut pour procéder à des tests mais il n'est pas conseillé de les utiliser pour une utilisation pro de la suite ufwi. Dans cette section nous allons voir comment générer ces certificats.

Génère notre propre Autorité de Certification :

```
mkdir private
```

```
chmod 700 private
```

```
openssl req -new -x509 -keyout private/CAkey.pem -out private/CAcert.pem
```

Génère les clefs privées pour le client et le daemon d'authentification :

```
openssl genrsa -out private/nufw-key.pem
```

```
openssl genrsa -out private/nuauth-key.pem
```

Génère les demandes de certificats pour le client et le daemon :

```
openssl req -new -key private/nufw-key.pem -out nufw.csr
```

```
openssl req -new -key private/nuauth-key.pem -out nuauth.csr
```

Signe les demandes de certificats grâce à l'AC :

```
openssl x509 -req -days 365 -in nufw.csr -CA private/CAcert.pem -CAkey private/CAkey.pem -CAcreateserial -out nufw-cert.pem
```

```
openssl x509 -req -days 365 -in nuauth.csr -CA private/CAcert.pem -CAkey private/CAkey.pem -CAcreateserial -out nuauth-cert.pem
```

Enfin on déplace les certificats dans le bon répertoire :

```
cp private/nufw-key.pem /etc/nufw/
```

```
cp nufw-cert.pem /etc/nufw/
```

```
cp private/nuauth-key.pem /etc/nufw/
```

```
cp nuauth-cert.pem /etc/nufw/
```

5.7 Configuration basique du daemon

Lors de l'installation de la suite ufwi un fichier de configuration de base est fournie `nuauth.conf`, qui est disponible dans le répertoire `conf`. Ce fichier est remplie de plusieurs directives, les deux plus importantes directives de configuration sont : `nuauth listen addr` : qui définit l'adresse à laquelle le daemon va attendre les requêtes des clients `nuauth` `nufw listen addr` : qui définit l'adresse à laquelle le daemon va attendre les requêtes de `nufw`. La liste des machines autorisées à se connecter au serveur d'authentification constitue la variable `nufw gw addr`.

Ensuite, on doit choisir le module d'authentification et de vérification des ACL. Les modules suivants sont disponibles :

- `libldap` : les informations utilisateur sont stockées dans un annuaire LDAP
- `dbm` : les informations utilisateur sont stockées dans une base gdbm
- `plaintext` : les informations utilisateur sont stockées dans un fichier texte
- `system` : l'authentification s'adosse à PAM et utilise les groupes existants dans le système.

Ceci procure un moyen pratique d'utiliser `nss` et/ou `pam-modules`

Le module d'authentification est paramétrable via l'option `nuauth user check module` dont la valeur par défaut est `libsystem` (si non défini dans le fichier de configuration). D'autres paramètres concernant la vérification des ACL doivent être précisés si on choisit une des deux authentification

suivante :

- libldap
- plaintext

Tout ceci en définissant la variable `nuauth acl check module`.

5.8 Procédure de test

Lors du projet j'ai monté une machine virtuel avec une image iso d'une machine Debian avec nufw pré-installé dessus. J'ai réussi à faire fonctionner nufw deux fois sur cette machine virtuelle avant d'obtenir des erreurs, je vais donc dans cette section détaillé la procédure que j'ai suivi.

5.8.1 Paramétrage Iptables

Tout d'abord j'ai définie les règles Iptables comme suit de façon à déclencher une demande d'authentification pour toute connection vers ssh :

```
iptables -A OUTPUT -s 192.168.1.0/24 -p tcp -dport 22 -m state --state NEW --syn -j NFQUEUE
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

5.8.2 Test du système d'authentification

Concernant la configuration de nuauth j'ai gardé la configuration de base fourni avec l'image ISO. En premier lieu, j'ai démarré le service nuauth dans un terminal :

```
nuauth -vvvvvvvvv (les v en argument corresponde au niveau de debug du daemon, ici niveau 9, niveau max)
```

Ensuite, on démarre nufw dans un autre terminal, avec ici aussi le niveau de debug max afin d'avoir le plus d'informations possible sur le daemon.

