Utilisation malveillante des suivis de connexions

Éric Leblond

OISF

SSTIC 2012

Eric Leblond (Regit)

- Spécialiste de la sécurité des réseaux
- Utilisateur et développeur de Logiciel Libre
- Créateur du projet NuFW (maintenant ufwi), co-fondateur d'EdenWall
- Développeur Netfilter :
  - Ulogd2: démon de journalisation de Netfilter
  - Contributions diverses:
    - Bibliothèques NFQUEUE et dépendances.
    - Travail sur le journalisation.
- Actuellement :
  - Consultant indépendant en sécurité
  - Développeur financé de l'IDS/IPS Suricata

Netfilter

Définition

C’est le système de filtrage de paquets au sein des Linux 2.4.x à 3.x.

Netfilter

Définition

C'est le système de filtrage de paquets au sein des Linux 2.4.x à 3.x.

Fonctionnalités

- Filtrage des paquets à état et sans état (pour IPv4 et IPv6).
- Traduction d'adresse et de port.
- Des couches d'API pour permettre des extensions tierces.

Netfilter

Définition

C'est le système de filtrage de paquets au sein des Linux 2.4.x à 3.x.

Fonctionnalités

- Filtrage des paquets à état et sans état (pour IPv4 et IPv6).
- Traduction d'adresse et de port.
- Des couches d'API pour permettre des extensions tierces.

Iptables

- Utilitaire en ligne de commande gérant les règles de filtrage.
- Il donne accès à toutes les fonctions de Netfilter.
- Deux binaires : iptables pour IPv4, ip6tables pour IPv6.

```
iptables -A FORWARD -p tcp --syn --dport 80 \
         -m connlimit --connlimit-above 2 -j REJECT
```

Le suivi d'état de Netfilter

- Netfilter maintient une liste des connexions actives.
- La connexion à laquelle appartient un paquet est recherchée dans cette liste (le *"conntrack"*).
- Chaque paquet est marqué avec un des états suivants :
  - NEW
  - ESTABLISHED
  - INVALID
- Cet état peut-être utilisé pour décider du sort du paquet :

```
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
iptables -A FORWARD -m state --state NEW -p tcp --dport 80 -j ACCEPT
```

Application Level Gateway

Cas des protocoles non-linéaires

On peut trouver des protocoles comme FTP ou SIP:

- Leur fonctionnement est basé sur un canal de signalisation
- qui est utilisé pour échanger des paramètres de connexions dynamiques.

Application Level Gateway

Cas des protocoles non-linéaires

On peut trouver des protocoles comme FTP ou SIP:

- Leur fonctionnement est basé sur un canal de signalisation
- qui est utilisé pour échanger des paramètres de connexions dynamiques.

Application Level Gateway (ALG)

- Les ALGs cherchent dans le trafic des messages de commandes.
- Elles extraient les informations sur les connexions attendues.
- Une *expectation* est créée:
  - Elle inclut les informations sur les connexions potentielles.
  - Elle est associée à un timeout.
- Une nouvelle connexion correspondant à une *expectation* peut être acceptée.

L'exemple de FTP

FTP client

```
Logged in to ftp.lip6.fr.
ncftp / > ls
etc/        jussieu/    lip6/
```

Tcpdump

```
195.83.118.1.21 > 10.62.101.203.52994
195.83.118.1.21 > 10.62.101.203.52994
10.62.101.203.57636 > 195.83.118.1.51155
10.62.101.203.52994 > 195.83.118.1.21
195.83.118.1.51155 > 10.62.101.203.57636
```

L'exemple de FTP

FTP client

```
Logged in to ftp.lip6.fr.
ncftp / > ls
etc/        jussieu/    lip6/
```

Tcpdump

```
195.83.118.1.21 > 10.62.101.203.52994
195.83.118.1.21 > 10.62.101.203.52994
10.62.101.203.57636 > 195.83.118.1.51155
10.62.101.203.52994 > 195.83.118.1.21
195.83.118.1.51155 > 10.62.101.203.57636
```

Protocol

```
C: PASV
S: 227 Entering Passive Mode (195,83,118,1,199,211)
C: MLSD
S: 150 Opening ASCII mode data connection for 'MLSD'.
S: 226 MLSD complete.
C: QUIT
```

L'exemple de FTP

FTP client

```
Logged in to ftp.lip6.fr.
ncftp / > ls
etc/        jussieu/    lip6/
```

