UFA47

Sécurité des réseaux (intro)

SISR - Cybersécurité

fpitance@bunkerity.com

SOMMAIRE

- Principes et bonne conduite
- L’accès au réseau local
- L’accès aux services
- Le déni de service

UFA47

Principes et bonne conduite

UFA47

Principes et bonne conduite

La sécurité repose sur 4 piliers :

- La **disponibilité**,
- L'**intégrité**,
- La **confidentialité**,
- Et l'**authentification**.

Principes et bonne conduite

La mise en œuvre de la sécurité est une tâche complexe :

- Dépend des **biens à protéger** et des **risques** associés.
- Les **mesures** prises peuvent être **techniques** ou **organisationnelles**.
- Nécessite d'avoir une bonne **vision globale** de l'informatique …
- … mais aussi de connaître les **moindres détails** de certaines technologies.

Principes et bonne conduite

Il est nécessaire de connaître les **techniques utilisées par les pirates** pour mieux **s'en protéger**. Comme dit Sun Tzu : « *To know your Enemy, you must become your Enemy* ».

Cependant, un grand pouvoir implique de **grandes responsabilités**. Il va de soi que quelques **règles basiques** sont à respecter :

- On fait les différents tests sur les **équipements qui nous appartiennent** pas ceux des voisins.
- On **réfléchit à deux fois** avant de lancer une attaque (surtout si c'est sur le serveur de production …).
- Internet **n'est pas un terrain de jeux** : il y a des **lois**.
- Le port du **chapeau blanc** est **obligatoire**, le noir vous amènera en prison.
- Si vous trouvez une vulnérabilité, soyez **responsables** et **avertissez** la personne en charge du système.

Principes et bonne conduite

Quizz time !

Quels sont les piliers de la sécurité ?

- Disponibilité, Intégrité, Confidentialité et Authentification.

J’ai intégré les dernières technologies de cyber-sécurité à base d’intelligence artificielle, ça devrait être bon ?

- Il n’y a pas que les aspects techniques …
- … il y a aussi ceux organisationnels (ex : sensibilisation au *phishing*).

J’ai envie de tester le cassage de clé WEP, je fais ça sur le WiFi de mon voisin ?

- NON !
- ~~À la rigueur celui de mes copains pour les épater …~~
- Je préviens mon voisin/copain qu’il faut mettre du WPA2 (je fais une démo sur le WEP s’ils sont d’accord) !

L’accès au réseau local

UFA47

L’accès au réseau local

Une des lignes directrices en sécurité informatique est la **réduction de la surface d’attaque**.

Cela revient à **limiter au strict nécessaire les points d’entrées** possibles pour un attaquant (réseau, applicatif, système, …).

Et le premier point d’entrée est l’accès au réseau local :

- Est-ce que les points d’accès **Wi-Fi** sont tous **indispensables et sécurisés** ?
- Des **prises filaires** sont-elles présentes à des **endroits accessibles à tous** ?
- Les **commutateurs de desserte** sont-ils tous bien configurés ?
- Suis-je en mesure d’avoir une **vision complète des équipements connectés** sur le SI ?

L’accès au réseau local

Un attaquant ayant un accès physique à proximité de vos locaux pourra essayer de :

- **Se connecter au réseau Wi-Fi** :
  - Soit car **le mot de passe WPA2 est faible** …
  - … ou bien via un peu de **filoutage** (ex : création d’un *fake AP*).
  - Contre-mesures : **limiter** les accès Wi-Fi, **durcir la configuration** des *AP*, **monitorer** les *AP* Wi-Fi, …
  - Outils : aircrack-ng, WiFi Pineapple, …

- **Se connecter à une prise filaire** :
  - Pour l’utiliser directement avec son **ordinateur portable** …
  - … ou bien déposer un **élément qui peut être dirigé depuis l’extérieur** (ex : *raspberry pi* avec une clé 3G).
  - Contre-mesures : **limiter** les prises filaires, authentifier les accès via **802.1X**, **monitorer** les équipements présents sur le SI, …

