

Intrusion Detection

Badis HAMMI

Cybersecurity threats



Cyberattacks

- Estimated \$6 trillion in damages by 2021
- Ransomware attack every 14 seconds
- Public administration organizations receive one malicious email per 302 emails
- Over 24,000 malicious mobile apps are blocked daily
- More than 21% of files aren't protected (6.2 billion files were analysed)
- 30% of phishing emails in the U.S. are opened
- 300 billion passwords worldwide by 2020
- In 2016, Adware affected 75% of organizations
- Average ransomware demand is \$1,077
- China have the most rate of malware infection in the world (Over 55% of China's computers are infected with malware).
- Only 10% of cybercrimes are reported in the U.S each year
- Companies take over 6 months to notice a data breach

Cybersecurity

- Cybersecurity
 - Vital issue for any company or institution
 - Development of cybersecurity
 - Today a real concern for the different actors of the economy: companies and operators
 - Complex field

Firewall



Badis HAMMI

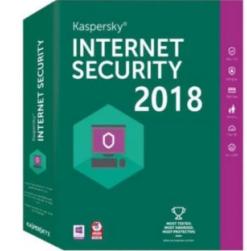
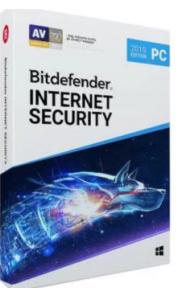
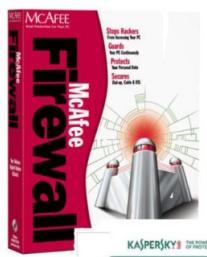
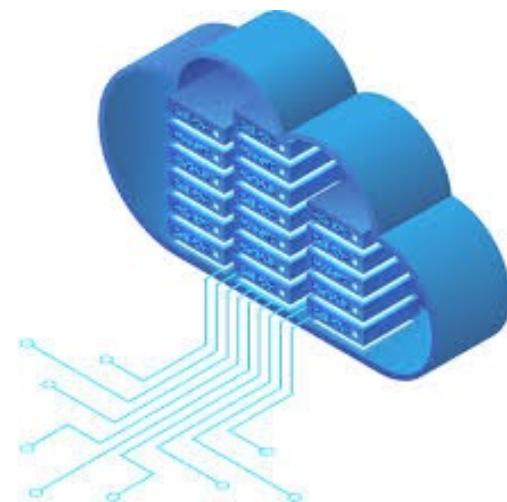
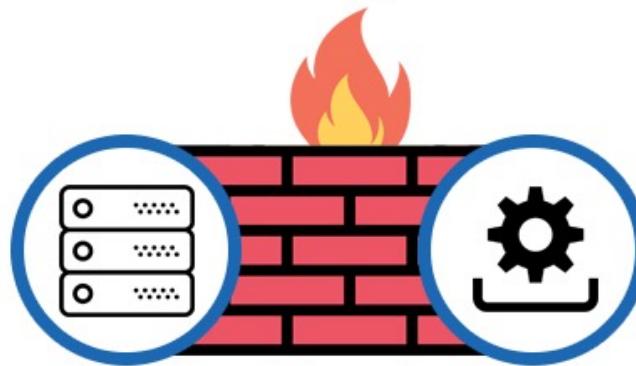
Firewall

- A firewall is a system that provides network security by filtering incoming and outgoing network traffic based on a set of user-defined rules
- These rules come from a security policy
- Rules or filters have different names depending on the publisher:
 - Access Control List or ACL (CISCO), Policy (Juniper Networks)
- The purpose of a firewall is to reduce or eliminate the occurrence of unwanted network communications while allowing all legitimate communication to flow freely.



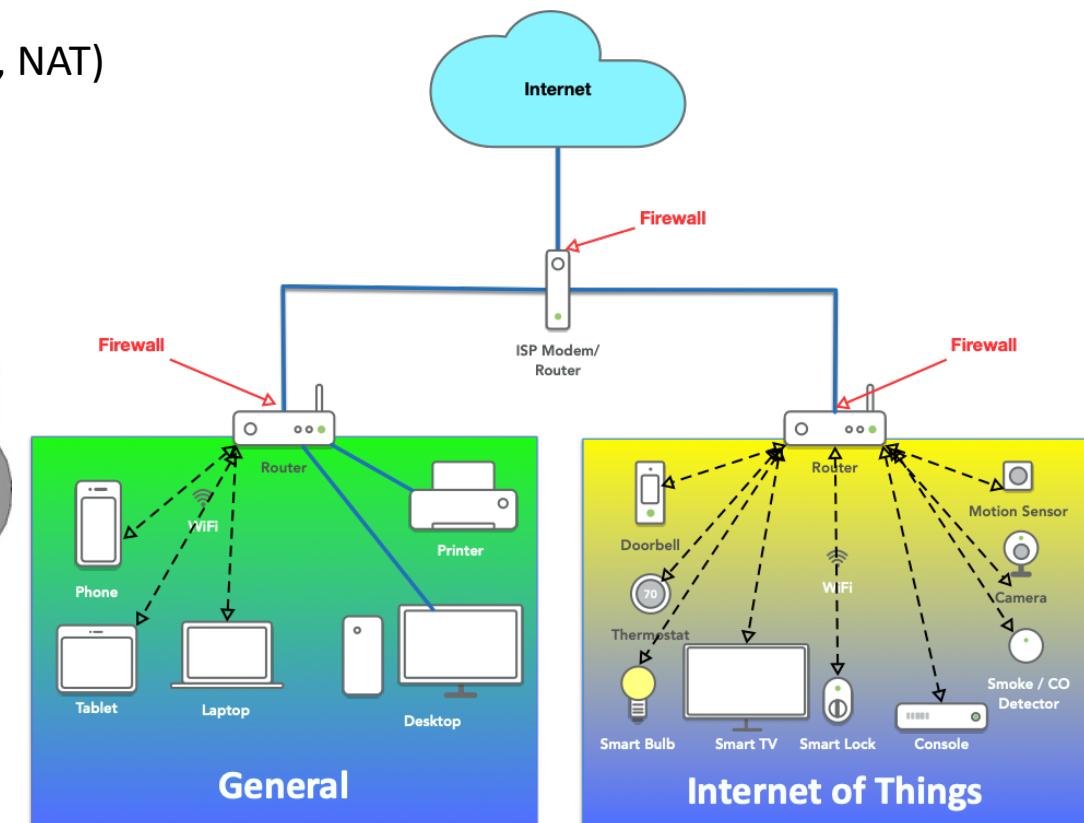
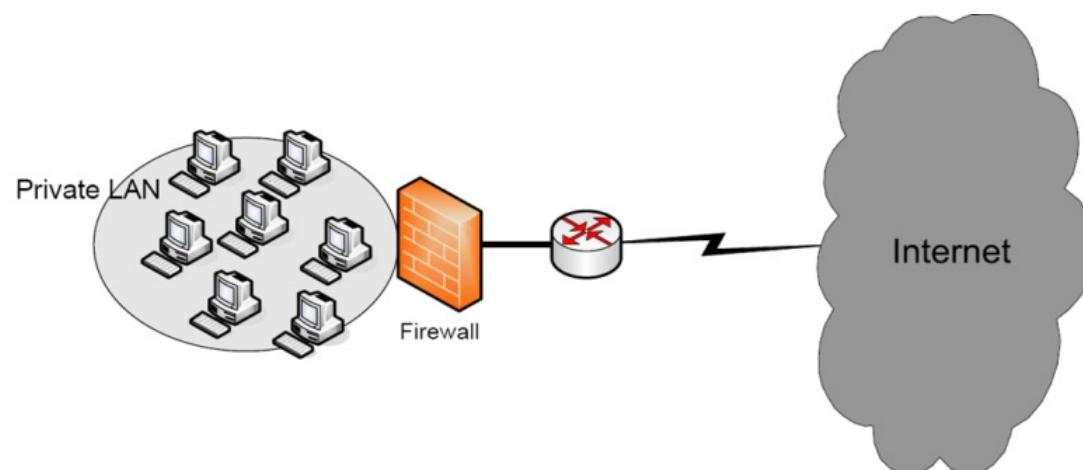
Firewall

Hardware Firewalls v/s Software Firewalls

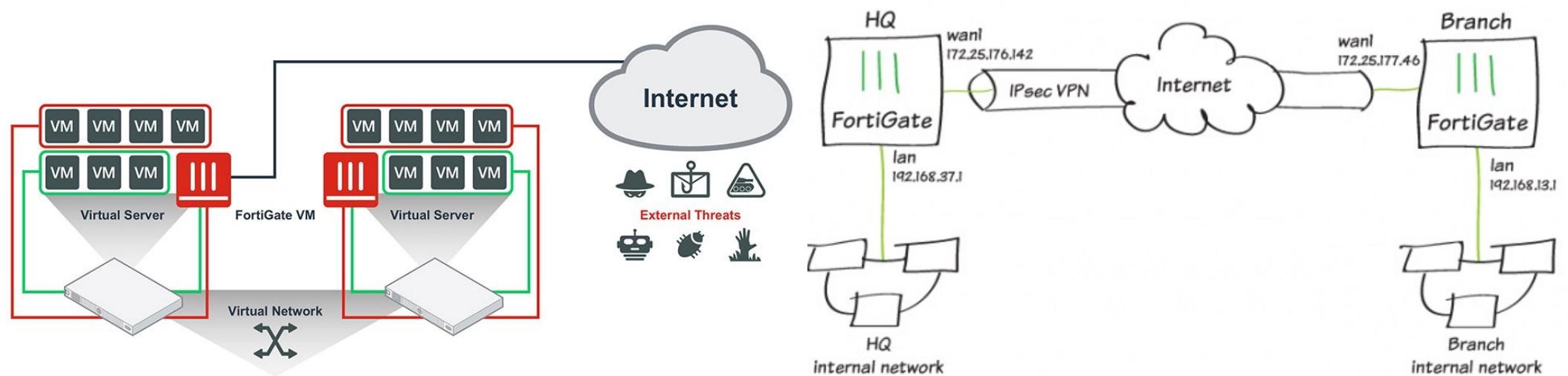


Firewall: Architecture

- A firewall can be placed
 - Between an Internal network (Intranet) and the global Internet network
 - Between internal networks for the partitioning
 - At a host level (Personal firewall)
- Possible management of complex networks (VPN, DMZ, NAT)



Firewall: Architecture



Firewall: Types

There are three basic types of network firewalls: packet filtering (stateless), stateful, and application layer

- **Packet filtering, or stateless**, firewalls work by inspecting individual packets in isolation. As such, they are unaware of connection state and can only allow or deny packets based on individual packet headers
- **Stateful** firewalls are able to determine the connection state of packets, which makes them much more flexible than stateless firewalls. They work by collecting related packets until the connection state can be determined before any firewall rules are applied to the traffic.
- **Application firewalls** go one step further by analyzing the data being transmitted, which allows network traffic to be matched against firewall rules that are specific to individual services or applications. These are also known as **proxy-based firewalls**.

Firewall: Rules

Network traffic that traverses a firewall is matched against rules to determine if it should be allowed through or not.

Suppose you have a server with this list of firewall rules that apply to incoming traffic:

1. **Accept** new and established incoming traffic to the public network interface on port 80 and 443 (HTTP and HTTPS web traffic)
 2. **Drop** incoming traffic from IP addresses of the non-technical employees in your office to port 22 (SSH)
 3. **Accept** new and established incoming traffic from the office IP range to the private network interface on port 22 (SSH)
-
- **Accept** means to allow the traffic through,
 - **reject** means to block the traffic but reply with an “**unreachable**” error
 - **drop** means to block the traffic and send **no reply**.

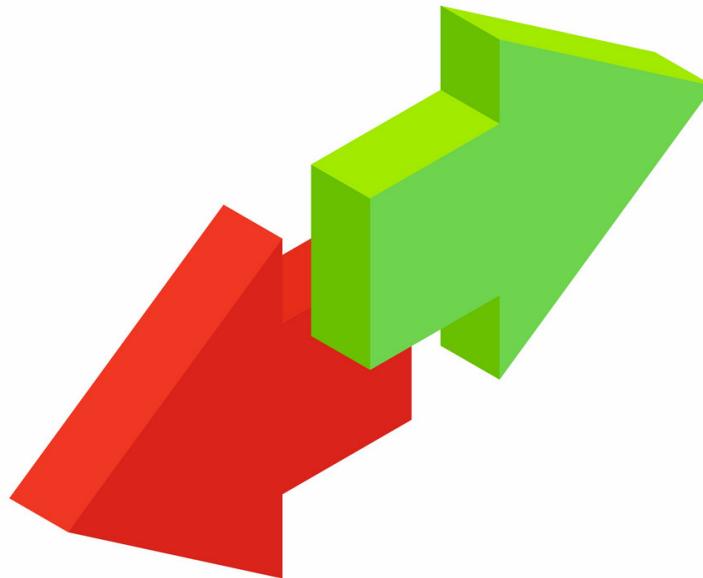
Firewall: Default Policy

It is typical for a chain of firewall rules to not explicitly cover every possible condition. For this reason, firewall chains must always have a default policy specified, which consists only of an action (accept, reject, or drop).



Firewall: Incoming and Outgoing Traffic

As network traffic, from the perspective of a server, can be either incoming or outgoing, a firewall maintains a distinct set of rules for either case. Traffic that originates elsewhere, incoming traffic, is treated differently than outgoing traffic that the server sends. It is typical for a **server to allow** most outgoing traffic because the server is usually, to itself, trustworthy. On the contrary, an enterprise's firewall/router mainly **rejects all** incoming traffic.



Firewall: Advantages

- Centralized security management
- Network traffic auditing capacity
 - Everything goes through the firewall
- Possibility of traceback
- Focus on one or more machines rather than all machines on the network
- No modification on client or server workstations

Firewall: Disadvantages

- Network bottleneck
- Network hotspot
 - Favorite target of hackers
- Syntax of rules specific for each provider
- Lack of standards at all levels
 - Architectures, rules, management, certification, interface
- Need
 - Total control of the protocols crossed (TCP, IP, RTP, RTSP, SIP, HTTP, FTP, H323, SQL, ...)
 - Understanding of how the firewall works (interface between the various filtering levels, address translation, etc.)