```
nufw -vvvvvvvvv
```

Enfin, on peut essayer de connecter un utilisateur. On utilise nutpc pour réaliser ceci avec la commande :

```
nutpc -d -H [NUAUTH IP] ou NUAUTH IP correspond à l'adresse IP de la machine qui exécute nuauth.
```

Dans le terminal qui exécute nuauth, nous voyons apparaître une ligne de ce stile :

```
user simon@nufw uses OS Linux, 3.0.10, 1 Tue Mar 19 23 :00 :32 CEST 2012
```

Cette ligne nous donne le nom de l'utilisateur qui se connecte ici simon l'OS utiliser par cette utilisateur ainsi que la date et l'heure.

5.8.3 Déroulement de l'authentification

La connection SSH va déclencher la procédure d'authentification :

- nufw reçoit un paquet de Netfilter :
[PID] Sending request for 3352783904
- nufw ouvre une connexion TLS vers nuauth :
[PID] Trying TLS connection
- nuauth reçoit la requête de nufw :
** Message : Packet :
** Message : Connection : src=192.168.1.12 dst=192.168.1.12 proto=6
** Message : sport=32848 dport=22

- nauth envoie une demande d'authentification au client en fonction de l'IP source :
 - ** Message : need to warn client
 - ** Message : sending request
- nauth reçoit la réponse du client :
 - ** Message : User :
 - ** Message : Connection : src=192.168.1.12 dst=192.168.1.12 proto=6
 - ** Message : sport=32848 dport=22
 - ** Message : OS : Linux 2.6.9 1 Fri Mar 16 21 :51 :32 CEST 2012
 - ** Message : Application : /usr/bin/ssh
- nauth renvoie sa réponse à nufw :
 - Sending auth answer 1 for 3352783904 on 0x42428482 ...
- nufw renvoie le paquet dans le noyau :
 - [PID] Accepting 3352783904

Remarque :

Ne JAMAIS nutpc avec l'adresse [NAUTH IP] égale à localhost ou 127.0.0.1, même si nauth est installé sur la même machine. Dans ce cas, les paquets envoyés à nauth par le pare-feu proviendront de la machine (avec une adresse égale à, par exemple, 192.168.0.1) alors que nauth attend pour l'authentification une adresse égale à 127.0.0.1. Dès lors, l'authentification échouera systématiquement.

5.8.4 Utilisation du module LDAP

Après avoir réalisé un test de base de nauth, j'ai voulu tenter l'utilisation de LDAP pour vérifier les ACL. Malgré l'échec de cette configuration je vais tout de même mettre ma démarche dans cette section. Tout d'abord j'ai mis le fichier `acls.schema` dans le répertoire `/etc/ldap/schema` et j'ai ajouté la ligne suivante dans le fichier `/etc/ldap/slapd.conf` : `include /etc/ldap/schema/acls.schema`.

J'ai ensuite configuré nauth (`nauth.conf`) pour l'utilisation de LDAP comme suit :

Tout d'abord j'ai modifier la directive `nauth_acl_check_module` pour lui indiquer l'utilisation de `libldap`.

Ensuite, j'ai précisé les paramètres de connexion à l'annuaire :

```
ldap_bind_dn="uid=nauth,ou=Users,dc=inl,dc=fr"
```

```
ldap_bind_password="secretpassword"
```

```
ldap_basedn="dc=inl,dc=fr"
```

```
ldap_acls_base_dn="ou=Acls,dc=inl,dc=fr"
```

5.9 Suivi des connexions avec SQL

J'ai aussi configuré nauth pour permettre le suivi des connexions grâce à une base de données SQL. Pour commencer pour activer le suivi de connexion avec NuFW, j'ai paramétré les options suivantes dans le fichier `nauth.conf` :

```
nauth_log_users_sync=1
```

```
nauth_log_users=8
```

Pour journaliser en MySQL, il faut avoir une base contenant la bonne structure. Pour ce faire on peut utiliser le fichier `nulog.mysql.dump` du répertoire `conf` et les commandes suivantes (dans un terminal) :

```
mysqladmin create nulog
cat conf/nulog.mysql.dump/mysql nulog
```

Il suffit ensuite de créer un utilisateur MySQL qui aura les droits SELECT, DELETE, UPDATE et INSERT sur la table. Enfin, ajoute les informations de connexions au serveur SQL dans le fichier `nuauth.conf`, en modifiant les directives suivantes :