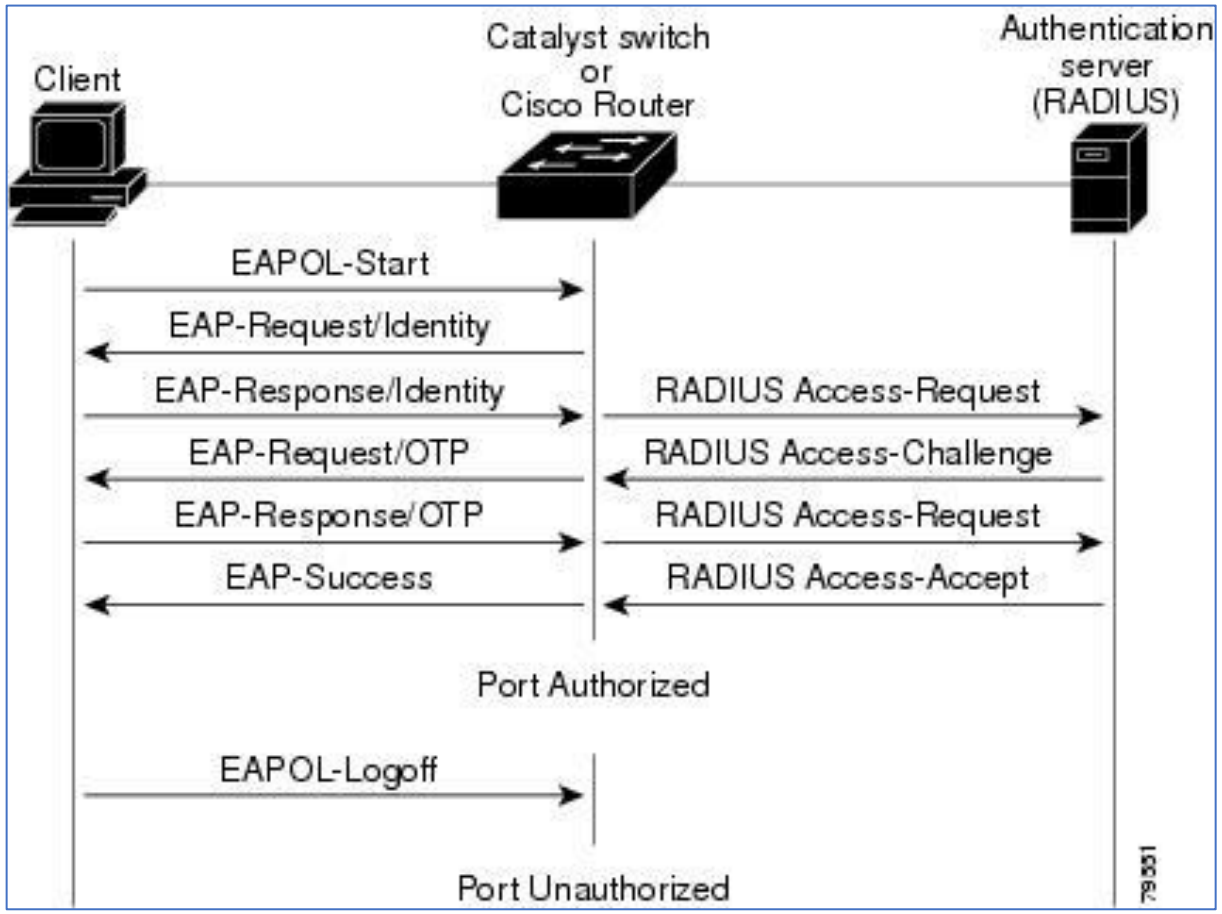
L’accès au réseau local

[1/2] Qu’est-ce que le **802.1X** ?

- Standard permettant de **contrôler l’accès aux ports**.
- Basé sur le protocole **E**xtensible **A**uthentication **P**rotocol (**EAP**).
- L’équipement contacte un **serveur d’authentification** (ex : **RADIUS**, **TACACS**, …).
- Divers **mécanismes d’authentification** : mot de passe, certificat, …
- Nécessite de configurer les **commutateurs** / **AP WiFi** mais aussi les **clients**.

L’accès au réseau local

[2/2] Qu’est-ce que le **802.1X** ?



UFA47

L’accès au réseau local

Une fois connecté au réseau local, un attaquant dispose de divers moyens pour étendre ses droits de manière latérale :

- **Attaque de type MITM :**
  - Via le classique « ***arp spoofing*** » …
  - … ou bien via d’autres techniques telles que : ***rogue DHCP***, ***rogue WPAD***, …
  - Contre-mesures : ***port security***, ***dynamic ARP inspection***, ***DHCP snooping***, **PVLAN**, …
  - Outils : ***dsniff***, ***Responder***, …

- **Ecoute passive** :
  - Récupérer des informations sur la **topologie réseau** : *Cisco Discovery Protocol* (CDP), *Link Layer Discovery Protocol* (LLDP), …
  - *Flooder* le commutateur pour **remplir la table *CAM***.
  - Contre-mesures : **limiter** les protocoles « verbeux » sur les dessertes clientes, ***port security***, …

L’accès au réseau local

Qu’est-ce que le « DHCP snooping » ?

- Permet de **contrer** le risque d’utilisation d’un **serveur DHCP malicieux**.
- **Autorise** seulement **un ou plusieurs ports** à émettre des réponses DHCP.
- Garde en mémoire l’historique IP / MAC / PORT.

Que-ce que le « ARP inspection » ?

- Permet de **contrer** le risque d’utilisation de **requêtes ARP malicieuses**.
- Se base sur l’**historique** du « **DHCP snooping** ».
- Si la requête ARP **ne correspond pas à une entrée**, elle n’est **pas renvoyée**.

Qu’est-ce que le « private VLAN » ?

- Permet de **bloquer le trafic** de niveau 2 de **client à client**.
- Réduction du **domaine de diffusion**.

L’accès au réseau local

Quelques commandes Cisco pour se protéger :

```
Switch(config)# ip dhcp snooping
Switch(config-if)# ip dhcp snooping trust
Switch# show ip dhcp snooping binding
```

```
Switch(config)# ip arp inspection vlan
Switch(config-if)# ip arp inspection trust
Switch# show ip arp inspection statistics
```

```
Switch(config-vlan)# private-vlan isolated
Switch(config-if)# switchport mode private-vlan promiscuous
Switch(config-if)# switchport private-vlan host-association X
```

L’accès au réseau local

[1/4] Cas pratique d’une attaque sur un réseau local : LLMNR+WPAD

Certaines attaques peuvent être mitigées par la mise en place de **PVLAN**.