Firewall

A firewall does not provide:

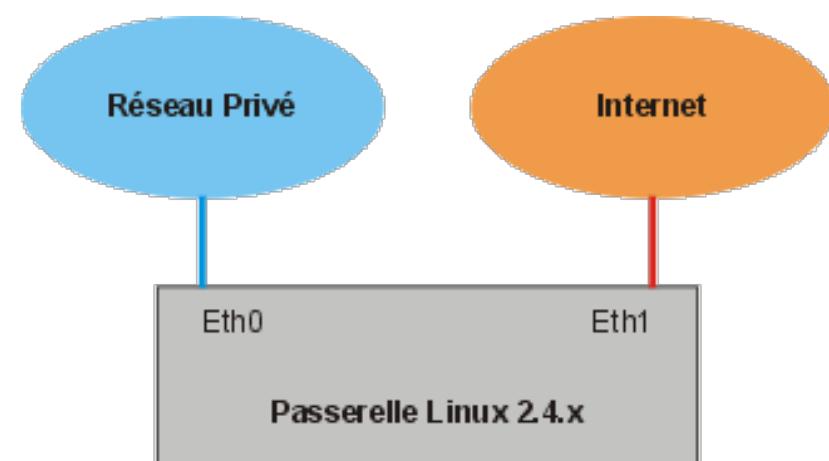
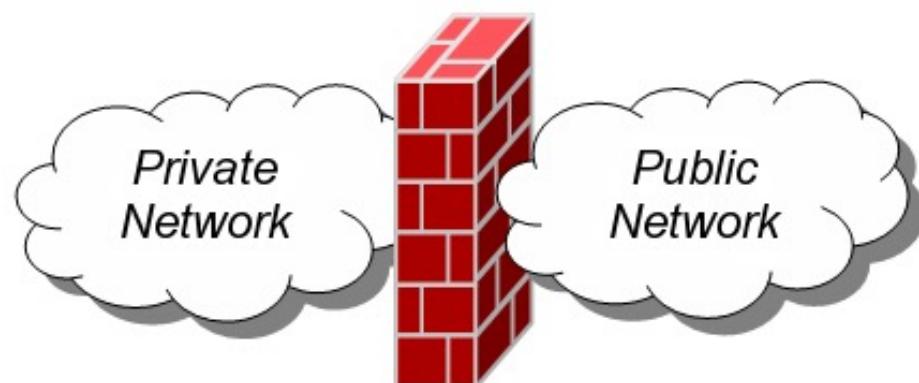
- data confidentiality
 - But we can implement a protocol between two firewalls to ensure encryption (VPN)
- data integrity
 - does not guarantee that the packet in transit is like the one that was transmitted
 - possibility of interfacing a firewall with another tool to check data integrity
 - no deep data inspection (DPI or WAF)
- authentication of data origin
 - ip spoofing
- does not protect traffic that does not pass through it
- does not protect against internal threats:
 - A firewall is little deployed and configured between internal networks

Firewall

- **Netfilter/Iptables** (Linux 2.4),
- PacketFilter (OpenBsd)
- IP-Filter (FreeBsd, NetBsd, HPUX, Solaris)
- Microsoft ISA Server
- Netscreen (Juniper Networks)
- **Fortinet**
- **PaloAlto**
- Netasq and Arkoon → **Stormshield**
- **Checkpoint** software (Firewall-1)
- Cisco (PIX, Centri Firewall)
- Raptor systems (Eagle)
- Bull soft (M >Wall)

Firewall: Netfilter

Netfilter is a framework provided by the Linux kernel that allows various networking-related operations to be implemented in the form of customized handlers. Netfilter offers various functions and operations for packet filtering, network address translation, and port translation, which provide the functionality required for directing packets through a network and prohibiting packets from reaching sensitive locations within a network.

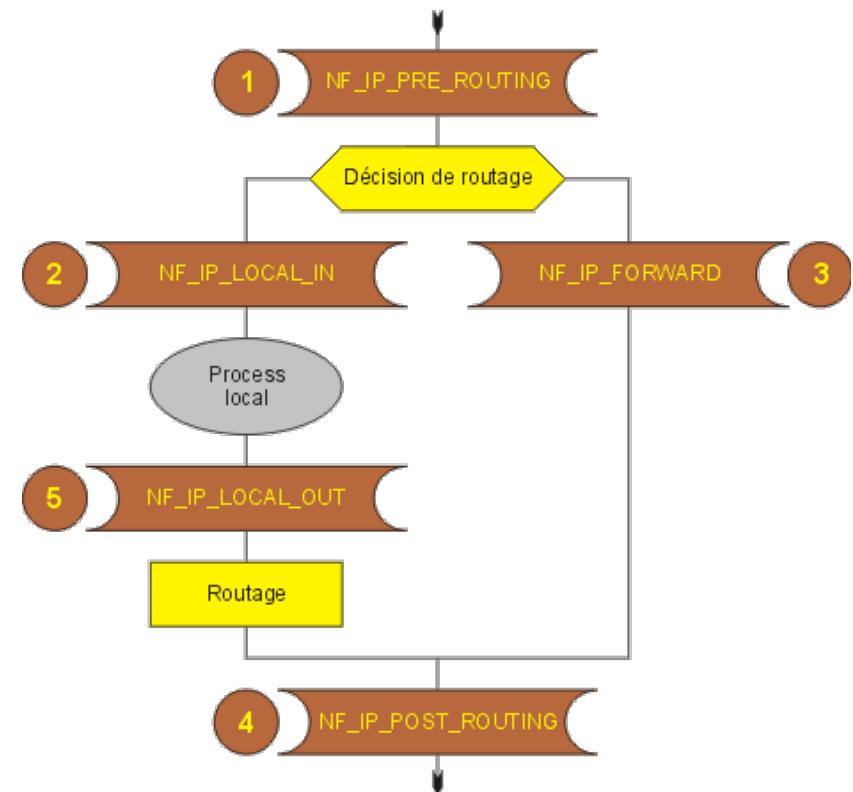
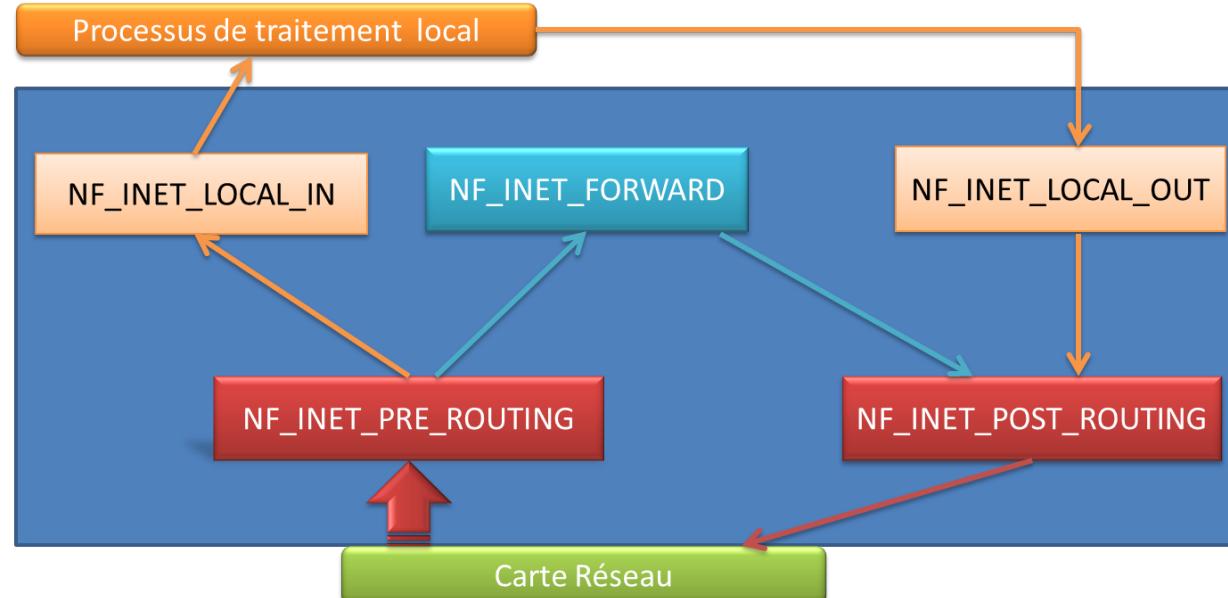


Firewall: Netfilter

Netfilter is a Linux kernel level Firewall

- Packets are processed at the kernel level by modules integrated into the kernel (Kernel space)
- The kernel provides hooks in the path of packets in the protocol stack
- Netfilter is the platform which manages these hooks by offering the possibility to modules and record additional processing functions on packets and to return the results in the form of specific actions
- A hook does not tell you how the package should be handled, but just where it will be processed.
- NF_IP_PRE_ROUTING
- NF_IP_LOCAL_IN
- NF_IP_FORWARD
- NF_IP_POSTROUTING
- NF_IP_LOCAL_OUT

Firewall: Netfilter



Firewall: Netfilter

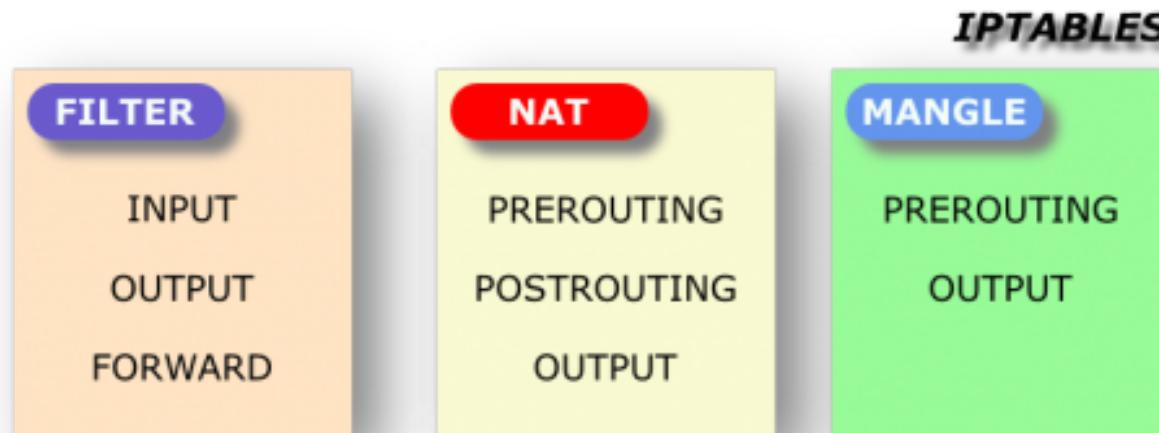
Through these five hooks, Netfilter will be able to:

1. Perform packet **filtering**, mainly to provide firewall functions. We can for example prohibit all packets coming from the Internet and addressing port 80 (HTTP) to pass.
2. Perform **NAT** (Network Address Translation) operations. These functions are particularly useful when you want to communicate all or part of a private network, mounted with private IP addresses (192.168.x.x for example) with the Internet.
3. To carry out operations of **marking** of the packets, to apply a special treatment to them. These features are particularly interesting on a corporate network gateway, a little less for a home network case.

Firewall: Netfilter

How Netfilter works ?

- Netfilter has an all-purpose command: IPtables.
- Iptables command will allow, among other things, to write chains of rules in tables.
- There are three tables in Netfilter which correspond to the three main functions seen above.



Firewall: Netfilter

- Each table contains a set of chains
- Chains are sets of rules written to each table
- These chains will make it possible to identify packets which correspond to certain criteria and match with certain patterns

TABLE 1

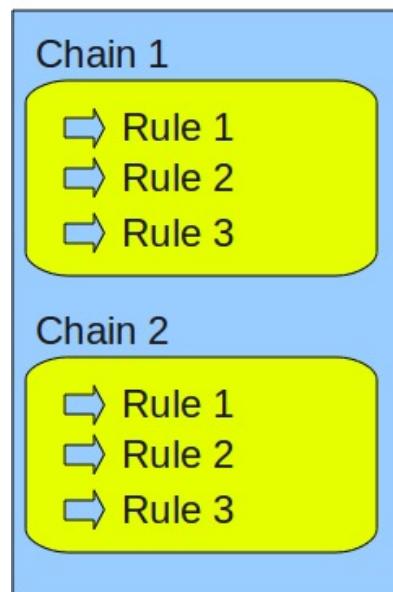
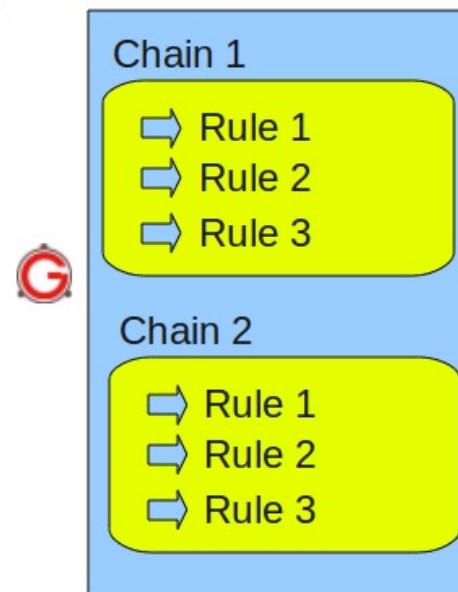


TABLE 2



Firewall: Netfilter

Filter table

This table will contain all the rules that will filter the packets. This table contains three chains:

1. INPUT chain:

This chain will decide the fate of packets entering locally on the host

2. OUTPUT chain:

Here, only packets sent by the local host will be filtered

3. FORWARD chain:

The packets which cross the host, following the established routes, will be filtered here

Firewall: Netfilter

NAT table

This table allows to perform all the necessary address translations

1. PREROUTING chain

Allows to translate the destination address. This method is interesting if you want to make the outside world believe, for example, that there is a WEB server on port 80 of the gateway, while it is hosted by a host on the private network, on the port 8080

2. POSTROUTING chain

It allows source address translation, like address masking, the classic method to connect a private network as an Internet client, with a single "official" IP address.

3. OUTPUT chain

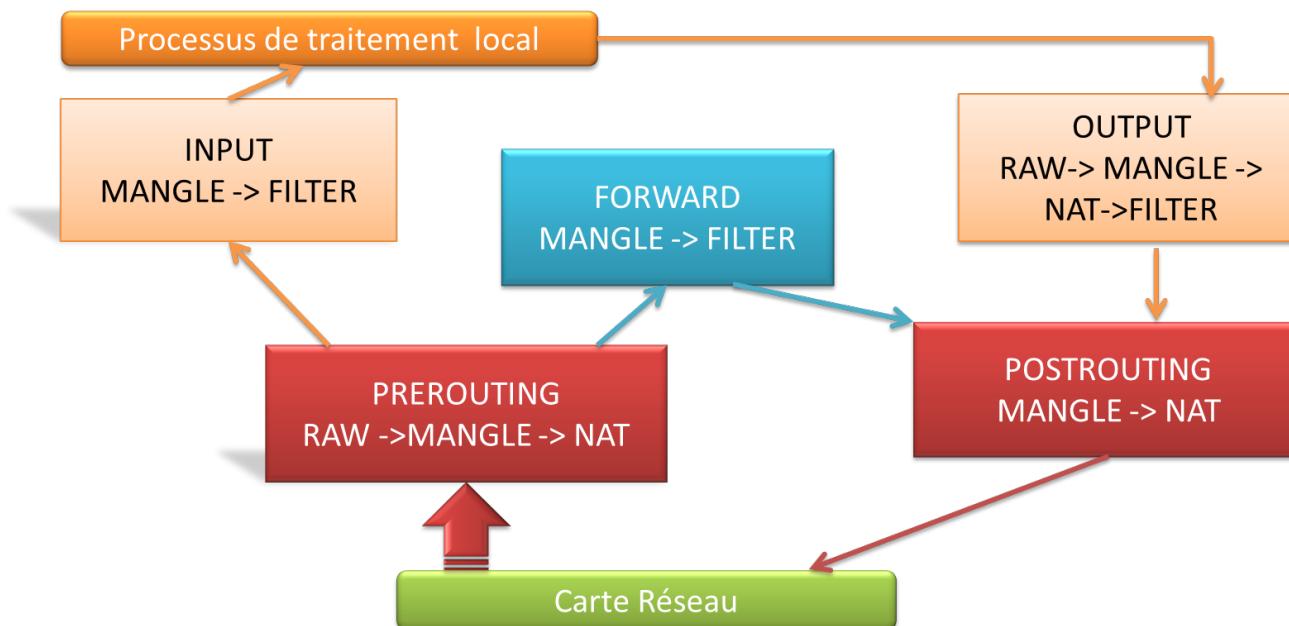
This will allow you to modify the destination of locally generated packets (by the gateway itself).