```
mysql_server_addr="localhost"
mysql_server_port="3306"
mysql_user="nuauth" (ici c'est le nom que j'ai donné à l'utilisateur SQL)
mysql_passwd="root"
mysql_table_name="nulog" (ici on renseigne le nom de la table créée ci-dessus)
```

Après différente recherche sur la journalisation des connexions avec SQL je suis tombé sur une discussion d'utilisateur, cette discussion avait pour sujet l'utilisation d'un script `clean_conntrack.pl` qui est disponible dans le répertoire `scripts` à partir de la version 1.0.12. Ce script doit être exécuté très régulièrement, à intervalles de quelques minutes seulement, par l'intermédiaire de `cron`, notamment en cas de trafic important. La seule opération réalisée par le script est de transférer les connexions "mortes" (ie les connexions fermées ou refusées) vers une autre table, qui sera une table d'archivage. Il faut envisager d'archiver régulièrement la table d'archivage, afin d'éviter qu'elle ne grossisse indéfiniment.

5.10 Authentification par fichier locale

Il existe un autre moyen de spécifier des utilisateurs à `nuauth`. `Nuauth` considère que la base des utilisateurs est locale, dans un fichier plat, ce type de configuration est très simple à mettre en place, et est donc parfait pour réaliser des tests. Tout d'abord il va falloir vérifier quelques options importantes de notre configuration `nuauth`, dans `/etc/nufw/nuauth.conf`.

```
nuauth_user_check_module="plaintext"
```

C'est grâce à cette directive que `nuauth` va savoir que nous utilisons un fichier plat. Bien sûr, sur un réseau d'entreprise, on interface plutôt `nuauth` à l'annuaire (LDAP, Active Directory, Radius, etc.)

```
nuauth_acl_check_module="libplaintext"
```

Cette directive indique à `nuauth` de chercher les droits de chaque utilisateur dans un fichier texte également.

À présent, on crée notre fichier texte de définition des utilisateurs `/etc/nufw/users.nufw` avec le contenu suivant.

```
simon :simon :1 :102
admin :iadmin :2 :100,102
suadmin :suroot :3 :103
```

Le fichier définit donc trois utilisateurs : `simon`, `admin` et `suadmin`. Le deuxième champ (après le " : ") est le mot de passe. Le troisième champ est un identifiant propre à chaque utilisateur. Enfin, le dernier champ définit la liste des groupes (numériques) auxquels l'utilisateur appartient. Ici, donc, l'utilisateur `simon` a comme mot de passe `simon`, comme identifiant 1 et appartient au groupe 102.

Bien évidemment, ces utilisateurs sont totalement indépendants des utilisateurs systèmes existants sur le pare-feu. Notre base d'utilisateur est ici parfaitement spécifique à NuFW. Voyons à présent les ACL. Nous allons créer le fichier `/etc/nufw/acls.nufw` avec le contenu suivant :