C’est, par exemple, le cas de l’attaque **LLMNR/WPAD** :

- L’attaquant doit être présent sur le **même réseau que la cible**,
- La cible doit utiliser **Internet Explorer** avec les paramètres par défaut,
- L’attaquant va pouvoir prendre le **contrôle du navigateur** en injectant une **configuration proxy** malicieuse.

L’accès au réseau local

[2/4] Cas pratique d’une attaque sur un réseau local : LLMNR+WPAD

Pour obtenir « **automatiquement** » cette **configuration**, Windows réalise les actions suivantes :

1) Tente de **résoudre le nom d’hôte « WPAD »** via le protocole **LLMNR**,

2) Si une réponse est fournie, Windows va tenter de télécharger un **fichier de configuration PAC** à l’adresse **http://<ip>/wpad.dat**,

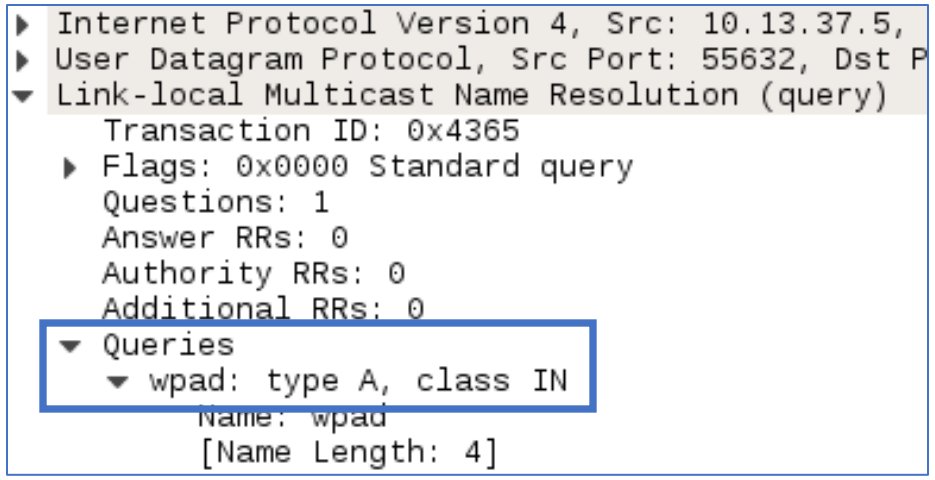
3) Ce fichier de configuration respecte un **format spécifique** et permet de **contrôler les paramètres proxy du navigateur**.

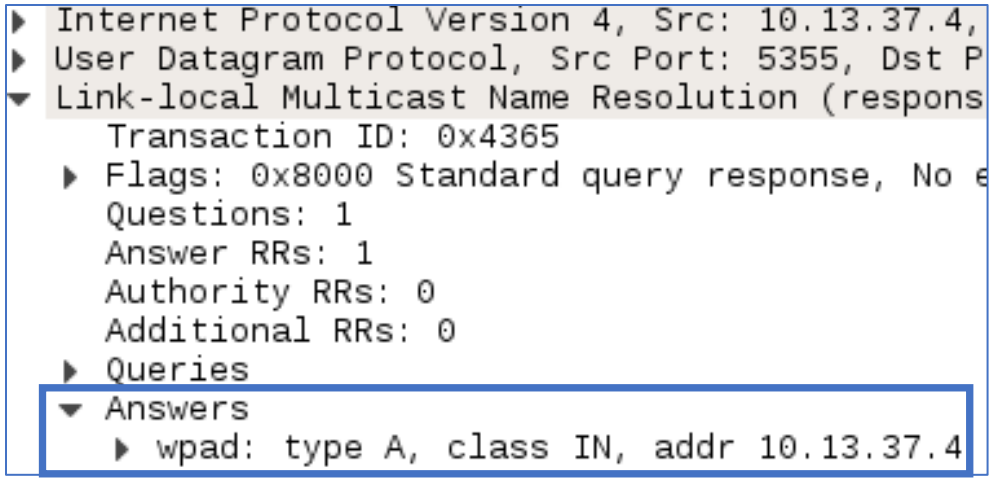
L’accès au réseau local

[3/4] Cas pratique d’une attaque sur un réseau local : LLMNR+WPAD

La vulnérabilité réside dans le fait que le protocole LLMNR (**Link-Local Multicast Name Resolution**) est un équivalent de DNS mais au **niveau 2**. C’est-à-dire que tout le monde peut répondre aux requêtes effectuées.