Tcpdump

```
195.83.118.1.21 > 10.62.101.203.52994
195.83.118.1.21 > 10.62.101.203.52994
10.62.101.203.57636 > 195.83.118.1.51155
10.62.101.203.52994 > 195.83.118.1.21
195.83.118.1.51155 > 10.62.101.203.57636
```

Protocol

```
C: PASV
S: 227 Entering Passive Mode (195,83,118,1,199,211)
C: MLSD
S: 150 Opening ASCII mode data connection for 'MLSD'.
S: 226 MLSD complete.
C: QUIT
```

Netfilter

```
# conntrack -E expect
    [NEW] 300 proto=6 src=10.62.101.203 dst=195.83.118.1 sport=0 dport=51155
[DESTROY] 300 proto=6 src=10.62.101.203 dst=195.83.118.1 sport=0 dport=51155
```

Details de l'implementation dans Netfilter

Les ALGs dans Netfilter

- ALGs sont appelées *Helpers*.
- Chaque protocole est implémenté comme un module noyau.
- Des options au chargement peuvent être utilisées pour configurer le *helper*.
- Une configuration fine peut être réalisée grâce à la cible *CT* de *iptables*.

Details de l'implementation dans Netfilter

Les ALGs dans Netfilter

- ALGs sont appelées *Helpers*.
- Chaque protocole est implémenté comme un module noyau.
- Des options au chargement peuvent être utilisées pour configurer le *helper*.
- Une configuration fine peut être réalisée grâce à la cible *CT* de *iptables*.

Liste des modules présents dans un noyau Linux Vanilla

amanda	pptp	broadcast	proto_dccp
ftp	proto_gre	h323	proto_sctp
ipv4	proto_udplite	ipv6	sane
irc	sip	netbios_ns	snmp
tftp			

Les *expectations* dans Netfilter

La table des *expectations*

- Les *expectations* sont stockées dans une table spécifique.
  - Elle est comparable à la table de suivi de connexions.
  - Seul un tuple est utilisé.
  - Un court timeout est posé sur chaque entrée.
- Une entrée est détruite quand elle matche un paquet.
- En réponse, une nouvelle connexion est crée.
- Elle est *RELATED* à la connexion de signalisation.

Les *expectations* dans Netfilter

La table des *expectations*

- Les *expectations* sont stockées dans une table spécifique.
  - Elle est comparable à la table de suivi de connexions.
  - Seul un tuple est utilisé.
  - Un court timeout est posé sur chaque entrée.
- Une entrée est détruite quand elle matche un paquet.
- En réponse, une nouvelle connexion est crée.
- Elle est *RELATED* à la connexion de signalisation.

Accepté les connexions RELATED

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

J'utilise des *helpers*

- Que se passe-t-il si je charge un *helper* ?

J'utilise des *helpers*

- Que se passe-t-il si je charge un *helper* ?
- Est-ce qu'un utilisateur peut envoyer des messages malveillants pour arriver à passer à travers le pare-feu ?

J'utilise des *helpers*

- Que se passe-t-il si je charge un *helper* ?
- Est-ce qu'un utilisateur peut envoyer des messages malveillants pour arriver à passer à travers le pare-feu ?
- Est-ce que les *helpers* transforment mon pare-feu en openbar ?

J'utilise des *helpers*

- Que se passe-t-il si je charge un *helper* ?
- Est-ce qu'un utilisateur peut envoyer des messages malveillants pour arriver à passer à travers le pare-feu ?
- Est-ce que les *helpers* transforment mon pare-feu en openbar ?

J'utilise des *helpers*

- Que se passe-t-il si je charge un *helper* ?
- Est-ce qu'un utilisateur peut envoyer des messages malveillants pour arriver à passer à travers le pare-feu ?
- Est-ce que les *helpers* transforment mon pare-feu en openbar ?

- Une étude est nécessaire.
- Regardons de plus prêt les *helpers*.

Degré de liberté induit par les *helpers* Netfilter

Module	Source	Destination	Port Dest	Option
ftp	Fixed	In CMD	In CMD	loose = 1 (dflt)
ftp	**Full**	In CMD	In CMD	loose = 0
h323	Fixed	Fixed	In CMD	
irc	**Full**	Fixed	In CMD	
sip signalling	Fixed	Fixed	In CMD	sip_direct_signalling = 1 (dflt)
sip signalling	**Full**	In CMD	In CMD	sip_direct_signalling = 0

- Légende :
  - Fixed: La valeur provient de paramètres IP de la connexion de signalisation. La valeur ne peut donc être forgée.
  - In CMD: La valeur est calculée en analysant le contenu du message protocolaire et peut donc être forgée.
  - Full: La liberté est totale, toutes les valeurs sont acceptées.
- Les options sont spécifiques à Netfilter.
- Cependant les degrés de libertés sont semblables pour tous les pare-feu utilisant des ALGs.