```
[http]
```

1 = acceptation, 0 = rejet

decision=1

groupes concernés par cette décision

gid=102

L'identifiant du protocole, tel que décrit par /etc/protocols. Ici il s'agit de TCP == 6

proto=6

SrcIP=0.0.0.0/0

SrcPort=1024-65535

DstIP=192.168.1.12

DstPort=80

Ici on autorise donc les membres du groupe 102 à surfer sur le site web de la machine 192.168.1.12. Dans notre exemple, il s'agit des utilisateurs simon et admin.

Recharge la configuration du serveur nuauth :

killall -HUP nuauth

à présent, lance le client :

nutcp -H IP_nuauth -U user

Si tout se passe bien, le mot de passe concernat l'user choisit est demandé. Pour tester il suffirait de faire un simple telnet sur la machine 192.168.1.12 sur le port 80 et vérifier que la connexion s'ouvre bien.

6

Daemon ufwi-filterd

6.1 Introduction

Le daemon ufwi-filterd (anciennement appelé nufw) n'est d'autre qu'un pare-feu basé sur NFQUEUE netfilter. Il permet d'écrire des règles de filtrage basées sur l'identité des utilisateurs, en plus des critères de réseau classiques. L'authentification est effectuée de façon transparente en requérant les informations d'identification de l'utilisateur avant qu'une quelconque décision de filtrage ne soit prise. En pratique, cela signifie que les politiques de filtrage peuvent intégrer l'annuaire utilisateur, et amène cette notion d'ID utilisateur au niveau de la couche IP.

Ufwi-filterd est capable de :

- Filtrer le trafic en fonction du système d'exploitation et des applications utilisées par les utilisateurs distants.
- marquer chaque paquet d'une connexion avec l'identifiant de son utilisateur et donc d'appliquer une politique de qualité de service spécifique à chaque utilisateur.
- contribue de manière très pointue à la surveillance de l'activité réseau des serveurs.
- dispose de modules de surveillance qui journalisent les événements principaux de l'activité du réseau en indiquant quels sont les utilisateurs à l'origine des flux.

6.2 Installation

Une installation typique de la suite logicielle NuFW comporte 2 démons : nufw (ufwi-filterd) et nuauth (ufwi-authd) et autant de clients que nécessaire.

Pré-requis :

- automake1.7 pour exécuter autogen.sh
- GNU libtool
- GNU make

Pré-requis pour la compilation et l'exécution de ufwi-filterd :

- ufwi-base
- ufwi-confparser
- ufwi-ssl

Il est recommandé d'utiliser un noyau récent afin de bénéficier de toutes les dernières nouveautés implémentées dans ce dernier. Une version de noyau supérieure à 2.6.18 est un bon choix.

6.3 Compilation

La compilation de ufwi-filterd est relativement simple elle se resume a utiliser les commandes suivantes :

- ./autogen.sh
- ./configure
- make
- makeinstall

Lors de la première installation, il ne faut pas oublier de copier le fichier de configuration "make install-conf" afin de chercher les changements entre votre fichier de conf actuelle et le nouveau.

Un fichier INSTALL avec toutes les instructions a suivre est fournie dans le dossier de ufwi-filerd disponible a partir de se lien [http ://ufwi.org/projects/ufwi-filterd/ repository](http://ufwi.org/projects/ufwi-filterd/repository) .

6.4 Commandes

Tout dabord, vous devez executer en root ufwi-filterd. ufwi-filterd -h vous donnera un message d'aide pour l'utilisation de ufwi-filterd.

6.5 Fichier de configuration

Le fichier de configuration de ufwi-filterd se nome tout simplement "filterd.conf". On pourra le trouver dans /etc/ufwi-filterd/. Dans se fichier on trouvera l'adresse ou le nom du serveur d'authentification nuauth (par default 127.0.0.1), on trouvera aussi les chemin absolu des fichiers :

- /etc/ufwi-filterd/key.pem (clé privé du serveur)
- /etc/ufwi-filterd/cert.pem (certificat du serveur)
- /etc/ufwi-filterd/cacert.pem
- /etc/ufwi-filterd/crl.pem (liste de révocation de certificat serveur)

7

Daemon ufwi-rcpd

7.1 Introduction

Le module ufwi-rcpd (anciennement appelé NuCentral) est le module qui gère les autres daemons.

7.2 Installation

7.2.1 Pré-requis

Avant de lancer l'installation du module, certains pré-requis sont nécessaires :

- Python 2.5
- Twisted web
- M2Crypto
- Jinja
- Subversion (svnadmin program)
- sudo
- pysvn
- pytz

Debian :

```
apt-get install python-twisted-web python-svn  
python-m2crypto sudo python-jinja subversion python-tz
```

Pour l'installation de ufwi-rcpd :

- (GNU) make
- sqlite3

Sous Debian :

```
apt-get install make sqlite3
```

Paquets optionnels : Pour compiler les fichiers de .ts à .qm et mettre à jours les transmissions :

- lrelease4 : Qt development tools
- pylupdate4, lrelease-qt4 : Python Qt development tools

Sous Debian :

```
apt-get install libqt4-dev pyqt4-dev-tools
```

Autres :

- python-twisted-snmp : pour le module SNMP.
- gnutls-bin : le programme certtool l'utilise pour générer les certificats ssl.

- `pytest` (Sous Debian, `python-codespeak-lib`) : utilisé pour les tests.
- `libconfig-inifiles-perl` : pour `tools/ufwi_rpcd_enmod`.
- `IPy` (`python-ipy`) : pour les tests.

7.3 Utilisation

Une fois démarrer, on peut accéder au fonction plus particulaire de `ufwi-rpcd` (`nucentral`).

7.3.1 Démarrage

Lancement en mode daemon

Pour lancer `ufwi-rpcd` en mode daemon il suffit d'exécuter une commande en root :