Firewall: Netfilter

MANGLE table

- This table allows the marking of incoming packets (**PREROUTING**) and locally generated packets (**OUTPUT**).
- Packet marking will allow specific processing of packets marked in the routing tables
- Since version 2.4.18 of the kernel, other tables have been added on all "hooks". We thus have at our disposal the additional tables **INPUT**, **POSTROUTING** and **FORWARD**.

Firewall: Netfilter



Firewall: Netfilter

Targets

Targets are sort of referrals that will direct packages that meet the criteria. The pre-built targets are:

- **ACCEPT**
 - Packets that meet the criteria are accepted, they continue their way in the stack,
- **DROP**
 - Packets that meet the criteria are rejected, we forget them, we don't even send an ICMP message. like a black hole.
- **LOG**
 - It is a particular target which makes it possible to trace by means of syslog the packets which satisfy the criteria.

Depending on the context, other targets become accessible, such as **REJECT** (similar to **DROP**, but it sends an ICMP error message to the source of the rejected package), **RETURN**, **REDIRECT**, **SNAT**, **DNAT**, **MASQUERADE** ...

Firewall: Netfilter

Example:

- By default, all incoming packets are dropped (DROP)
- Packets that enter through port 80 on the eth0 interface are accepted
- Packets that come out of port 80 on the eth0 interface are accepted
- etc.



Badis HAMMI

Firewall: Netfilter

Conntrack

- Connection tracking is an essential concept in Netfilter. It is a kind of artificial intelligence which makes it possible to establish cause and effect links between packets which pass through the stack.
- The principle of connection tracking makes it possible to create a "statefull firewall", that is to say that it will react intelligently to a given connection, depending on its state.

Firewall: Netfilter

No.	Time	Source	Destination	Protocol	Info
10	10.181832	192.168.0.10	212.27.35.1	TCP	4252 > http [SYN]
11	10.204707	212.27.35.1	192.168.0.10	TCP	http > 4252 [SYN, ACK]
12	10.204848	192.168.0.10	212.27.35.1	TCP	4252 > http [ACK]
13	10.205333	192.168.0.10	212.27.35.1	HTTP	GET / HTTP/1.1

- Frame n° 10:
 - The client [192.168.0.10] addresses the server [212.27.35.1] on the http port (80).
 - It waits for a response on port 4252.
- Frame n° 11:
 - The server responds to the client on the requested port (4252)

Firewall: Netfilter

The same thing, but more detailed, which gives the opportunity to see all the "flags" usable at TCP level:

```
Frame 10 (62 bytes on wire, 62 bytes captured)
...
Transmission Control Protocol
  Source port: 4252 (4252)
  Destination port: http (80)
  Sequence number: 3290220049
  Header length: 28 bytes
  Flags: 0x0002 (SYN)
    0... .... = Congestion Window Reduced (CWR): Not set
    .0... .... = ECN-Echo: Not set
    ..0. .... = Urgent: Not set
    ...0 .... = Acknowledgment: Not set
    .... 0... = Push: Not set
    .... .0.. = Reset: Not set
    .... ..1. = Syn: Set
    .... ...0 = Fin: Not set
...
...
```

```
Frame 11 (62 bytes on wire, 62 bytes captured)
...
Transmission Control Protocol
  Source port: http (80)
  Destination port: 4252 (4252)
  Sequence number: 1602605975
  Acknowledgement number: 3290220050
  Header length: 28 bytes
  Flags: 0x0012 (SYN, ACK)
    0... .... = Congestion Window Reduced (CWR): Not set
    .0... .... = ECN-Echo: Not set
    ..0. .... = Urgent: Not set
    ...1 .... = Acknowledgment: Set
    .... 0... = Push: Not set
    .... .0.. = Reset: Not set
    .... ..1. = Syn: Set
    .... ...0 = Fin: Not set
...
...
```

Firewall: Netfilter

```
Frame 12 (54 bytes on wire, 54 bytes captured)
...
Transmission Control Protocol
    Source port: 4252 (4252)
    Destination port: http (80)
    Sequence number: 3290220050
    Acknowledgement number: 1602605976
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0.... .... = Congestion Window Reduced (CWR): Not set
        .0... .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 16944
    Checksum: 0xe79b (correct)
```

From these first observations, we can deduce some interesting things. Considering TCP exchanges between two sockets :

- When a TCP packet contains the SYN flag, a new connection is started.
- when a TCP packet contains the SYN and ACK flags, the connection is accepted, it is therefore established,
- when a packet contains only the ACK flag, the connection continues.

Firewall: Netfilter

If we look further:

No.	Time	Source	Destination	Protocol	Info
29	10.427697	192.168.0.10	212.27.35.1	TCP	4252 > http [ACK]
30	10.731147	192.168.0.10	212.27.35.1	TCP	span class="bhly">>4253 > http span class="bhly">[S
31	10.752981	212.27.35.1	192.168.0.10	TCP	http > 4253 [SYN, ACK]
32	10.753165	192.168.0.10	212.27.35.1	TCP	4253 > http [ACK]
33	10.753707	192.168.0.10	212.27.35.1	HTTP	GET /images/titre.gif HTTP/1.1
34	10.780941	192.168.0.10	212.27.35.1	TCP	4252 > http [FIN, ACK]

- The same client (192.168.0.10) opens a new TCP connection on the same server (212.27.35.1), still on port 80, but waits for responses on a new port **4253**, while the previous connection still exists
- The client ends the previous connection (frame 34) with the flag [FIN] only once the second connection is established
- In these conditions, it is relevant to think that this new connection is directly **RELATED** with the first one

Firewall: Netfilter

If we set up a system capable of memorizing what is happening on the TCP layer, then it will become possible to know if a connection is in one of these states:

- **NEW**
 - new connection (it contains the SYN flag),
- **ESTABLISHED**
 - connection already established, it should not contain SYN or END,
- **RELATED**
 - the connection has a direct relationship with an already established connection,
- **INVALID**
 - the connection is not compliant, contains an abnormal set of flags, cannot be classified into one of the three previous categories.

Firewall: Netfilter

- This is interesting, because we can prohibit a priori any **NEW** connection from entering our installation,
 - there is no reason for a NEW connection if we do not have a server
 - Eventually, we can for example create exceptions for ssh ports, if we want to access our machine from the Net
- However, any **ESTABLISHED** or **RELATED** connection must be able to enter from the Net.
- In the other direction, from LAN to the Net, **NEW** packets must be able to pass, as well, of course as **ESTABLISHED** and **RELATED**.
- **INVALID** packets could possibly be traced in the logs.

And for UDP ?

- There is just no connection
- It will therefore be impossible to precisely define the state of a UDP exchange
- We can set up a "timer" to decide the state of a UDP packet.
- We can take the simple example of a DNS query from our private network:
 - The first UDP packet leaves our network, on a known and identified port (53) to a DNS server.
 - We can decide to let it pass and we call it "NEW". It starts a timer,
 - if before the timer expires, we receive a UDP packet from the DNS server, we will consider it to be an "ESTABLISHED" packet.

In Practice: a main connection tracking module is loaded dynamically if necessary, this is the **ip_conntrack** module

Firewall: Netfilter

```
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source          destination
ACCEPT     tcp  --  anywhere        anywhere         tcp dpt:www
ACCEPT     tcp  --  anywhere        anywhere         tcp dpt:ssh
ACCEPT     tcp  --  anywhere        anywhere         tcp dpt:imap2
```

- **target:** what the rule does. Here it is ACCEPT, that is to say that this line authorizes a port and/or an IP
- **prot:** the protocol used (tcp, udp, icmp)
- **source:** the source IP. For **INPUT**, the source is the remote computer that connects to you
- **destination:** the destination IP. For **OUTPUT**, it is the computer to which we connect
- Last column: it indicates the port after the colon " : »
 - This port is displayed in letters, but with -n you can get the corresponding number.

Firewall: Netfilter

- The order of the rules is important
- IPtables reads from top to bottom and the position of these rules influences the final result.
- The rules are numbered
 - To get the numbers, add **--line-numbers**:

```
# iptables -L --line-numbers
Chain INPUT (policy DROP)
num  target     prot opt source          destination
1    ACCEPT     tcp  --  anywhere       anywhere        tcp dpt:www
2    ACCEPT     tcp  --  anywhere       anywhere        tcp dpt:ssh
3    ACCEPT     tcp  --  anywhere       anywhere        tcp dpt:imap2
```

Add and remove rules

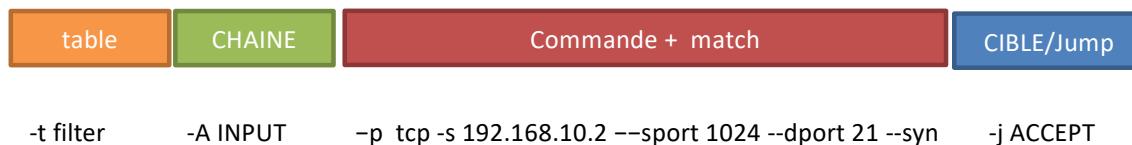
Here are the main commands to know:

- A chain:** adds a rule at the end of the list for the chain indicated (INPUT or OUTPUT, for example).
- D chain #rule:** removes rule no #rule for the chain indicated.
- I chain #rule:** inserts a rule in the middle of the list at the position indicated by #rule.
 - If you do not specify a position #rule: the rule will be inserted first, at the very top of the list.
- R chain #rule:** replaces rule #rule in the chain indicated.
- L:** list the rules
- F chain:** clears all the rules of the chain indicated.
- P chain rule:** modifies the default rule for the chain.
 - This allows you to say, for example, that by default all ports are closed, except those that have been specified in the rules.

Firewall: Netfilter

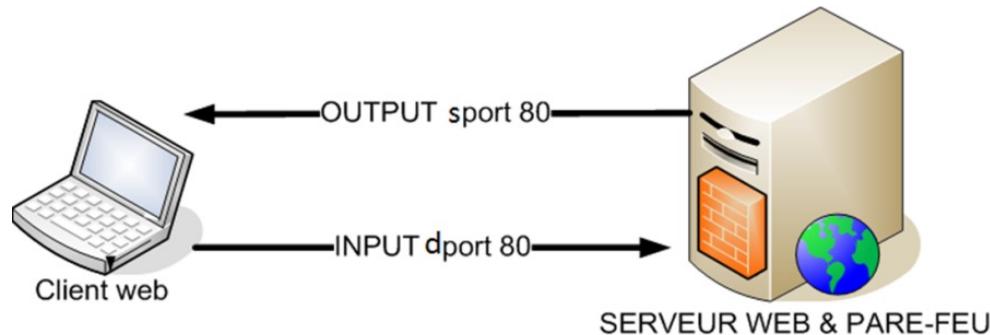
In general, adding a rule takes place according to this scheme:

- iptables -A (chain) -p (protocole) --dport (port) -j (decision/target)
- # iptables -A INPUT -p tcp --dport ssh -j ACCEPT
- # iptables -A INPUT -p icmp -j ACCEPT
- iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
- Default Policy:
- # iptables -P INPUT DROP



Firewall: Netfilter

- Example Web filtering
 - Allow a client to connect to a web server with a firewall

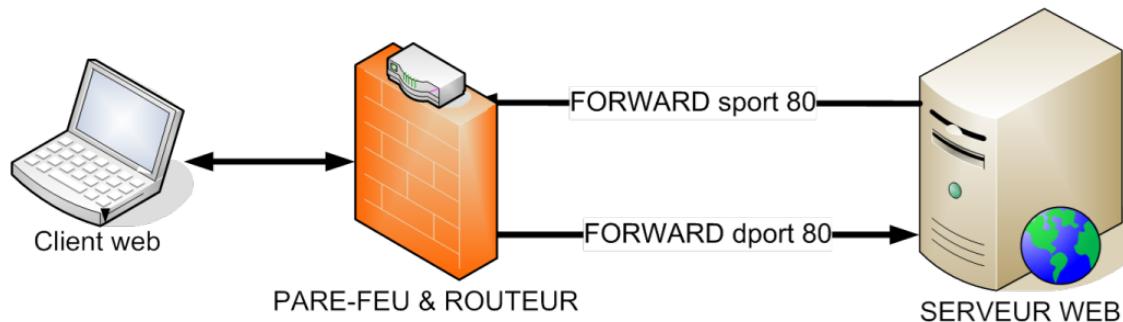


```
iptables -A INPUT -p tcp --sport 1024:65535 --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A OUTPUT -p tcp --sport 80 --dport 1024:65535 -m state --state ESTABLISHED -j ACCEPT
```

Firewall: Netfilter

- Example Web filtering
 - Allow a client to connect to a web server via a filtering router (equipped with a firewall)



```
iptables -A FORWARD -p tcp --sport 1024:65535 --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT  
iptables -A FORWARD -p tcp --sport 80 --dport 1024:65535 -m state --state ESTABLISHED -j ACCEPT
```