Voici un exemple ou 10.13.37.5 fait une requête LLMNR pour l’hôte « wpad » et 10.13.37.4 répond avec sa propre adresse IP :



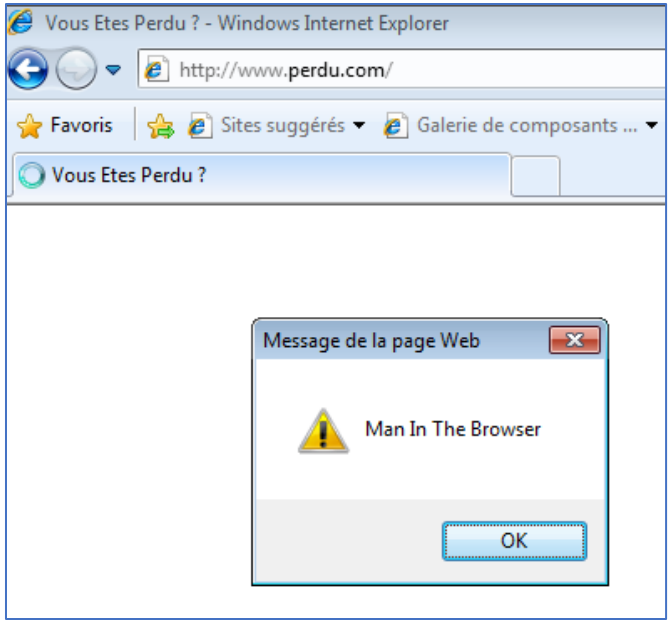


UFA47

L’accès au réseau local

[4/4] Cas pratique d’une attaque sur un réseau local : LLMNR+WPAD

Il est ensuite possible pour un attaquant de fournir un script PAC malicieux qui va modifier les paramètres du proxy de la cible. Ainsi il sera possible de lire et modifier les requêtes effectuées par le navigateur :



UFA47

L’accès au réseau local

Quizz time !

Quelle technologie me permet de contrôler l’accès aux prises filaires et AP WiFi ?

- 802.1X

Mes prises filaires sont dans un bunker ultra sécurisé, que puis-je faire de plus ?

- Est-ce que ça vaut le coup de mettre du 802.1X ?
- Nous avons déjà une mesure organisationnelle (le bunker) !

Comment mitiger le risque lié à la vulnérabilité LLMNR/WPAD ?

- Configurer mon parc Windows pour ne pas faire de la découverte sur le réseau,
- Configurer mes commutateurs pour faire du PVLAN.

Quels sont d’autres risques et contre-mesures associées sur le réseau local ?

- Rogue DHCP : DHCP snooping
- ARP spoofing : ARP inspection

L’accès aux services

UFA47

L’accès aux services

Comment savoir si un port TCP est ouvert ?

- On envoie un **SYN** vers la cible (IP + PORT).
- Si le serveur répond **SYN/ACK** : le port est **ouvert**.
- S’il répond **RST** : le port est **fermé**.
- S’il ne répond **rien** : il y a un **filtrage** (intermédiaire ou sur l’équipement).

Comment savoir si un port UDP est ouvert ?

- On tente d’envoyer des **données** vers la cible.
- Si on reçoit un **ICMP « *port unreachable* »** : le port est **fermé**.
- Si on ne reçoit **rien** : il y a peut être un **filtrage** …
- … mais contrairement à TCP, UDP est en mode **déconnecté**, …
- … il faut donc envoyer des **données spécifiques au service contacté** pour avoir une **réponse** ! Exemple : DNS, NTP, …

L’accès aux services

Afin de **corrompre** un serveur ou équipement, un attaquant peut utiliser des **vulnérabilités présentes sur des applicatifs** dont certains sont **accessibles depuis le réseau**.

Pour y arriver, il peut réaliser ce qu’on appelle un **scan de port** dont l’objectif est d’avoir **la liste des services hébergés par la machine**.

L’outil de référence est **nmap** : il est **gratuit**, disponible sur la **plupart des OS**, **mis à jour de manière régulière** depuis 1997, …

Son utilisation en ***CLI*** est relativement simple et il y a même une ***UI*** appelée **Zenmap**.

L’accès aux services

Scan de ports (top 1000) TCP en mode SYN (le ACK final n’est jamais renvoyé) :

```
root@debian:/opt# nmap -sS 10.13.37.5

Starting Nmap 7.40 ( https://nmap.org ) at 2018-09-04 08:37 EDT
Nmap scan report for 10.13.37.5
Host is up (0.00057s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
5357/tcp open  wsdapi
MAC Address: 08:00:27:FE:F1:57 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 9.33 seconds
root@debian:/opt#
```

L’accès aux services

Scan de ports (top 10) UDP :

```
root@debian:/opt# nmap -sU --top-ports 10 10.13.37.5

Starting Nmap 7.40 ( https://nmap.org ) at 2018-09-04 08:41 EDT
Nmap scan report for 10.13.37.5
Host is up (0.00057s latency).
PORT     STATE         SERVICE
53/udp   open|filtered domain
67/udp   open|filtered dhcps
123/udp  open|filtered ntp
135/udp  open|filtered msrpc
137/udp  open          netbios-ns
138/udp  open|filtered netbios-dgm
161/udp  open|filtered snmp
445/udp  open|filtered microsoft-ds
631/udp  open|filtered ipp
1434/udp open|filtered ms-sql-m
MAC Address: 08:00:27:FE:F1:57 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.64 seconds
root@debian:/opt#
```

