IP FIREWALL

TECHNICAL DOCUMENTATION

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

SOMMAIRE

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

INTRODUCTION

Pour réaliser ce projet, nous avons utiliser python ainsi que plusieurs librairies, soit scapy pour lire des paquets d'un fichier de capture, ainsi que netfilterqueue pour rediriger les paquets vers notre programme.

Pour réaliser à bien ce projet, nous avons donc un programme principal, qui va comparer chaque paquets reçus avec les règles que nous avons dans une base de données.

Un programme qui permet d'insérer des règles selon les critères voulus, on pourra insérer la position de la règle, si on veut ou non accepter le paquet selon cette règle etc…

Un dernier programme qui permet de supprimer une règle selon la position qu'il a dans la base de données (id).

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

PROCESSUS

Firewall – gestion des paquets

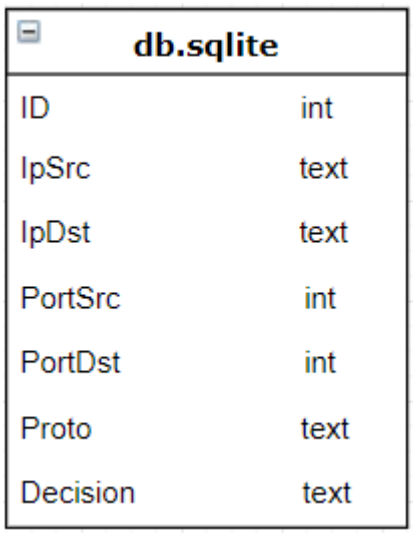
Création de règle

Suppression de règle

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

ARCHITECTURE

Nous avons une seule base de données dans laquelle nous stockons toute nos règles. On retrouve ainsi les éléments du protocole header : Id, ip source, ip de destination, protocole, port source et port de destination.



KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

DESCRIPTION DES ELEMENTS

Création de règle

```
Enter the different information about your rule:
Position: 1
Type: accept
Ip Source:
Ip Destination:
Port Source:
Port Destination:
Protocol: TCP
Actual rules:
```

Position : Demande à l'utilisateur la position à laquelle la règle doit s'insérer

Type : accept ou drop, si la règle match avec les éléments du paquet, alors il faudra soit accepter, soit jeter le paquet.

Ip Source : Demande à l'utilisateur s'il veut insérer une adresse ip source à accepter ou jeter

IP Destination : Demande à l'utilisateur s'il veut insérer une adresse ip de destination à accepter ou jeter

Port Source : Demande à l'utilisateur s'il veut insérer un port source à accepter ou jeter

Port Destination : Demande à l'utilisateur s'il veut insérer un port de destination à accepter ou jeter

Protocol : Demande à l'utilisateur s'il veut insérer un protocol (UDP, TCP, ICMP) à accepter ou jeter

Actual rules : affiche les règles qui sont dans la base de données, avec la nouvelle

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

Suppression de règle

```
Actual rules:

Position: 1 Type: drop IpSource:  IpDestination: 1.1.1.1 PortSource:  PortDestination:  Protocol:
Position: 2 Type: drop IpSource: 1.1.1.1 IpDestination:  PortSource:  PortDestination:  Protocol:
Position: 3 Type: accept IpSource:  IpDestination:  PortSource:  PortDestination:  Protocol: TCP
Position: 4 Type: accept IpSource:  IpDestination:  PortSource:  PortDestination:  Protocol: UDP
-----

Enter the ID of the rule you want to remove:
```

Affichage des règles stockées dans la base de données.

Demande à l'utilisateur de rentrer l'id de la règle qu'il veut supprimer dans la base de données.

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

EXPLICATIONS/FICHIERS

firewall.py :

La fonction « packet_manager » gère les paquets qui doivent être acceptés. Les paquets nfqueue du réseau, passés en paramètre sont converti en paquet Scapy. Ensuite, nous envoyons ceci à une autre fonction nommée check packet, qui renverra la décision, si nous devons accepter le paquet.

Nous créons deux queue nfqueue grâce à iptables en leur attribuant un numéro et un type (Input / Output).

Nous lions notre nfqueue deux fois, en leur assignant la fonction "packet_manager".

Nous lançons nfqueue, qui sera capable de détecter tous les paquets circulant dans notre réseau, et acceptera ou bloquera ces paquets.

A la fin, nous détachons nos queues nfqueue.

Check.py :

La fonction « get_protocol » retourne l'objet selon le protocole du paquet, ainsi que le nom de l'objet.

La fonction « rule_packet_match » compare chaque champ de la règle avec ceux du paquet en cours de vérification.

La fonction « check_packet » récupère toutes les règles de la base de données, en fonction de l'identifiant dans l'ordre croissant. Pour chacune des règles, nous comparons les champs avec le paquet courant grâce à la fonction "rule_packet_match". La fonction nous renvoie un booléen.

Connection.py :

Une connexion est établie avec notre base de données SQLITE, puis nous créer un curseur.

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

Pcap_program.py :

La fonction « packet_manager » gère les paquets qui doivent être acceptés. Nous envoyons ce paquet Scapy à une autre fonction nommée check_packet, qui renverra la décision, si nous devons accepter le paquet.

Nous récupérons tous les paquets du fichier "pcap" dans nos variables "paquets". Ensuite, nous créons une variable "count" qui sera incrémentée plus tard, afin de savoir quelle décision appartient à quel paquet. Puis, nous créons une itération qui va boucler pour chaque paquet de la variable "paquets". Le numéro de paquet et ses informations sont affichés. Enfin, nous envoyons notre paquet dans la fonction "packet_manager" pour connaître la décision, si nous acceptons le paquet.

Rule.py :

La classe « RuleConnectionClass » permet de créer des objets de règles que nous récupérons dans notre base de données. Ces règles, nous permettront de vérifier si nous pouvons accepter un paquet, grâce au fichier check.py. Cette classe possède six attributs, avec leur méthode d'accès : les adresses IP source et destination, les ports source et destination, le protocole de la règle et enfin la décision (Accept ou Drop).

La fonction « print_rules » extrait toutes les règles créées de la base de données et les affiche à l'utilisateur.

La fonction « get_field_not_null » renvoie le nombre de colonnes, sans valeur nulle, de la règle de paramètre.

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou

Rule_creation.py :

La fonction « rule_creation » crée une nouvelle règle, en demandant à l'utilisateur d'entrer différentes informations sur la règle. Chaque information entrée par l'utilisateur est stockée dans un dictionnaire. Avant d'envoyer cette nouvelle règle dans notre base de données, nous vérifions s'il y a des règles ayant le même identifiant ou ayant un numéro d'identification plus élevé. Si oui, nous allons incrémenter l'identifiant de ces règles, puis nous enverrons notre nouvelle règle dans la base de données. Enfin, nous affichons les règles de notre base de données.

Rule_remove.py :

La fonction « rule_remove » supprime une règle de notre base de données, grâce à l'id entrée par l'utilisateur. Avant d'envoyer cette nouvelle règle dans notre base de données, nous vérifions s'il y a des règles ayant un numéro d'identification plus élevé. Si oui, nous allons décrémenter l'identifiant de ces règles. Enfin, nous affichons les règles de notre base de données.

KAGRAMANOV Emile
CAPUANO Hugo
THELCIDE Dimitri
NDIAYE Jules François Amadou