Firewall: Netfilter

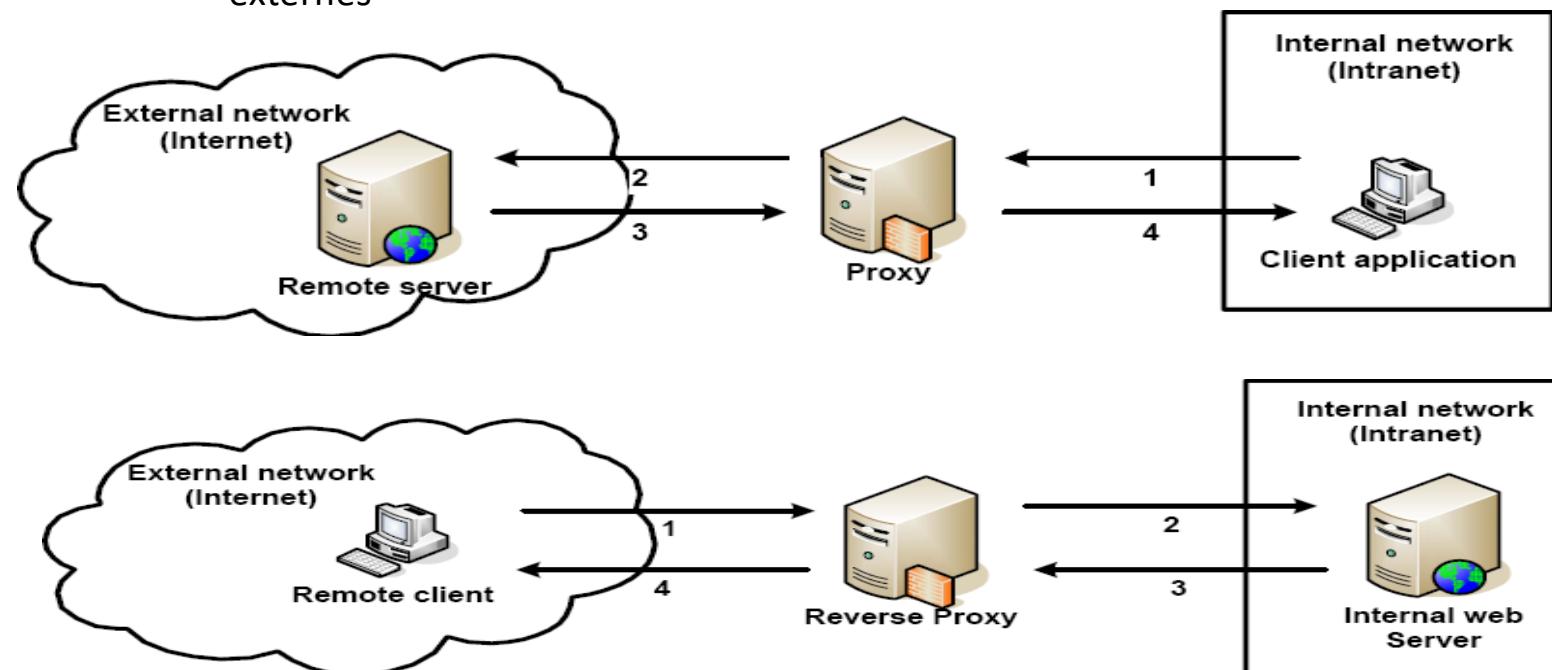
- Le filtrage Web au niveau du noyau...!
 - Le filtrage applicatif est possible au niveau de Netfilter/iptables mais à quel prix?
 - Exemple d'une règle contre le SQLi par tautologie: or 1=1--

`iptables -A INPUT -m string --algo bm --hex-string '|6f 72 20 31 3d 31 20 2d 2d|' -j DROP`

- Aucune sémantique sur le contenu
- Le nombre de règles peut vite exploser pour couvrir une seule classe d'attaque
- Ressources consommées (noyau monopolisé par la pile protocolaire)

Proxy/Reverse Proxy

- Les proxy sont des relais applicatif dans le sens réseau interne vers le réseau externe
 - On les appelle également des ALG (Application Level Gateway)
 - Dans le sens inverse on les appelle reverse proxy
- Leur but est de protéger l'accès des clients internes vers des serveurs externes
 - Dans le sens inverse on protège les serveurs internes des accés des clients externes



Proxy/Reverse Proxy

- Avantages:
 - Centralisation de la gestion des serveurs (ou des applications):
 - Politique de sécurité communes
 - Centralisation de la journalisation des logs
 - Optimisation des performances
 - Système de cache
 - Répartition de charge
 - Déploiement distribué d'une application sur plusieurs serveurs
 - Fonctions de sécurité avancées
 - Extension des mécanismes d'authentification http (LDAP par exemple)
 - Contrôle d'accès par adresse IP.

Proxy/Reverse Proxy

- Inconvénients:
 - Problème de finesse du filtrage réalisé.
 - difficile de réaliser un filtrage qui ne laisse rien passer, vu le nombre de protocoles de niveau 7.
 - connaître les règles protocolaires de chaque protocole filtré pose des problèmes d'adaptabilité à de nouveaux protocoles puisque chaque service applicatif nécessite un proxy spécifique.
 - Le filtrage applicatif apporte plus de sécurité que le filtrage de paquet avec état,
 - mais cela se paie en performance.
 - Ce qui exclut l'utilisation d'une technologie 100 % proxy pour les réseaux à gros trafic..
 - Chaque application ou service doit être supporté par le reverse proxy
 - Le reverse proxy applique la politique de sécurité associée au réseau
 - Si une application n'est pas supportée par le reverse proxy cela ne peut pas « fonctionner »

Architecture de sécurité

- *demilitarized zone (DMZ)*

DMZ

Une **zone démilitarisée**, ou **DMZ** (en anglais, *demilitarized zone*) est un **sous-réseau séparé** du réseau local et **isolé** de celui-ci et d'Internet (ou d'un autre réseau) par un pare-feu. Ce sous-réseau contient les machines étant susceptibles d'être accédées depuis Internet, et qui n'ont pas besoin d'accéder au réseau local.

Objectifs de la DMZ:

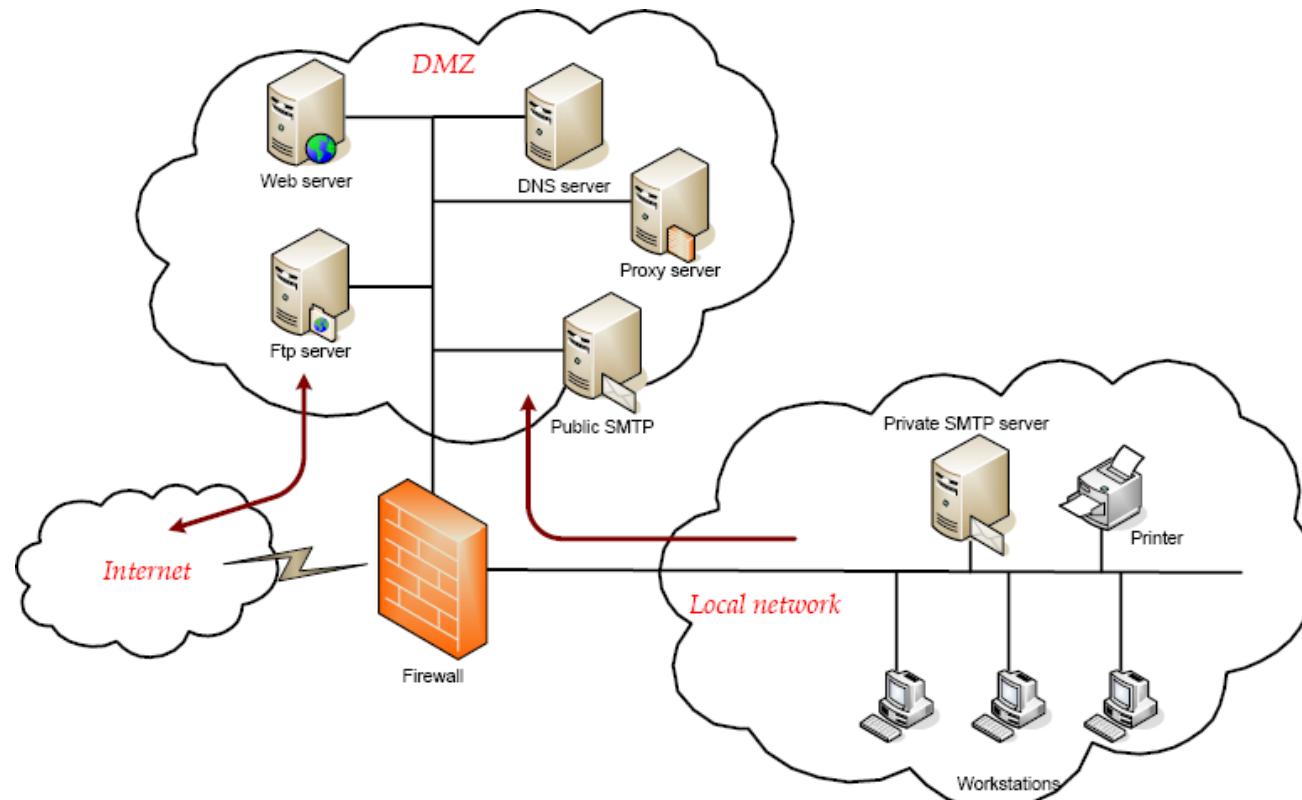
- donner accès à des services tout en protégeant l'accès au réseau Interne

Mettre en place une zone tampon:

- entre le réseau externe et le réseau Interne
- la zone tampon caractérise la DMZ
- placement dans cette zone des services accessibles de l'extérieur

DMZ

- La communication entre la DMZ et le réseau Interne n'est généralement pas autorisée
- Une politique de sécurité spécifique est définie pour la DMZ
- Elle doit être différente de la politique de sécurité du réseau Interne
- Pour des raisons de continuité de service: on duplique certains services entre la DMZ et le réseau Interne
- Les requêtes de services sont entrantes vers la DMZ et non le contraire



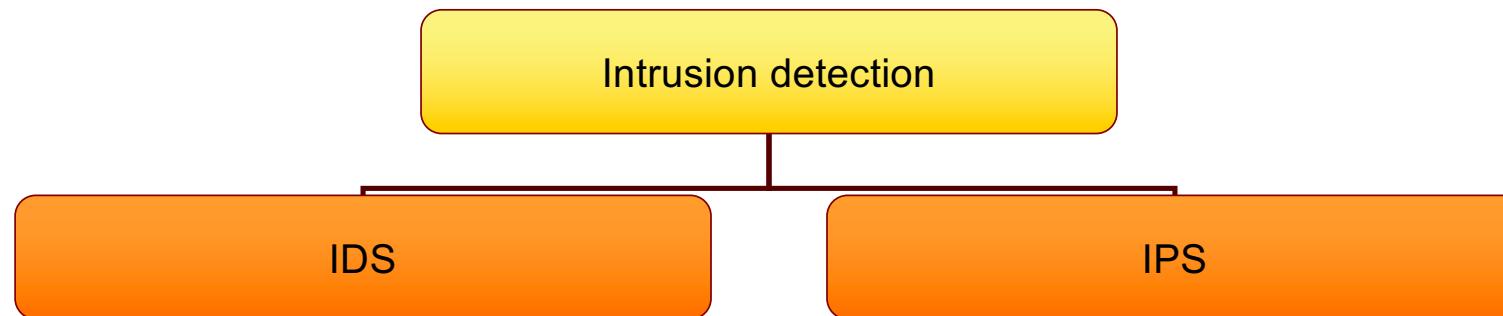
Intrusion Detection/Prevention Systems



Intrusion Detection/Prevention Systems

- Définition

- *Intrusion detection is the act of detecting unwanted traffic on a network or a device*



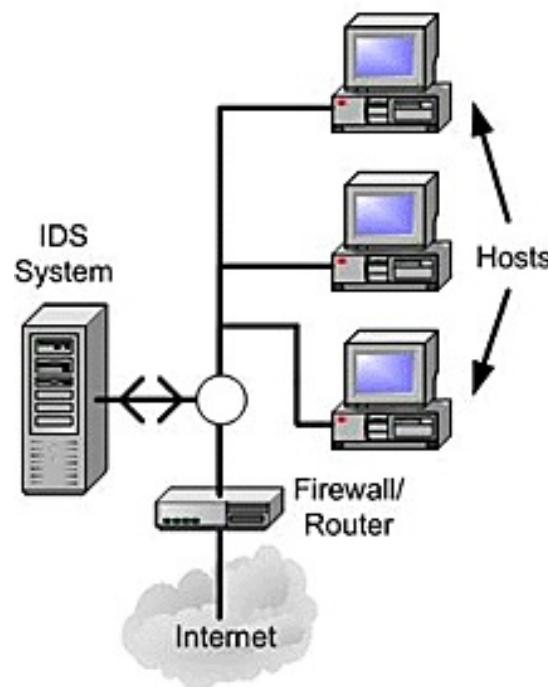
- **IDS (système de détection d'intrusion)** ensemble de composants logiciels ou matériels dont la fonction principale est de détecter et analyser toute tentative d'effraction volontaire ou non dans un SI ainsi que toute altération éventuelle de ces données
 - **IPS (système de prévention d'intrusion)** ensemble de composants logiciels ou matériels dont la fonction principale est d'empêcher toute activité suspecte détectée au sein d'un système

Intrusion Detection/Prevention Systems

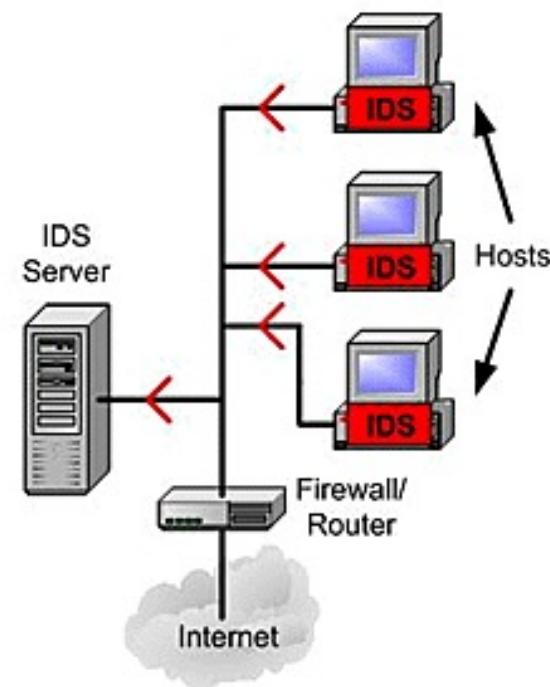


Intrusion Detection/Prevention Systems

Network Based IDS

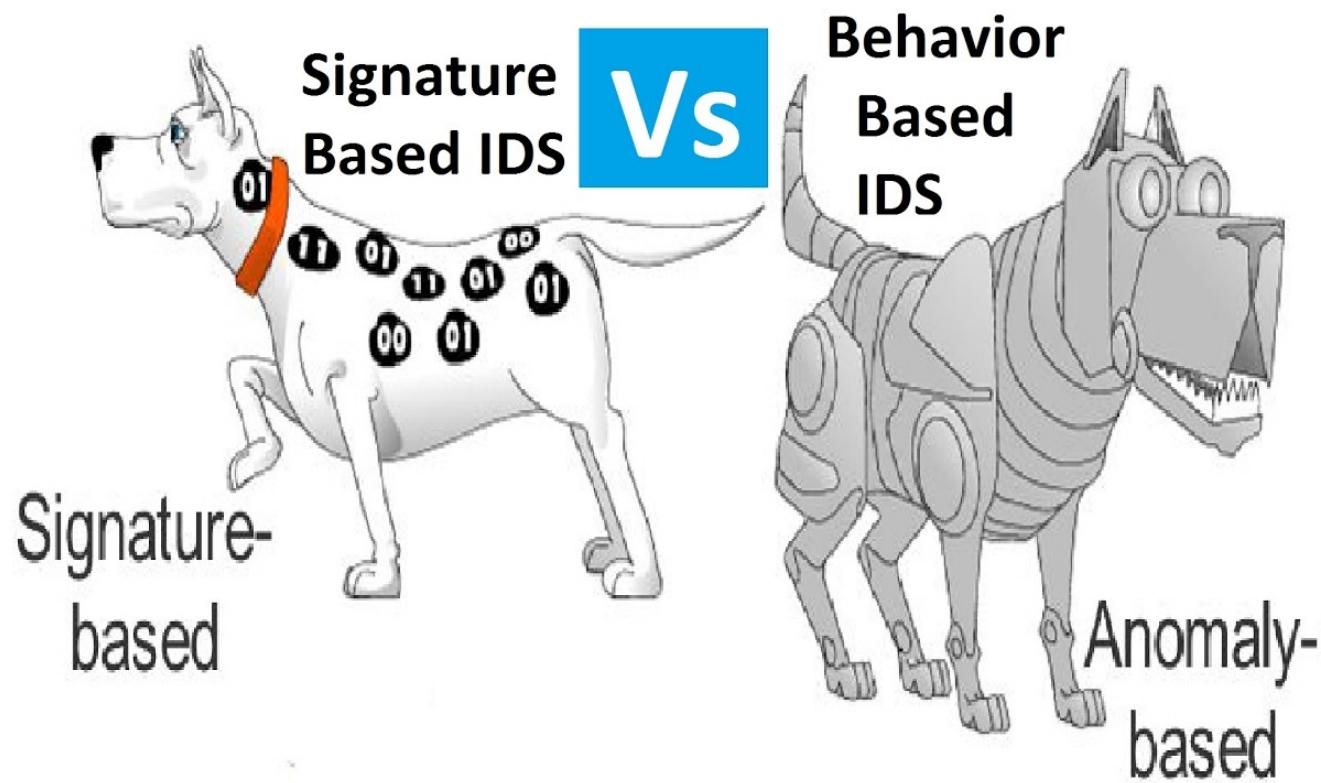


Host Based IDS



Intrusion Detection/Prevention Systems

- Deux approches de détection
 - Détection comportementale
 - Détection à base de signature