```
twistd -y /usr/sbin/ufwi-rpcd.tac --pidfile=/var/run/ufwi-rpcd.pid  
-l /var/log/ufwi-rpcd-twisted.log
```

Pour arrêter `ufwi-rpcd` en mode daemon (en root) :

```
kill $(cat /var/run/ufwi-rpcd.pid)
```

Lancement en mode "premier plan"

Pour facilité le développement, on peut également lancer `ufwi-rpcd` en premier plan :

```
twistd -n -y /usr/sbin/ufwi-rpcd.tac -l /var/log/ufwi-rpcd-twisted.log
```

Pour arrêter le processus : CTRL+c.

7.3.2 `ufwi_rpcd_client` (`nucentral_client`)

Le client

Pour lancer le client d'`ufwi_rpcd` en HTTP (tcp/8080) :

```
ufwi_rpcd_client --host 127.0.0.1 --cleartext -u admin -p admin
```

Par défaut, `ufwi_rpcd_client` utilise HTTPS (tcp/8443) :

```
ufwi_rpcd_client --host 127.0.0.1 -u admin -p admin
```

Une fois connecté sur le client, le prompt devient :

```
nucentral>
```

Le menu

Une fois lancer, on peut accéder aux différentes fonctions d'`ufwi-rpcd` :

- `call('component', 'service', ...)`
Appelle un service d'`NuCentral`
- `authenticate('login', 'password')`
Authentification ou mise à jour d'un groupe ou d'un utilisateur.
- `components()`
Affiche la liste des différents composants.
- `services('component')`
Affiche la liste des services d'un composant.
- `proxy('component')`
Crée un composant pour le proxy qui pourra être utilisé avec `component.service()`.

- `nucentralStatus()`
Afficher l'état du serveur et la session d'informations.
- `help('component')`
Utilisation d'un composant.
- `help('component', 'service')`
Utilisation d'un service d'un composant.
- `help(proxy)`
Utilisation du proxy.
- `exit()`
Quitter.
- `pyhelp(object)`
Aide Python pour l'objet spécifié.

Différents composants sont disponibles :

`CORE, access, acl, audit, auth_cert, bind, config, contact, hostname, hosts, httpout, localfw, lock, logger, network, ntp, nuauth, nuauth_command, nuconf, nuface, nulog, nupki, nurestore, reporting, resolv, session, status, streaming, system, system_info, tools, update, users_config, versionning`

8

Client Ufwi/nufw

Comme vu précédemment dans l'introduction Ufwi a besoin d'un petit client sur chaque poste utilisant ce logiciel. Cependant il y'a un moyen de contourner ce problème sous GNU/Linux.

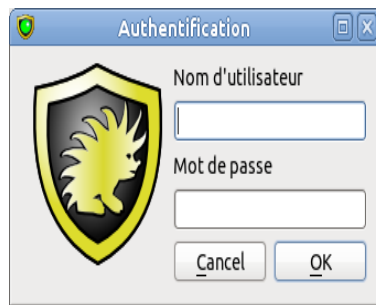
Il est possible de s'authentifier directement au serveur UFWI au moment de l'authentification de l'utilisateur sur un poste linux. Pour cela, il est nécessaire d'installer le paquet `pam_nufw` présent dans la plupart des distributions GNU/Linux. Il est possible de trouver les sources de ce logiciel dans les sources du projet NuFW. Pour configurer, il suffit de renseigner la ligne suivante dans `/etc/pam.d/common-auth` :

```
auth optional pam_nufw.so server=adresse_amon port=4129 noauth=root,mdp
```

et dans `/etc/pam.d/common-session` :