UFA47

L’accès aux services

Prise d’empreinte (*fingerprinting*) d’un système :

```
root@debian:/opt# nmap -AT4 10.13.37.5

Starting Nmap 7.40 ( https://nmap.org ) at 2018-09-04 08:45 EDT
Nmap scan report for 10.13.37.5
Host is up (0.00073s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE VERSION
5357/tcp open  http    Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Service Unavailable
MAC Address: 08:00:27:FE:F1:57 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Microsoft Windows 7|8|8.1|Vista|2008
OS CPE: cpe:/o:microsoft:windows_7::-:professional cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows_8.1:r1
cpe:/o:microsoft:windows_vista::- cpe:/o:microsoft:windows_vista::sp1 cpe:/o:microsoft:windows_server_2008::sp1
OS details: Microsoft Windows 7 Professional or Windows 8, Microsoft Windows 8.1 R1, Microsoft Windows Vista SP0 or SP1, Windows
Server 2008 SP1, or Windows 7, Microsoft Windows Vista SP2, Windows 7 SP1, or Windows Server 2008
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

TRACEROUTE
HOP RTT     ADDRESS
1   0.73 ms 10.13.37.5

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.14 seconds
root@debian:/opt#
```

UFA47

L’accès aux services

Afin de **réduire au maximum la surface d’attaque**, il est nécessaire de mettre en place un **filtrage au niveau réseau** et ce, notamment, sur les équipements les plus importants.

Le filtrage peut se faire de différente manière :

- Utiliser des **équipements dédiés** à cette fonction (ex : appliances),
- Effectuer le **filtrage** directement **sur les machines**,
- Utiliser la fonction **filtrage** sur des **équipements existants** (ex : routeurs).

Nous allons ici étudier le filtrage au travers de **netfilter** :

- Partie intégrante du **noyau Linux**,
- **Complet** : NAT, suivi des connexions, filtrage, …
- **Gratuit**,
- Régulièrement **mis à jour**.

L’accès aux services

Un pare-feu est composé de **règles définies par l’administrateur**. Ces règles possèdent **plusieurs attributs** dont voici les principaux :

- Une adresse source,
- Une adresse destination,
- Un protocole,
- Un port source,
- Un port destination,
- Une action.

Exemples de règles :

@ SRC	@ DST	PROTO	PORT SRC	PORT DST	ACTION
10.42.13.37	10.42.1.1	TCP	*	22	DROP
0.0.0.0/0	1.3.3.7	UDP	*	53	ACCEPT

L’accès aux services

Lorsqu’un paquet arrive, les règles du pare-feu sont **parcourues l’une après l’autre dans un ordre chronologie**. Si le paquet « ***match*** » une règle alors **l’action associée est réalisée**.

En plus des règles, un pare-feu contient une **politique par défaut** à appliquer si **aucune des règles ne « *match* »** : **accepter** le paquet ou le **refuser**.

Appliquer **une politique de refus sera toujours préférable**. Ainsi, il est possible de faire ce que l’on appelle une **liste blanche** : accepter seulement les flux autorisés. À l’inverse de la **liste noire** qui autorise tout sauf certains flux.

L’accès aux services

La **gestion** du pare-feu **netfilter** se fait en ligne de commande via l’outil **iptables**.

```
[…]

DESCRIPTION
       Iptables  and  ip6tables  are used to set up, maintain, and inspect the tables of IPv4 and
IPv6 packet filter rules in the Linux kernel.  Several different tables may be defined.
       Each table contains a number of built-in chains and may also contain user-defined chains.

       Each chain is a list of rules which can match a set of packets.  Each rule specifies what
to do with a packet that matches.  This is called a `target', which may be a jump  to  a
       user-defined chain in the same table.

[…]
```

UFA47

L’accès aux services

iptables intervient sur **différentes tables**, qui ont chacune un **objectif différent** dont voici les principales :

- **filter** : gestion des règles « classiques » pour le **filtrage** (table par défaut),
- **nat** : pour faire du « ***network address translation*** »,
- **mangle** : permet de **modifier** les paquets.

Ces **tables** sont **composées de chaînes** qui possèdent chacune un **champ d’action différent,** des **règles** à appliquer et **une politique par défaut**. Voici les chaînes de la table **filter** :

- **INPUT** : règles à appliquer lorsque le pare-feu **reçoit un paquet qui lui est destiné**,
- **OUTPUT** : règles à appliquer lorsque le pare-feu **émet un paquet**,
- **FORWARD** : règles à appliquer lorsque le pare-feu doit **retransmettre un paquet**.

Il est aussi possible de **créer ses propres chaînes** pour organiser de manière plus claire notre ***ruleset***.