Intrusion Detection/Prevention Systems

Signature-Based Detection

- Attaques courantes peuvent être caractérisées avec une signature
- Base de données de signatures dans l'IDS
 - comparer le trafic actuel avec ces signatures et détecter une attaque classique
 - Base de données des signatures doit être mise à jour très régulièrement pour être efficace
- Pas de détection pour une attaque non connue (zero day attack)

Intrusion Detection/Prevention Systems

- Méthodes de détection : par Signature
 - Exemple
 - Trouver le « motif /winnt/system32/cmd.exe » dans une requête http
 - Voir code source du botnet Hybrid_V1.0
 - Avantages
 - Simplicité de mise en œuvre
 - Rapidité de diagnostic
 - Précision (en fonction des règles)
 - Identification du procédé d'attaque (Cibles, Sources, Outils)
 - Inconvénients
 - Ne détecte que les attaques connues
 - Maintenance de la base
 - Techniques d'évasion possibles dès lors que les signatures sont connues

Intrusion Detection/Prevention Systems

- Anomaly-Based Detection
 - L'IDS établit un modèle de trafic (fonctionnement) normal
 - Nombre de connexions
 - Types de trafic à telles heures
 - Comparaison du trafic actuel au modèle
 - Si les deux trafics sont trop différents => anormal peut-être qu'une attaque est en cours

Intrusion Detection/Prevention Systems

- Modélisation du système : création d'un profil normal
 - Phase d'apprentissage
 - Déetecter une intrusion consiste a déetecter un écart
 - Exemple de profil :
 - Volumes des échanges réseau
 - Appels systèmes d'une application
 - Commandes usuelles d'un utilisateur
 - Repose sur des outils de complexité diverses
 - Seuils
 - Statistique
 - Méthodes probabilistes
 - Complexité de l'implémentation et du déploiement

Intrusion Detection/Prevention Systems

- Méthodes de détection : Par anomalie

- Avantages

- Permet la détection d'attaque inconnue
 - Facilite la création de règles adaptées à ces attaques
 - Difficile à tromper

- Inconvénients

- Les faux-positifs sont nombreux
 - Générer un profil est complexe
 - Durée de la phase d'apprentissage
 - Activité saine du système durant cette phase ?

Intrusion Detection/Prevention Systems

- Méthodes de détection : Par intégrité
 - Vérification d'intégrité
 - Le principe
 - Prendre une photographie (à base de checksum)
 - contrôle d 'intégrité et liste des process autorisés
 - à l 'instant T initial et « sain »
 - Reprendre régulièrement une photographie
 - à des intervalles réguliers (et planifiés)
 - Comparer et alerter si nécessaire

Intrusion Detection/Prevention Systems

- Composants d'un IDS

- **Capteurs**

- Permettant de collecter les données et de les envoyer à l'analyseur

- **Analyseur**

- Reçoit les données des sondes, pour décider si une intrusion a réellement eu lieu

- **Interface utilisateur**

- Permettant à l'utilisateur d'avoir une vue sur ce qu'il s'est passé et contrôler le fonctionnement du système

Intrusion Detection/Prevention Systems

- **Types de detections**

- **Vrai positif :**

- C'est le cas lorsqu'une attaque est détectée et qu'elle a bien lieu

- **Faux positif :**

- C'est le cas lorsqu'une attaque est détectée alors qu'en réalité elle n'a pas lieu

- **Vrai négatif :**

- C'est le cas lorsqu'aucune attaque n'est détectée et qu'il n'y en a effectivement aucune

- **Faux négatif :**

- C'est le cas lorsque l'IDS n'a pas détecté une attaque en cours

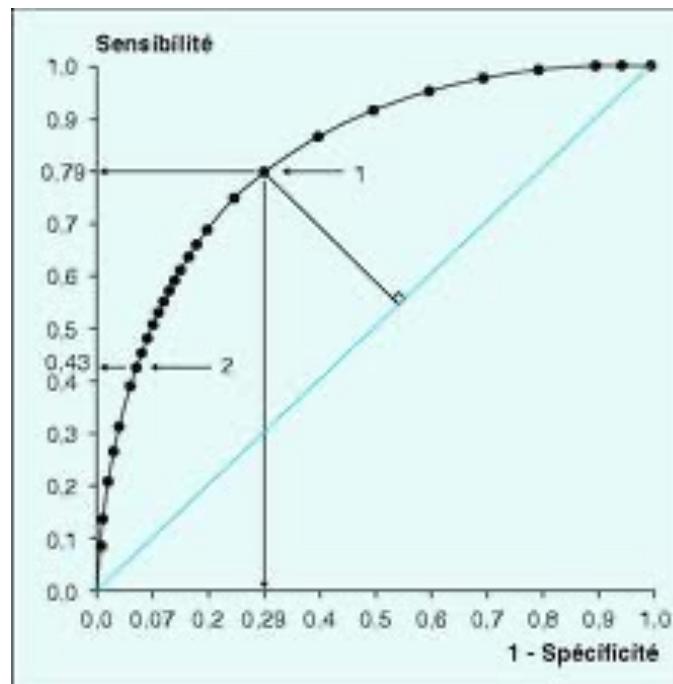
Intrusion Detection/Prevention Systems

- **Courbe ROC (receiver operating characteristic)**

- Les performances des IDS sont généralement évaluées à l'aide de leur
 - sensibilité
 - spécificité
 - valeurs prédictives positives et négatives

$$Se = VP / (VP + FN)$$

$$Sp = VN / (VN + FP)$$

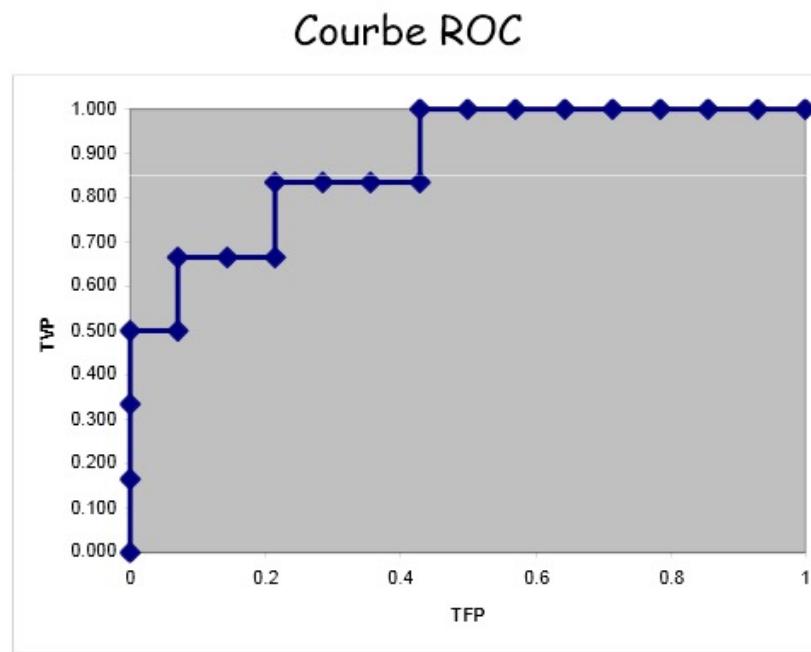


Intrusion Detection/Prevention Systems

- Sensibilité
 - Capacité d'un test à détecter les cas d'une attaque
 - Capacité d'identifier correctement une attaque qui existe
- Spécificité
 - Capacité d'un test à détecter correctement qu'il n'y a pas une attaque
- Si un test a une sensibilité à 100%
 - toutes les attaques sont correctement détectées , il n'y a aucun faux négatif
- Si un test a une spécificité de 100%
 - tous les cas normaux sont correctement identifiés, il n'y a aucun faux positif
- Spécificité et sensibilité sont entre 0 et 1, exprimées en %
- ROC utilisé aussi dans le domaine médical
 - Détection maladie/non maladie

Intrusion Detection/Prevention Systems

Classe	TFP	TVP
	0	0.000
+	0.000	0.167
+	0.000	0.333
+	0.000	0.500
-	0.071	0.500
+	0.071	0.667
-	0.143	0.667
-	0.214	0.667
+	0.214	0.833
-	0.286	0.833
-	0.357	0.833
-	0.429	0.833
+	0.429	1.000
-	0.500	1.000
-	0.571	1.000
-	0.643	1.000
-	0.714	1.000
-	0.786	1.000
-	0.857	1.000
-	0.929	1.000
-	1.000	1.000



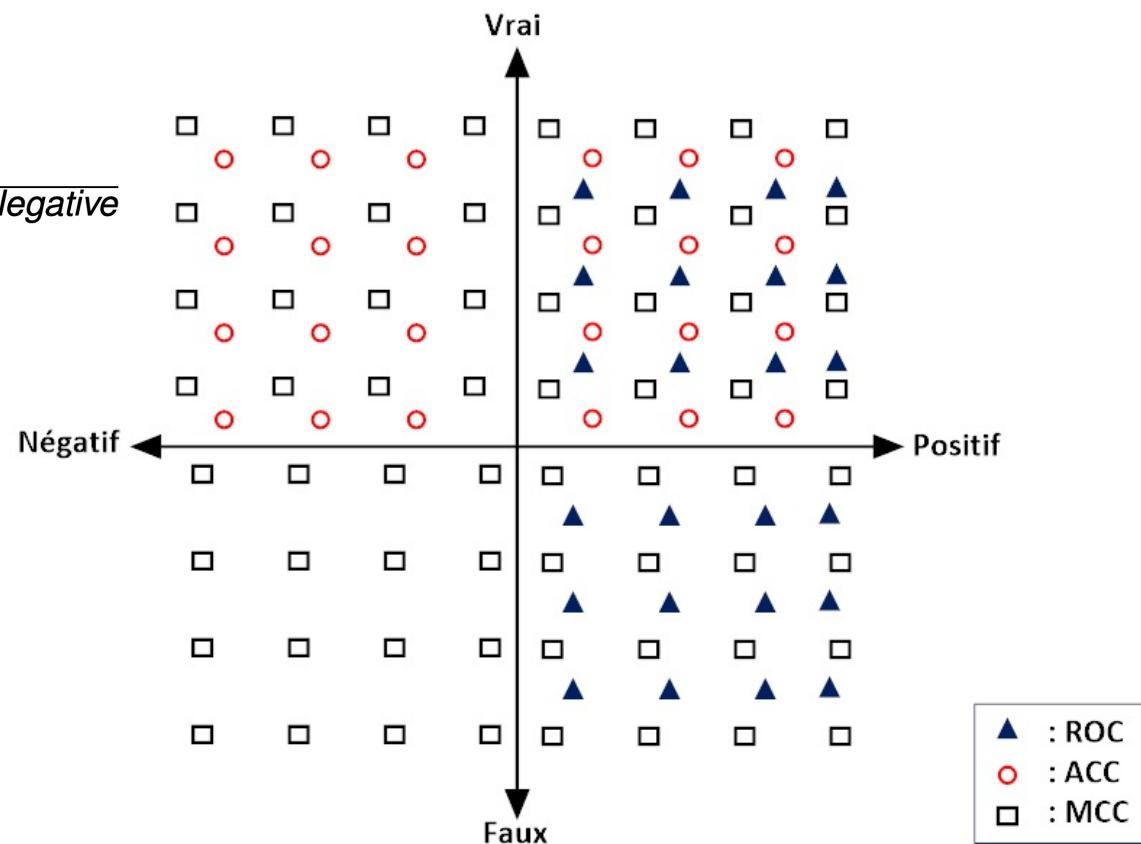
TVP = Sensibilité = VP/Positifs

TFP = 1 – Spécificité = FP/Négatifs

Intrusion Detection/Prevention Systems

$$\bullet \text{ } MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

$$\bullet \text{ } ACC = \frac{\text{TruePositive} + \text{TrueNegative}}{\text{TruePositive} + \text{FalsePositive} + \text{TrueNegative} + \text{FalseNegative}}$$



Intrusion Detection/Prevention Systems

- Host-based IDS
 - AIDE—Advanced Intrusion Detection Environment
 - CSP Alert-Plus
 - eEye® Retina®
 - Hewlett Packard®-Unix (HP-UX®) 11i Host Intrusion Detection System (HIDS)
 - IBM® RealSecure® Server Sensor
 - Lumension® Application Control
 - McAfee® Host Intrusion Prevention
 - Osiris®
 - Tripwire® Enterprise, Tripwire for Servers

McAfee®



Intrusion Detection/Prevention Systems

- AIDE : Advanced Intrusion Detection Environment
 - Construire une base de signatures de fichiers
 - Re-calculer ces empreintes périodiquement ou au besoin, en les confrontant à la base
 - Algorithmes d'empreinte
 - MD5, SHA1, RMD160, Tiger, SHA256, SHA512, Whirlpool, Haval
 - Très utiles en cas d'intrusion, afin de découvrir ce qui a été changé
 - journaux modifiés, fichiers ajoutés à certains endroits, comme netstat, lsof, who, sshd modifiés, fichiers de configuration, pages web, etc.)
 - Très utile pour l'administration pour détecter des erreurs commises (fichiers de configuration changés, ajoutés ou effacés, modifications de binaires)
 - Open source
 - BSD Platforms (FreeBSD/NetBSD/ OpenBSD/Apple Mac® OS X), Linux, Solaris®, IBM® AIX

Intrusion Detection/Prevention Systems

- OSSEC



- OSSEC: est un Host-based Intrusion Detection System Open Source soutenu par la société Trend Micro, il peut fonctionner en mode indépendant ou en mode client serveur.
- Il opère au niveau des systèmes hôtes multiplateformes: Linux, MacOS, Solaris, HP-UX, AIX et Windows
- Il permet l'analyse des logs, de vérifier l'intégrité des fichiers sensibles et de détecter les rootkits.
- Des alertes ou des réponses actives sont envoyées en temps réel pour contrer les tentatives d'intrusion.
- La détection se fait sur la base de règles stockées dans des fichiers XML.
- Les règles peuvent être « atomiques » décrivant un événement simple ou « composite » permettant de corrélérer plusieurs événements.
- Téléchargement: http://www.ossec.net/?page_id=19

Intrusion Detection/Prevention Systems



- OSSEC

- Points forts:

- Architecture client serveurs qui permet d'alléger la charge des agents.
 - Écriture de nouvelles règles, l'externalisation
 - Corrélations entre les évènements pour une détection plus efficaces des attaques complexes et éviter les faux négatifs.
 - Installation facile avec un assistant.