```
session optional pam_nufw.so server=adresse_amon port=4129 noauth=root,mdp
```

Un client graphique autonome est proposé librement par la société INL. Ce client, NuApplet2, est disponible dans la plupart des distributions GNU/Linux. L'adresse IP correspond à l'amon et le port correspond au service Nuauth (4129 par défaut).



Un autre client existe sous GNU/Linux Nutcpd celui-ci est celui que nous avons utilisé pour nos tests, c'est un client non-graphique uniquement en ligne de commande. Ce client est le client de base proposé dans l'iso de Nufw. Le client a besoin de certificats d'authentification SSL, pour les configurer deux solutions sont disponibles. La première (recommandée) est de mettre directement les certificats d'authentification dans le dossier c'est-à-dire dans le `/home/user/.nufw`. La deuxième solution consistant à définir les certificats d'authentification en paramètre de la commande.

Quelques commandes pour le client nutcpd :

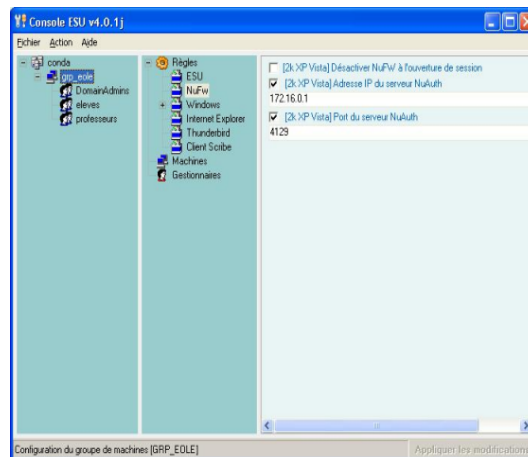
- h : Affiche les différentes commandes.
- v : Affiche la version du client.
- d : affiche le mode debug.
- S : utilise le SSL pour se connecter au serveur Nuauth.
- k : Avant de démarrer, le programme supprime tous les processus existants de l'utilisateur.
- l : ne vérifie pas les fichiers de logs avant de démarrer et n'en crée pas.
- H + IP serveur Nuauth : Envoie un paquet d'authentification au serveur.
- p + port du serveur Nuauth : Envoie un paquet d'authentification au serveur (port).
- U : Définit l'utilisateur (UserID).
- P : Set le mot de passe utilisateur.
- C : Définit le certificat qui autorise la connexion TLS au Nuauth.
- A : Définit l'AuthorityFile et vérifie la validité des certificats de Nuauth.
- k : Définit la KeyFile qui autorise la connexion TLS.

Un exemple typique pour tester si le client dialogue bien avec le serveur est de faire :

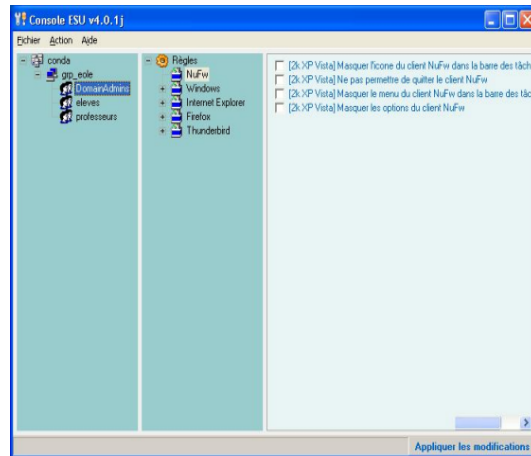
```
nutcpc -d 9 -H 192.168.1.137 -A Nuauth_cert.pem
```

Donc dans la commande ci-dessus on active le mode debug avec le plus haut niveau pour plus d'info en cas de problème ou d'erreur, ensuite on configure l'IP du serveur Nuauth et enfin on définit le certificat si il n'est pas déjà placé dans .nufw de l'utilisateur et vérifie la validité des certificats de Nuauth. Il existe toutefois un client nufw pour windows non libre pour le moment. NuWinC est

le client NuFW pour les OS de Microsoft, Windows. Le client NuWinC est intégré au client Scribe mais il est désactivé par défaut. Avant tout, il faut activer le client NuWinC. Cela se fait dans ESU. Après l'avoir activé, il faut paramétrer l'IP de l'AMon et le port du serveur d'authentification Nuauth (4129).



ESU permet de régler certains paramètres spécifiques à NuWinC. Il est, par exemple, possible de protéger ou de masquer le client NuFW.



9

Difficulté rencontré et avis personnel

9.1 Daemon d'authentification

9.1.1 Problèmes rencontrés

Malheureusement ils sont assez nombreux. Tout d'abord lors de l'installation du module initial (ufwi-authd), impossible de l'installer, lors du `make install` des erreurs ont été générées, aucune solution trouvée afin de résoudre ces soucis. Dès lors notre choix a été d'utiliser l'ancienne version, c'est à dire `nufw` (`nuauth`). Après une étude du fichier de configuration et le suivi de la procédure de test décrite dans la partie 5.8, j'ai réussi à faire fonctionner deux fois le module `nuauth` sur une machine virtuelle, dès le troisième lancement de `nuauth` j'ai rencontré un nouveau problème, une erreur liée au socket. Après de nombreuses recherches je n'ai pas réussi à résoudre ce soucis. J'ai retenté par la suite de relancer `nuauth` afin de refaire des tests, et l'erreur concernant les sockets avait disparu sans que je sache pourquoi, mais cette bonne nouvelle a fait apparaître un nouveau problème, une erreur de segmentation à la fin de l'initialisation de `nuauth`. J'ai effectué des recherches sur différents forums et il semblerait que cette erreur soit due à un bug de `nufw`, sans en trouver la correction. J'ai aussi rencontré des problèmes sur les certificats entre le client et `nuauth`, il marchait une fois sur deux sans savoir la non plus pourquoi.