L’accès aux services

Obtenir la liste des règles de la table filter :

```
root@debian:/opt# iptables -t filter -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@debian:/opt#
```

UFA47

L’accès aux services

Définir la politique par défaut pour la chaîne FORWARD :

```
root@debian:/opt# iptables -t filter -P FORWARD DROP
root@debian:/opt# iptables -t filter -nvL FORWARD
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@debian:/opt#
```

UFA47

L’accès aux services

Création d’une chaîne personnalisée nommée « MACHAINE » :

```
root@debian:/opt# iptables -t filter -N MACHAINE
root@debian:/opt# iptables -t filter -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain MACHAINE (0 references)
 pkts bytes target     prot opt in     out     source               destination
root@debian:/opt#
```

UFA47

L’accès aux services

Ajout de la chaîne MACHAINE aux règles de la chaîne FORWARD :

```
root@debian:/opt# iptables -t filter -A FORWARD -j MACHAINE
root@debian:/opt# iptables -t filter -nvL FORWARD
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MACHAINE   all  --  *      *       0.0.0.0/0            0.0.0.0/0
root@debian:/opt#
```

L’accès aux services

Suppression des règles :

```
root@debian:/opt# iptables -t filter -F
root@debian:/opt# iptables -t filter -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain MACHAINE (0 references)
 pkts bytes target     prot opt in     out     source               destination
root@debian:/opt#
```

L’accès aux services

Suppression de la chaîne MACHAINE :

```
root@debian:/opt# iptables -t filter -X MACHAINE
root@debian:/opt# iptables –t filter -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@debian:/opt#
```

L’accès aux services

iptables permet de définir des **critères précis pour qu’une règle match** dont voici les principaux :

- -p *protocole* : tcp, udp, icmp, …
- -s *@src* : adresse IP source
- -d *@dst* : adresse IP destination
- -i *interface* : interface d’entrée
- -o *interface* : interface de sortie
- --sport *port* : port source
- --dport *port* : port destination
- -j *action* : action à réaliser

L’accès aux services

Ajouter une règle :

```
root@debian:/opt# iptables -t filter -A INPUT -s 10.13.37.1 –d 10.13.37.42 -j ACCEPT
root@debian:/opt# iptables -t filter -nvL INPUT
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       10.13.37.1           10.13.37.42
root@debian:/opt#
```

Retirer la règle que nous venons d’ajouter :

```
root@debian:/opt# iptables -t filter -D INPUT -s 10.13.37.1 –d 10.13.37.42 -j ACCEPT
root@debian:/opt# iptables -t filter -nvL INPUT
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@debian:/opt#
```

UFA47

L’accès aux services

Exemple de règles avec des critères sur les ports TCP :

```
root@debian:/opt# iptables -t filter -A FORWARD -i int0 -p tcp --dport 80 -j ACCEPT
root@debian:/opt# iptables -t filter -A FORWARD -o int1 -p tcp --sport 80 -j ACCEPT
root@debian:/opt# iptables -t filter -nvL FORWARD
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     tcp  --  int0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     tcp  --  *      int1    0.0.0.0/0            0.0.0.0/0           tcp spt:80
root@debian:/opt#
```

L’accès aux services

Exemple de règles avec des critères sur les ports UDP :

```
root@debian:/opt# iptables -t filter -A FORWARD -i int0 -p udp --dport 53 -j ACCEPT
root@debian:/opt# iptables -t filter -A FORWARD -o int1 -p udp --sport 53 -j ACCEPT
root@debian:/opt# iptables -t filter -nvL FORWARD
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  int0   *       0.0.0.0/0            0.0.0.0/0           udp dpt:53
    0     0 ACCEPT     udp  --  *      int1    0.0.0.0/0            0.0.0.0/0           udp spt:53
root@debian:/opt#
```

L’accès aux services

Que pouvons dire des règles suivantes (*policy* DROP) ?

iptables –A FORWARD –i localnet0 –o localnet1 –p tcp –dport 443 –j ACCEPT

iptables –A FORWARD –i localnet1 –o localnet0 –p tcp –sport 443 –j ACCEPT

- Tout ce qui vient du réseau local à destination d’Internet sur le port TCP/443 sera forwardé.
- Tout ce qui vient d’Internet à destination du réseau local depuis le port source TCP/443 sera forwardé.
- Problème #1 : les paquets « **bogués** » TCP (ex : flags SYN+FIN) sont autorisés à arriver et sortir vers les réseaux.
- Problème #2 : les paquets avec comme port source TCP/443 peuvent contacter **l’ensemble des serveurs de l’autre réseau**.

L’accès aux services

Interdire les paquets TCP avec les flags SYN et RST positionnés :

```
root@debian:/opt# iptables -t filter -A INPUT -p tcp --tcp-flags ALL SYN,RST -j DROP
root@debian:/opt# iptables -t filter -nvL INPUT
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp
flags:0x3F/0x06
root@debian:/opt#
```

UFA47

L’accès aux services

La seconde problématique peut être traitée en utilisant la fonctionnalité de **gestion des états** qui est intégrée à netfilter dont voici les principaux :

- **NEW** : il s’agit du **premier paquet** dans une communication.
- **ESTABLISHED** : il s’agit d’un paquet qui intervient sur une **communication déjà existante**.
- **RELATED** : il s’agit d’un paquet **en relation** avec une communication déjà existante (exemple : erreur ICMP).