- Points faibles:

- Nombre d'agents est limité à 256 par manager « serveur »
 - Nombre d'alerte limité par heure.

Intrusion Detection/Prevention Systems

- Network-based IDS
 - Arbor Networks Peakflow® X
 - ArcSight®
 - Bro
 - Snort
 - Cisco® ASA 5500 Series IPS Edition
 - Cisco Guard XT
 - Juniper Networks® IDP



Intrusion Detection/Prevention Systems

- SNORT

- Snort: est le moteur de détection d'Intrusion OpenSource le plus populaire et le plus mature.
- Originellement conçu pour être un sniffer réseau plus puissant que tcpdump
- Les signatures d'attaques sont décrites à partir d'un langage déclaratif de règles sous forme de : [action][header][option]
- Exemple de règle:

alert any any -> any any (flags: SF,12; msg: "Possible SYN FIN scan";)



```
Activities Terminal ▾ mars 30 00:33 •
badis@badis-HP: ~
>_
alert icmp any any -> $HOME_NET any (msg:"BBB - Un PING from Windows - B"; content:"|61 62 63 64 65 66 67 |"; itype:8; depth:32; sid:1000001;)
alert icmp any any -> $HOME_NET any (msg: "BBB - requête echo"; sid: 1000002; rev: 1; itype: 8; icode: 0;)
alert icmp any any -> $HOME_NET any (msg:"BBB - Ping Windows détecté"; content:"abcdefghijklmnop"; depth:32; sid: 1000003;)

alert icmp any any -> any any (msg:"BBB - ICMP flooding attack!!!"; threshold: type both, track by_src, count 5, seconds 1; sid:1000000; rev:1;)
alert tcp any any -> $HOME_NET 80 (msg : "BBB -TCP SYN flooding attack !!!"; flags: S; threshold: type both, track by_dst, count 10, seconds 1; sid:1000004;)
alert ip any any -> any any (msg:"BBB - Fragmentation alert !!!"; fragbits:M; sid:1000005;)
alert tcp any any -> any any (msg: "BBB- XMAS scan alert !!!"; flags: FPU; sid:1000006;)
```

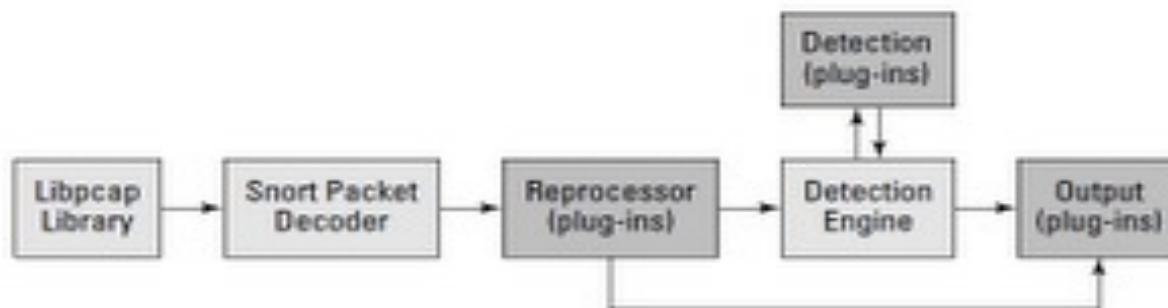
Intrusion Detection/Prevention Systems

- Léger
 - Pas d'interface graphique
 - Peu coûteux en ressources
- Puissant et temps réel
 - Permet une définition précise des signatures
 - Robuste et portable
- Détection dans IP, TCP, UDP et ICMP
 - Dans les entêtes
 - Dans le contenu des paquets
- Détection de Nmap (scans, OS fingerprint)
- Détection de déni de service et de débordement de buffer



Intrusion Detection/Prevention Systems

- Architecture Modulaire de Snort
 - Décodeur De Paquets (Packet decoder)
 - Préprocesseurs (Preprocessors)
 - Moteur De Détection (Detection Engine)
 - Système d'alerte et d'enregistrement de log (Logging and Alerting System)
 - Modules De Sortie (Output Module) :
 - Possibilité d'enregistrer les logs dans une BDD (MYSQL/PSQL)



Intrusion Detection/Prevention Systems

- Configuration sous Unix
 - **/etc/snort/snort.conf**
 - Configuration des variables pour le réseau
 - Configuration des réseaux à écouter
 - Configuration des services à logguer (http/dns/etc...)
 - Configuration des pré-processeurs
 - Configuration des plugins de sortie Mysql/pgsql/écran/etc...
 - Choix des règles à utiliser
 - **/etc/snort/rules** (ensemble des signatures)

Intrusion Detection/Prevention Systems

- **Réactions de Snort**

- Les alertes émises par snort peuvent être de différentes nature
- Redirection de l'intégralité des alarmes sur la sortie standard
 - observer l'évolution des attaques
- Adopter des comportements visant à interdire l'accès à certaines adresses IP
 - L'IDS peut interagir avec le **firewall** afin qu'il mette à jour ses règles d'accès
 - Il existe une solution robuste, telle que "snortsam »
 - SnortSam was a plugin for Snort. The plugin allowed for automated blocking of IP addresses on following firewalls

Intrusion Detection/Prevention Systems

- Bro
 - BRO: est un IDS Open Source issu des travaux de recherche académique pour devenir actuellement un outil de détection à déploiement industriel.
 - Son architecture en couche est très proche de snort.
 - Les attaques peuvent être décrites de deux façons:
 - un langage implicite de script proche du C; exemple:

```
const watched_servers: set[addr] = {192.168.1.100, 192.168.1.101, 192.168.1.102,}  
&redef redef Notice::policy += {[${action} = Notice::ACTION_EMAIL,$pred(n: Notice::Info) ={return n$note  
== SSH::Login && n$id$resp_h in watched_servers;}];}  
• un langage de signatures pour les utilisateurs familiers snort, mais le mode préféré de Bro est le mode script.
```
 - Possibilité d'exécuter des programmes tiers après détection d'intrusion
 - Exemple: reconfigurer un routeur
 - Compatible avec les règles Snort
 - Grâce à snort2bro
 - Téléchargement du moteur, des outils et des scripts:
 - <http://www.bro.org/download/index.html>



Intrusion Detection/Prevention Systems

- Bro

- Point forts:

- Il a hérité de l'architecture fonctionnel de snort.
 - Code source évolué profitant nativement du multi-cœurs et des cartes d'accélération matériel
 - Performances plus élevées que celles de snort.
 - Reconnait plusieurs OS
 - Spécifiquement concentré sur les protocoles http et SSL.

- Points Faibles:

- Ne dispose pas d'un sniffeur « capture externe ».
 - Comme chez snort les parseurs protocolaires sont écrit en dur et non modulaires.

Intrusion Detection/Prevention Systems

Suricata: est un IDS/IPS Open Source développé par OISF en s'inspirant principalement de snort.

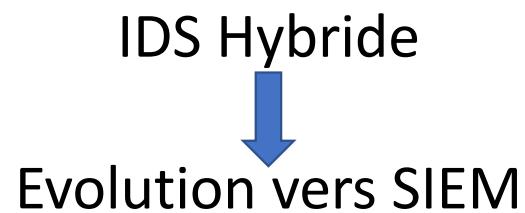
- Il est actuellement considéré comme un Proof of Concept que vraiment un concurrent à snort.
- Il reprend le même formalisme des règles de snort.
- Le code source est écrit pour profiter du multithreading et des accélérations matérielles multi-cores dès le début.
- Il propose un langage de script en LUA JIT « Just In Time » pour les traitements plus avancés sur les paquets ou sur des flux.

Téléchargement du moteur et des signatures:

<http://suricata-ids.org/download/>



Intrusion Detection/Prevention Systems



Le SIEM (*security information and event management*) gère les événements du système d'information. Le SIEM permet de gérer et corrélérer les logs. On parle de corrélation car ces solutions sont munies de moteurs de corrélation qui permettent de relier plusieurs événements à une même cause.

Intrusion Detection/Prevention Systems



- PRELUDE

- Prelude: est une plateforme Open Source de gestion des informations de sécurité « SIEM » qui collecte, normalise, trie, agrège, corrèle et affiche les événements de sécurité.
 - Prelude est composé de deux type de détecteurs:
 - Un LML (Log Monitoring Lackey) capable de traiter tout type de logs
 - Une interface de communication native avec des IDS /HIDS, et antivirus.
 - Points forts:
 - Standardisation des communications IDMEF, XML
 - Un module de corrélation des événements de plusieurs systèmes IDS/HIDS
 - Interface de visualisation graphique des alertes et des statistiques.
 - Points Faibles:
 - Ne possède pas de moteur de détection propre.
 - Téléchargement: <https://www.prelude-ids.org/projects/prelude/files>
 - Version évaluation pour un petit parc.

Prelude



- IDS hybride 1998:
 - *NIDS* : NetWork Intrusion Detection System ;
 - *HIDS* : Host based Intrusion Detection System
 - *LML* : Log Monitoring Lackey
- Architecture modulaire, distribuée et sécurisée
- Standard IDMEF (Intrusion Detection Message Exchange Format)
- Possibilité de stocker les logs dans une BDD MYSQL/PSQL
- Supporte :
 - SNORT / NESSUS et + de 30 analyseurs de logs

Intrusion Detection/Prevention Systems

- Fonctionnement de Prelude
 - Les capteurs remontent des alertes à un manager Prelude
 - Snort...
 - Syslog...
 - Prelude Iml (sensor pour une machine)
 - Le manager :
 - Collecte les alertes
 - Transforme les alertes au format de Prelude en un format lisible
 - Permet des contre-mesures à une attaque
 - La communication entre les différents programmes se fait au format IDMEF (Intrusion Detection Message Exchange Format)



Intrusion Detection/Prevention Systems

- Composition de Prelude
 - Libprelude (la librairie Prelude) : la base
 - Gestion de la connexion et communication entre composants
 - Interface permettant l'intégration de plugins
 - Prelude-LML (la sonde locale)
 - Alerte locale
 - Pour la surveillance des systèmes
 - Unix : syslog
 - Windows : ntsyslog.
 - Prelude-Manager (le contrôleur)
 - Prelude-manager centralise les messages des sondes réseaux et locales, et les traduit en alertes.
 - responsable de la centralisation et de la journalisation



Intrusion Detection/Prevention Systems

- Configuration de Prelude
 - Installation de l'ensemble du framework
 - Configuration du manager
 - /etc/prelude-manager/prelude-manager.conf
 - Configuration de lml
 - /etc/prelude-lml/prelude-lml.conf
 - Configuration de prelude
 - /etc/prelude/default/
 - Client.conf
 - Idmef-client.conf
 - Global.conf
 - Ajout de sonde : exemple snort
 - prelude-admin register snort "idmef:w" x.x.x.x --uid=0 --gid=0



Intrusion Detection/Prevention Systems



Network Security: SOC

Le SOC (Security Operating Center) est une plateforme permettant la supervision et l'administration de la sécurité du système d'information au travers d'outils de collecte, de corrélation d'événements et d'intervention à distance. Cette plateforme dont la fonction principale est de fournir des services de détection des incidents de sécurité propose également des services de résolution de ceux-ci. Le centre de sécurité va ainsi collecter les événements (sous forme de logs notamment) remontés par les composants de sécurité, les analyser, détecter les anomalies et définir des réactions en cas d'émission d'alerte.



Network Security: SIEM vs SOC

- A SIEM (Security Information and Event Management) is a specific kind of technology, providing network visibility in a security context (by indicating suspicious/illegitimate activity through set-up rules and correlation intelligence), and enabling security analysts to act on suspected threats.
- A SOC (Security Operations Centre) encompasses the People, Processes, as well as Technology involved in protectively-monitoring a network, responding to incidents, and researching/actively searching for known/unknown threats.

Network Security: SandBox



Microsoft Detours

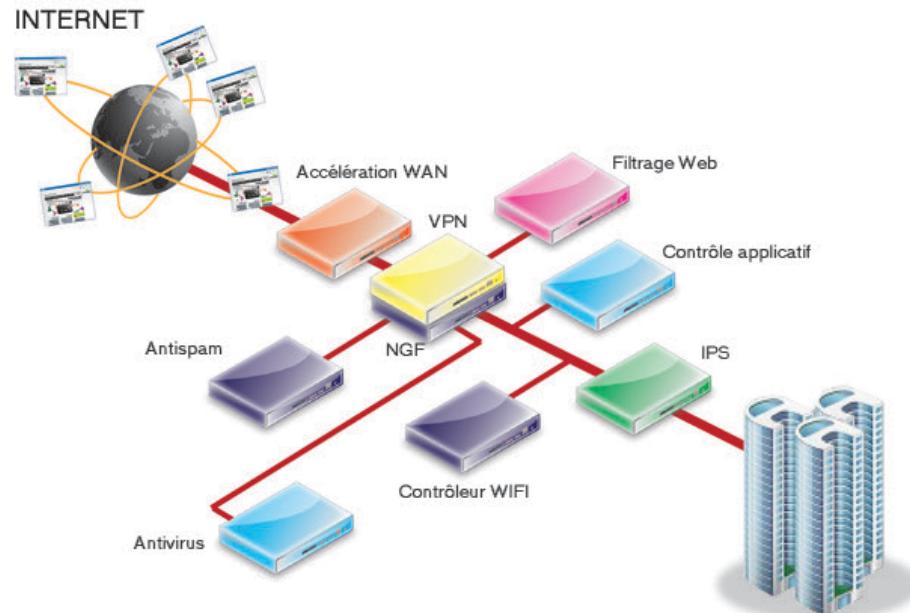
How it is in practice ?