9.1.2 Avis personnel

Pour ma part ce projet tuteuré a tout de même été une bonne expérience, malgré "l'échec" de l'utilisation de ce logiciel. `Nufw` peut être très puissant et réellement utile pour une entreprise. Une fois que ce logiciel fonctionne il est assez simple de l'utiliser, le seul problème je dirais est le manque de documentation concernant ce logiciel. Si l'on rencontre un ou plusieurs problèmes il est difficile d'en trouver les solutions. Concernant la reprise de `nufw` par FSF France (ufwi), je trouve que c'est une bonne chose, car la disparition de `nufw` aurait été vraiment dommage, car ce logiciel mérite sa place dans les réseaux d'entreprise même si il est peu connu. La encore le soucis est le manque de documentation et d'aide au sujet de ce nouveau projet ufwi, (les premiers documents d'aide concernant ufwi ont été mis en ligne sur le dépôt git du projet ufwi à la fin de la période de notre projet).

10

Planning

Maxime Robin :

- Semaine 4 à 8 (Du 23 janvier au 27 février) :
 - Mise en place du dépôt github.
 - Étude et installation du daemon ufwi-rpcd.
 - Commencement du rapport.
- Semaine 9 :
 - Étude de la durée de vie des autorisations données.
- Semaine 10 et 11 :
 - Installation de l'ISO de NuFW dans une machine virtual disponible sur le site de nufw.
 - Mise en place sur le réseau de la nouvelle machine.
 - Étude des fonctions préinstallées sur la nouvelle machine préinstaller.
- Semaine 12 :
 - Étude des différentes fonctions du module nucetral (ufwi-rpcd).

Simon Barotte :

- Début janvier :
 - Etude du daemon ufwi-auth.
 - Installation du daemon (sans succès)
- Mois janvier début février :
 - Rédaction de la partie ufwi-auth du rapport
 - Etude des erreurs lors de la tentative d'installation du daemon
- Mis février début mars :
 - Etude des commandes du daemon ufwi-auth après le succès de l'installation
- Mars :
 - Prise en main de mon daemon (ufwi-auth)
 - Mettre en place différent test et valider leur fonctionnement
 - Utiliser différentes configuration en tirer les avantages et inconvénients
 - Générer des certificats personnel pour nuauth
 - Test du système d'authentification
 - Réaliser premier test et débogage

Valentin Frolich :

- Semaine 4 à 7 :
 - Etude du daemon Ufwi-filtred.
 - Ecriture de la partie Daemon Ufwi-filtred du rapport.
 - Etude des prerequis et dependance pour l'instalation de la suite Ufwi.

- Semaine 9 :
 - Rédaction de la partit Introduction du rapport.
- Mars :
 - Tester different configuration de la suite Ufwi.
 - Simuler un environnement test (entreprise...).
 - Définir les limite, avantage, inconvénient de la suite.

Cyril PIERRÉ :

- Semaine 4 à 8 :
 - Mise en place d'un réseau virtualisé à l'aide de VirtualBox comprenant une passerelle entre le réseau ASRALL (192.168.1.0/24) et le réseau interne virtuel (172.20.8.0) et d'une seconde machine avec un annuaire LDAP, un serveur MySQL et un serveur de résolution DNS (bind9)
 - Étude des PyQT Managements Tools
 - Installation de l'ensemble des modules d'UFWI sur la passerelle
 - Recherche des différents pare-feu par authentification
 - Mise en place d'un tutoriel pour l'externalisation des logs de pare-feu avec Nulog
- Semaine 9 :
 - Correction d'un problème de RAM sur la passerelle
 - Étude de l'intégration d'UFWI avec d'autre logiciels d'authentification :
 - Kerberos