L’accès aux services

Autoriser le surf Internet à « traverser » notre pare-feu :

```
root@debian:/opt# iptables -t filter -A FORWARD -p tcp --syn --dport 80 -m state --state NEW -j ACCEPT
root@debian:/opt# iptables -t filter -A FORWARD -p tcp --syn --dport 443 -m state --state NEW -j ACCEPT
root@debian:/opt# iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
root@debian:/opt# iptables -nvL FORWARD
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
flags:0x17/0x02 state NEW
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
flags:0x17/0x02 state NEW
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
root@debian:/opt#
```

UFA47

L’accès aux services

Autoriser seulement 10.13.37.1 à accéder à notre SSH :

```
root@debian:/opt# iptables -t filter -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
root@debian:/opt# iptables -t filter -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
root@debian:/opt# iptables -t filter -A INPUT -s 10.13.37.1 -p tcp --syn --dport 22 -m state --state
NEW -j ACCEPT
root@debian:/opt# iptables -t filter -nvL INPUT ; iptables -t filter -nvL OUTPUT
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       10.13.37.1           0.0.0.0/0           tcp dpt:22
flags:0x17/0x02 state NEW
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
root@debian:/opt#
```

UFA47

Analyse de l’entête IP

Quizz time !

Quelles sont les 3 chaînes de la table filter de netfilter ainsi que leur rôle ?

- INPUT : paquets à destination de la machine,
- OUTPUT : paquets émis par la machine,
- FORWARD : paquets qui « traversent » la machine.

Quels sont les 3 états principaux du module state de netfilter ?

- NEW : premier paquet,
- ESTABLISHED : paquet dans une communication existante,
- RELATED : paquet en relation avec une communication (ex : ICMP)

À quoi je dois penser à part les règles ?

- À la politique par défaut (préférer la liste blanche si possible).
- Attention cependant à ne pas se bloquer (ex : policy DROP avant ACCEPT pour le SSH).

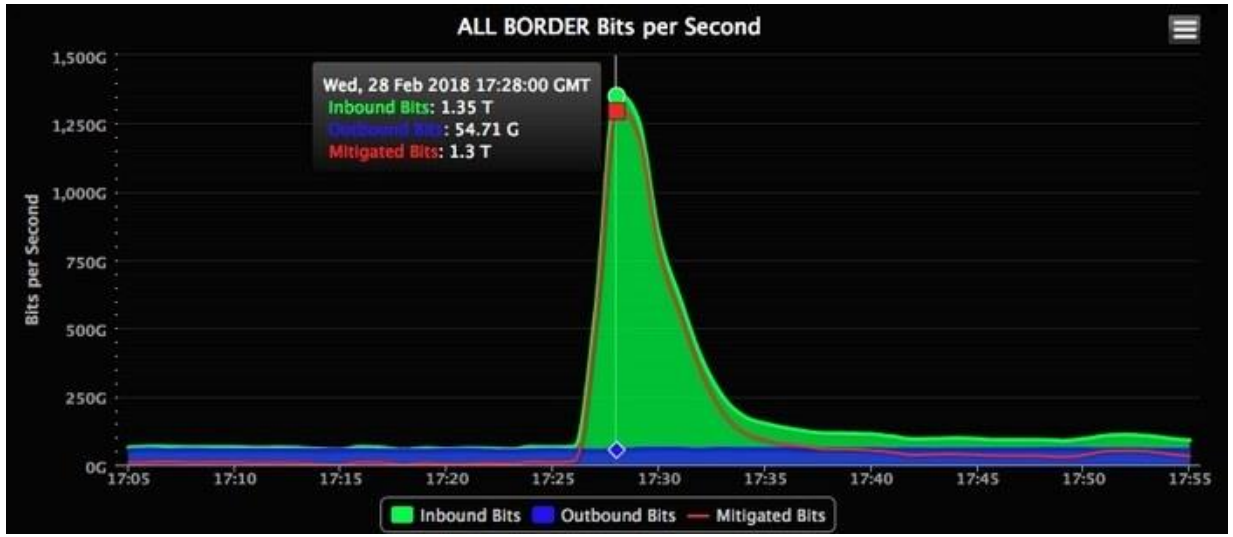
Le déni de service

UFA47

Le déni de service

Le **déni de service** consiste à **rendre indisponible l'accès à un système informatique**. Le système en question peut être un **réseau**, un **serveur** ou un **applicatif web** par exemple.

Le terme « ***Denial of Service*** » (DoS) est utilisé lorsqu'**une seule machine** arrive à rendre la cible indisponible. Dans le cas où **plusieurs machines** sont utilisés nous parlons de « ***Distributed Denial of Service*** » (DDoS).



1.3 Tbps contre GitHub début 2018

UFA47

Le déni de service

Différentes techniques sont communément utilisées par les pirates :

- **Utilisation d'un botnet** : saturer la bande passante de la cible par « une puissance de frappe » (BP botnet > BP cible).
- **L'attaque par amplification** : utiliser des serveurs verbeux dans leur réponse couplé avec du spoofing d'IP.
- **Vulnérabilité « système »** : envoi massif de paquets SYN sans réellement valider la connexion (SYN flood), garder beaucoup de connexions ouvertes (Slowloris), …
- **Vulnérabilité « applicative »** : envoi de requêtes qui met en péril l'applicatif qui utilise le réseau (ex : espace disque, temps d'utilisation, mémoire, …).