Technologies et solutions

• Unified Threat Management ou UTM)

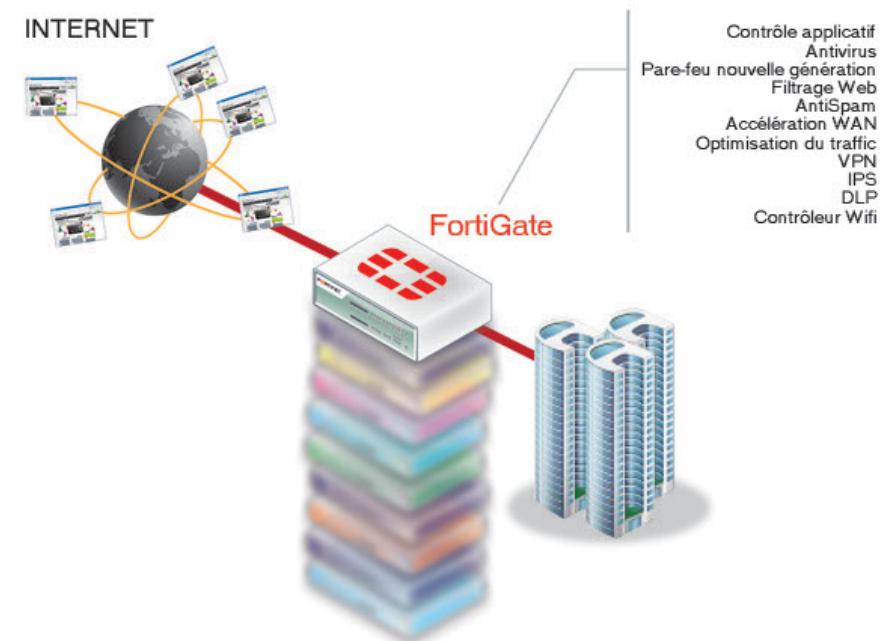
SOLUTIONS TRADITIONNELLES

Une prolifération d'appliances pour une infrastructure complexe et coûteuse



L'UTM SELON FORTINET

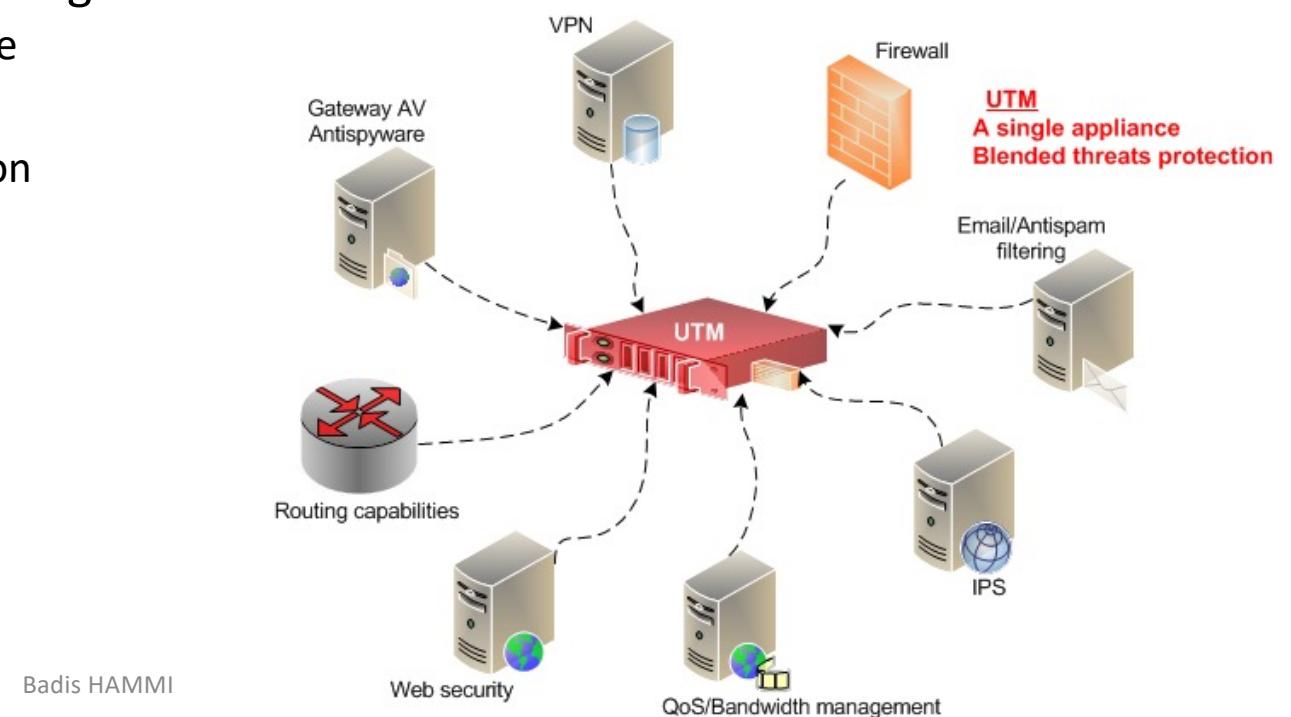
Une sécurité consolidée, simple et économique



Badis HAMMI

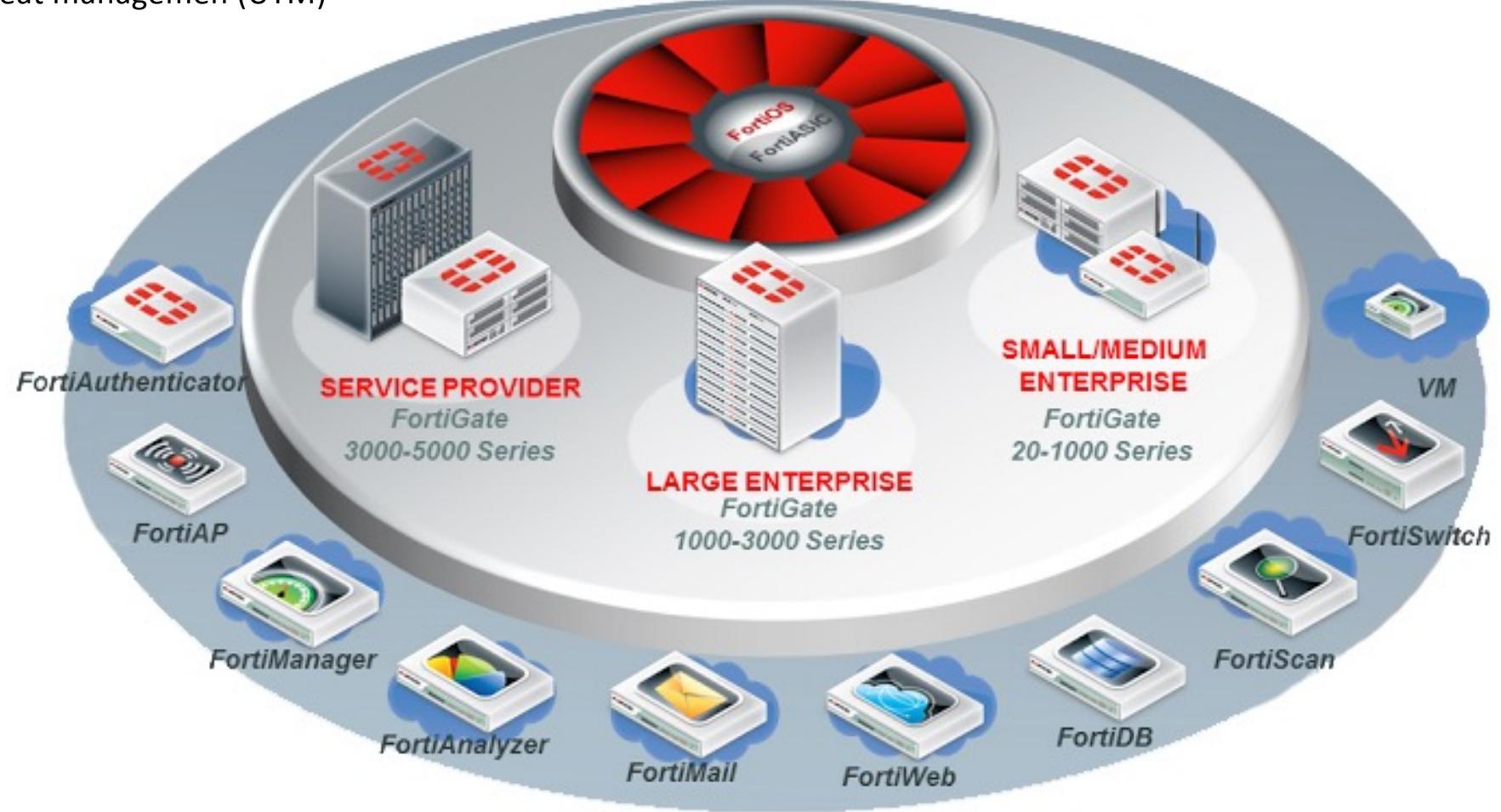
Technologies et solutions

- UTM (Unified Threat Management)
 - Systèmes de gestion unifiée des menaces
 - Outils les plus couramment utilisés dans la sécurité
 - Concept à la mode
 - Proposent de nombreuses technologies de sécurité
 - Intégrées sur une seule plate-forme
 - Fournies par un seul éditeur
 - Une seule interface d'administration



Intrusion Detection/Prevention Systems

Unified threat management (UTM)



Technologies et solutions

- Gestion unifiée des menaces (UTM)
 - Terme de sécurité proposé en 2004
 - Proposé par Charles Kolodgy du cabinet de conseil IDC (International Data Corporation)
 - Solution basée sur des pare-feu avec des fonctionnalités supplémentaires par rapport aux pare-feux traditionnels
- Leaders d'UTM sur le marché
 - Fortinet, Symantec, Juniper, WatchGuard, Kerio, Checkpoint



JUNIPER
NETWORKS



KERIO

Intrusion Detection/Prevention Systems

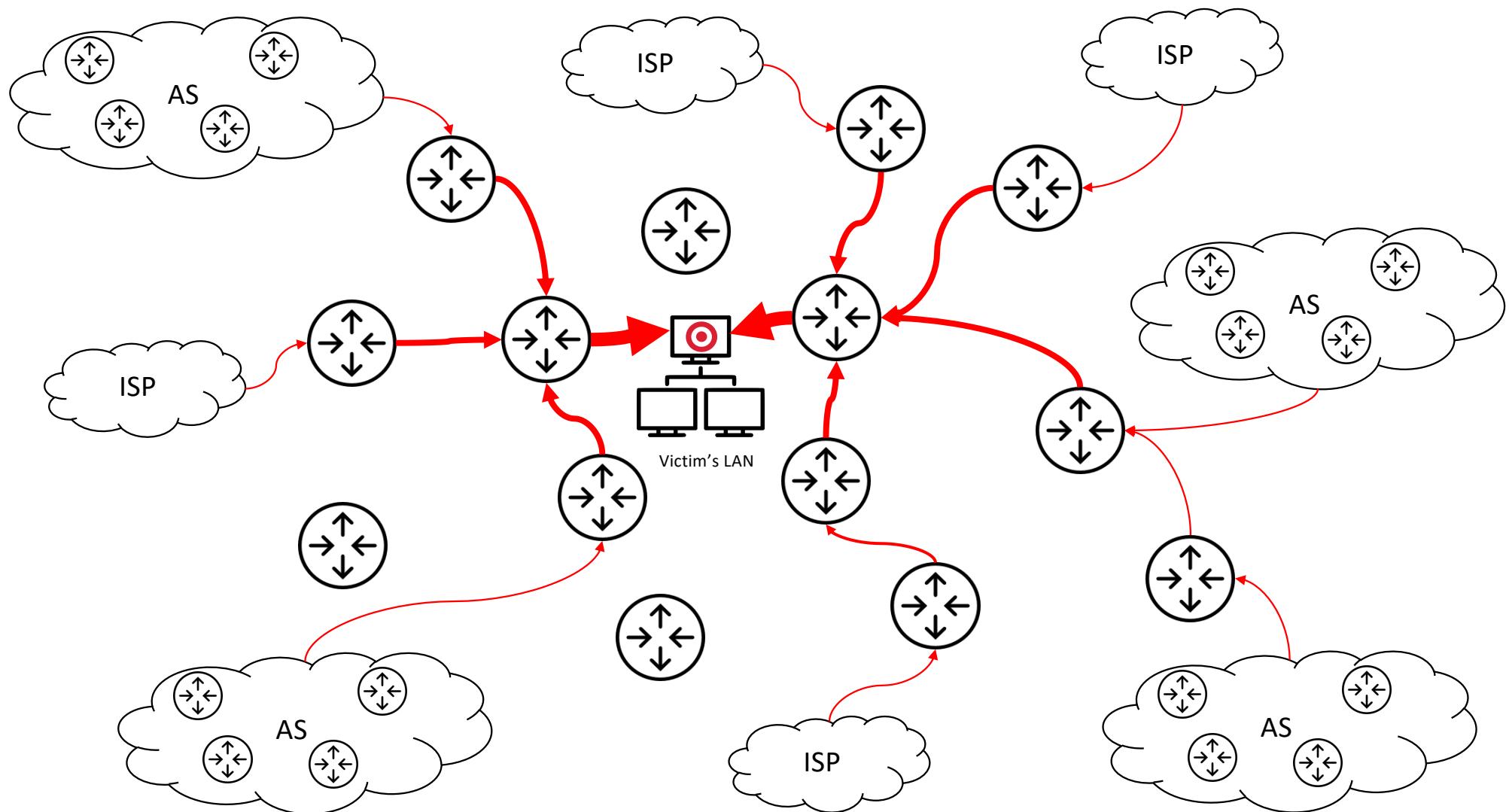
- Avantages d'une solution UTM
 - Déploiement simple
 - Beaucoup moins d'étapes au niveau de l'installation et de la configuration
 - Administration facile
 - Une seule console d'administration
 - Un seul processus de mise à jour
 - Résolution plus rapide des problèmes
 - Moins de possibilités de conflits entre les modules
 - Support assuré par un seul et même éditeur
 - Logs homogènes avec des corrélations utiles entre les différents types de données.

Intrusion Detection/Prevention System (IDS/IPS)

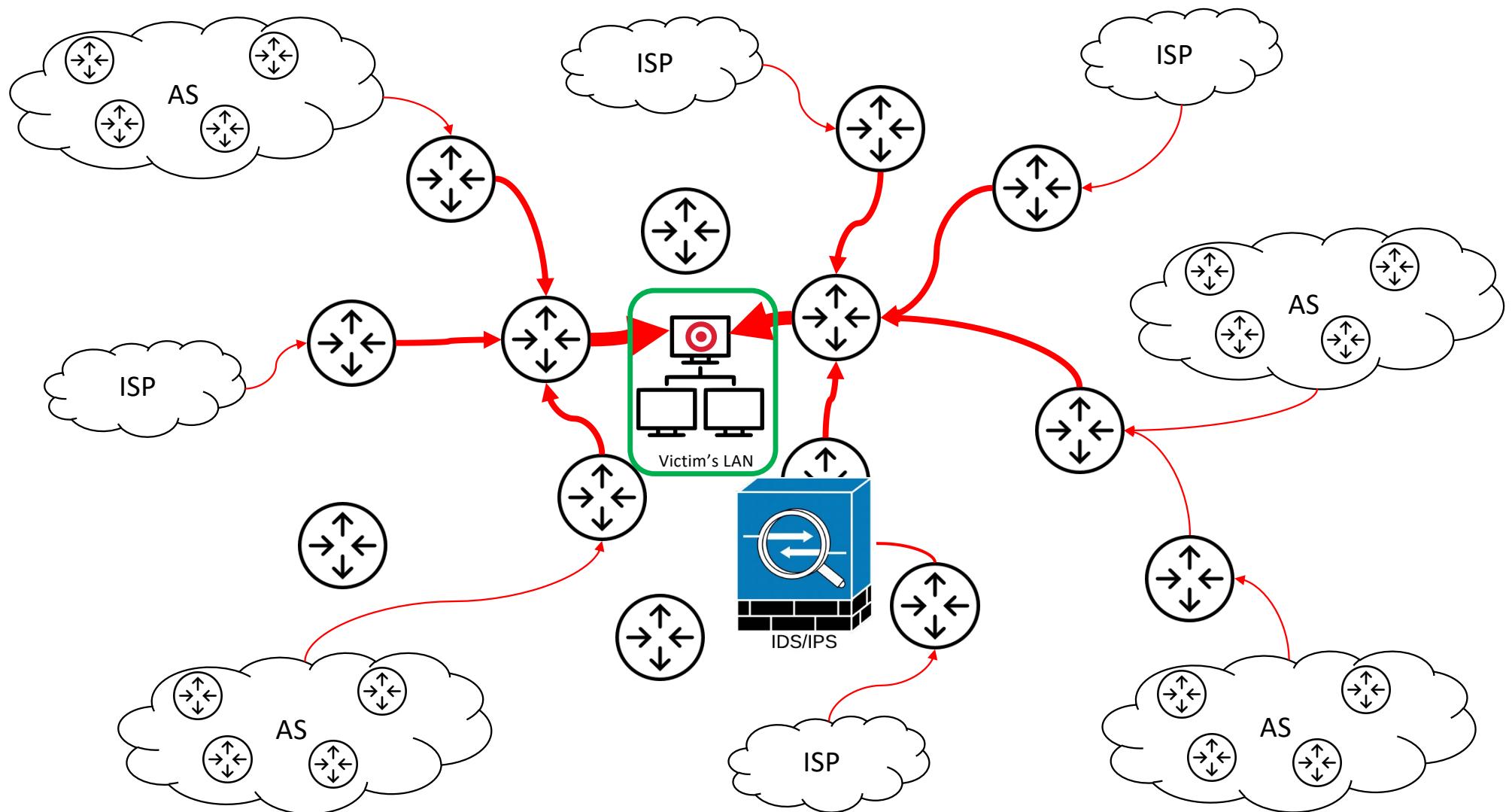
- HoneyPots
- Détection au niveau de la cible
- Détection au niveau du réseau intermédiaire
- Détection au niveau de la source (ex: IP source Spoofing !!)



Botnets/DDoS: mitigation



Botnets/DDoS: mitigation: at the target level



DDoS

- Attaque sur Spamhaus > 100 GBps 2013 → **cyberBunker**

Security

BIGGEST DDoS ATTACK IN HISTORY hammers Spamhaus

Plucky mail scrubbers battle internet carpet bombers

By John Leyden 27 Mar 2013 at 17:03

124 SHARE ▾



SPAMHAUS

The Biggest Cyber Attack In History Is Taking Place Right Now

Dylan Love 27 Mar 2013, 14:14

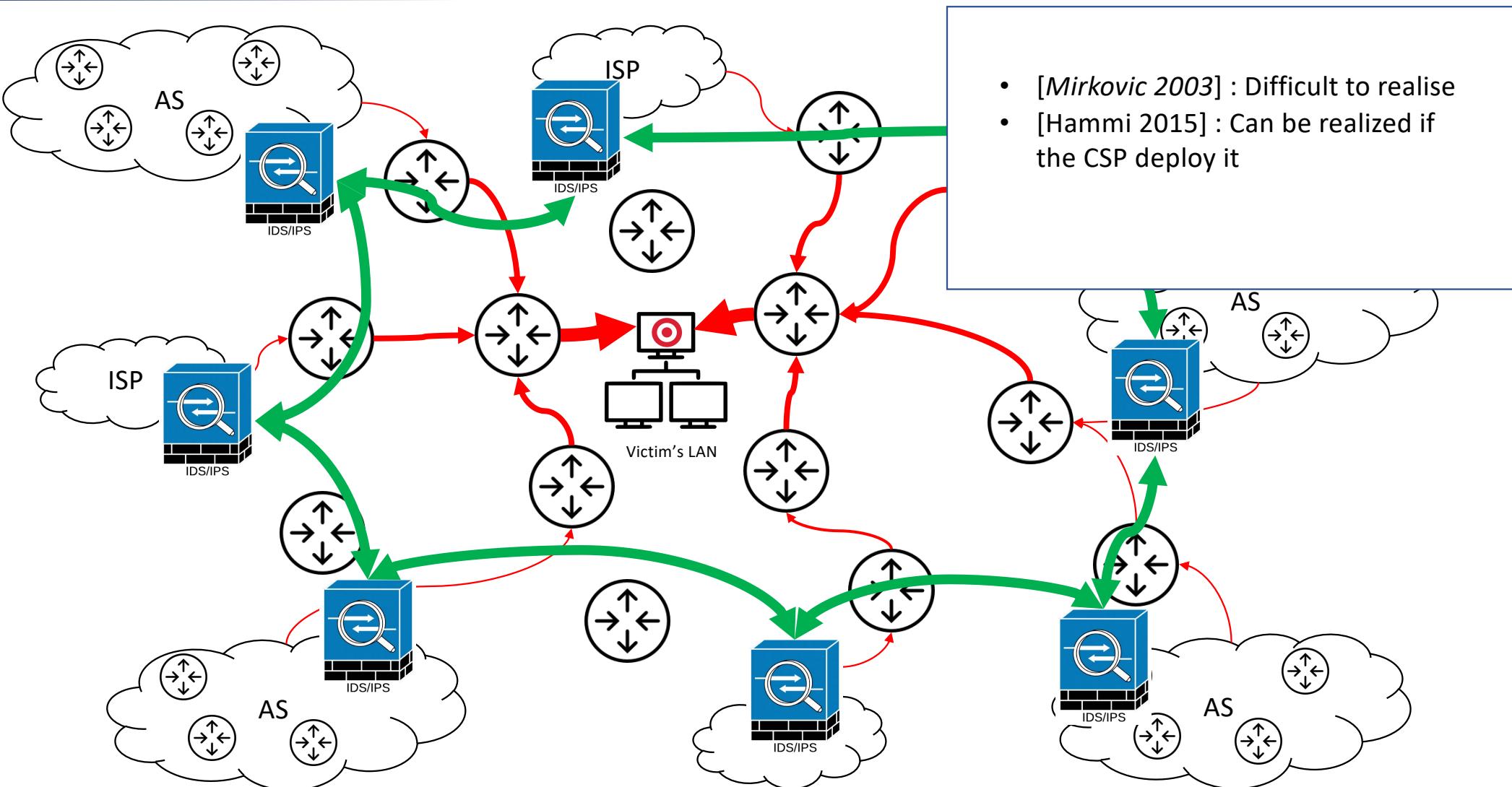


CLOUDFLARE.

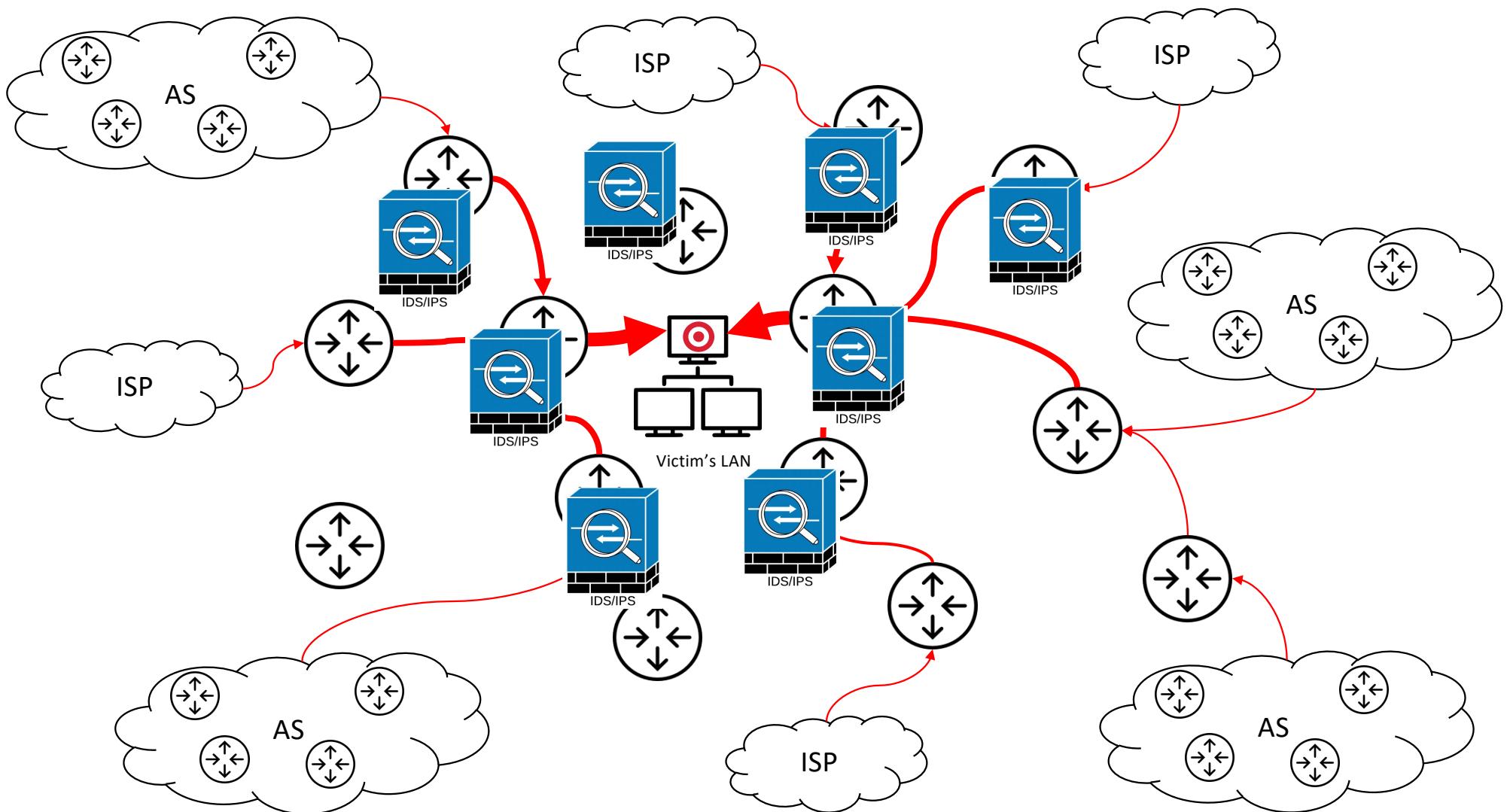
Badis HAMMI

100

Botnets/DDoS: mitigation: at the source level



Botnets/DDoS: mitigation: at the intermediary



Intrusion Detection/Prevention Systems



Checking your browser before accessing manamoa.net.

This process is automatic. Your browser will redirect to your requested content shortly.

Please allow up to 5 seconds...

DDoS protection by [Cloudflare](#)

Ray ID: 56fa48259e13a85b

Botnets/DDoS: mitigation/research perspectives

[François et al, 2012]

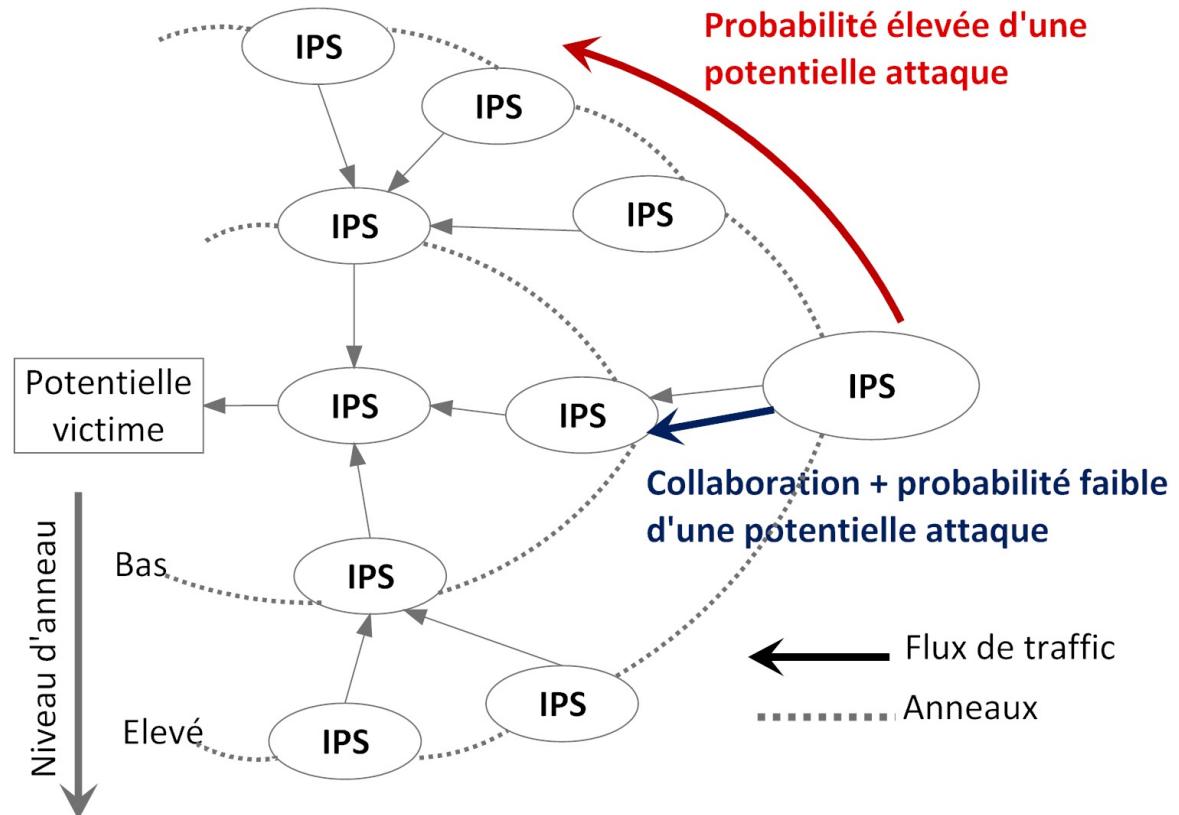


Problem:

- Cost
- Information sharing between devices for collaboration

Solution:

- The use of blockchain for information sharing



Network Security: Security by design

- Private key generation
- DMZ
- Hardware Security Module (HSM)
- Read only mode



Cisco.com