Le déni de service

[1/2] Principe de fonctionnement d'une attaque DDoS par amplification

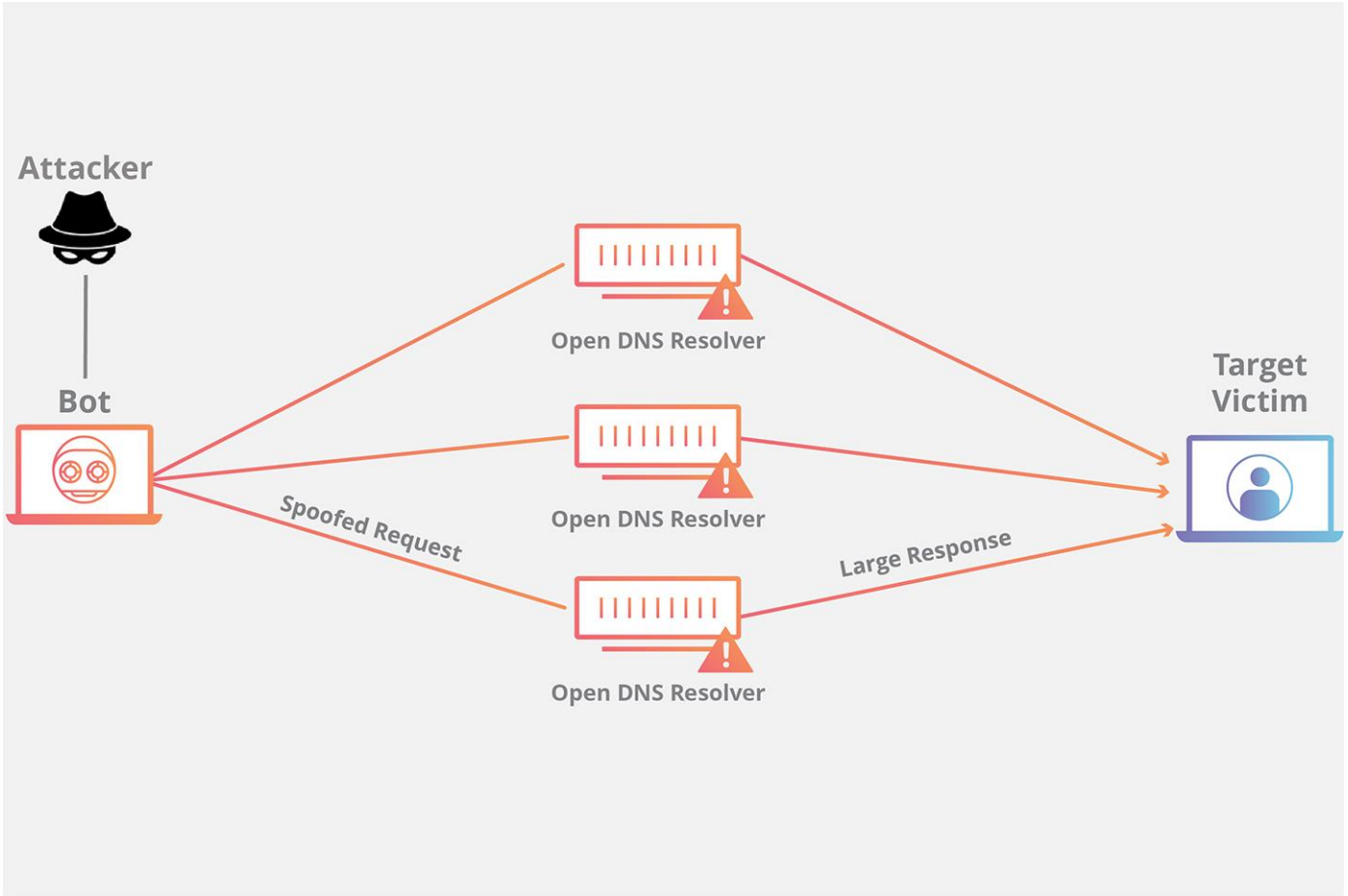
1. L'attaquant **forge des requêtes UDP malicieuses** avec comme critères :
   - IP source : adresse de la cible
   - IP destination : adresse d'un serveur tiers mal configuré
   - Port source : aléatoire
   - Port destination : port de l'applicatif mal configuré

2. L'applicatif reçoit la requête et va effectuer une **réponse à l'adresse IP source** du paquet. Or, sa réponse contient **plus de données que la requête initiale** du pirate (d'où le terme **amplification**).

3. La cible reçoit des paquets qui seront ignorés car inconnus dans la pile IP du système. Mais cela utilise tout de même de sa **bande passante** !

Le déni de service

[2/2] Principe de fonctionnement d'une attaque DDoS par amplification



UFA47

Le déni de service

Quelques contre-mesures pour ce prémunir des attaques par déni de service :

- Si le serveur est accessible depuis Internet et hébergé par un tiers, s'assurer que le fournisseur propose un **service anti-DDoS**.
- Si le serveur est accessible depuis Internet mais hébergé par vos soins, **définir et mettre en place une politique de filtrage** au plus proche du réseau Internet (pour éviter les dommages collatéraux sur le réseau local).
- **Limiter les requêtes** faites pour chaque **adresse IP** sur différents critères (taille, nombre par seconde, …).
- Mettre les systèmes à **l'état de l'art** : audit, mise à jour, configuration (ex : SYN cookie contre le SYN flood), …