Analyse globale

Des valeurs par défaut saines

- Les extensions dangereuses des protocoles sont désactivées.
- Si on étudie l'attaque d'un client contre un serveur :
  - Il est impossible d'ouvrir une connexion arbitraire sur serveur
  - Le niveau de sécurité est donc acceptable

Analyse globale

Des valeurs par défaut saines

- Les extensions dangereuses des protocoles sont désactivées.
- Si on étudie l'attaque d'un client contre un serveur :
  - Il est impossible d'ouvrir une connexion arbitraire sur serveur
  - Le niveau de sécurité est donc acceptable

Dans la limite des protocoles

- La sécurité des helpers s'arrête là ou commence l'utilisabilité du protocole.
- Le *helper* IRC est très sympathique.

Analyse de FTP

Si on suit la RFC (*loose* = 0).

- Un serveur FTP peut participer à l'initialisation d'une connexion depuis un client vers un autre serveur.
- Il peut donc ouvrir des connexions arbitraires à travers le pare-feu.

Analyse de FTP

Si on suit la RFC (*loose* = 0).

- Un serveur FTP peut participer à l'initialisation d'une connexion depuis un client vers un autre serveur.
- Il peut donc ouvrir des connexions arbitraires à travers le pare-feu.

Si on s'occupe de la sécurité (*loose* = 1).

- Les *Expectation* sont statiquement liées à l'adresse du serveur.
- Les connexions possibles sont acceptables.
- C'est la valeur par défaut.

Analyse d'IRC

La commande DCC

La commande DCC permet un transfert entre deux clients d'un même serveur.

- Il est impossible de connaître l'adresse source.
- Le port destination est fixé par le le client.

Analyse d'IRC

La commande DCC

La commande DCC permet un transfert entre deux clients d'un même serveur.

- Il est impossible de connaître l'adresse source.
- Le port destination est fixé par le le client.

Conséquences

- Permettre DCC revient donc à autoriser des connexions arbitraires depuis internet vers cette IP.
- L'ordinateur client a une liberté totale d'ouverture de connexions.

Analyse d'IRC

La commande DCC

La commande DCC permet un transfert entre deux clients d'un même serveur.

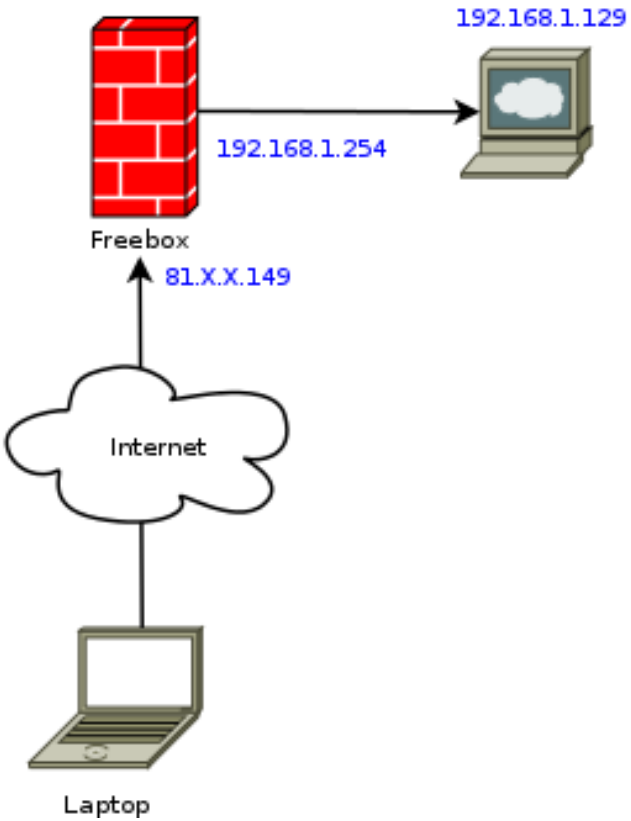
- Il est impossible de connaître l'adresse source.
- Le port destination est fixé par le le client.

Conséquences

- Permettre DCC revient donc à autoriser des connexions arbitraires depuis internet vers cette IP.
- L'ordinateur client a une liberté totale d'ouverture de connexions.

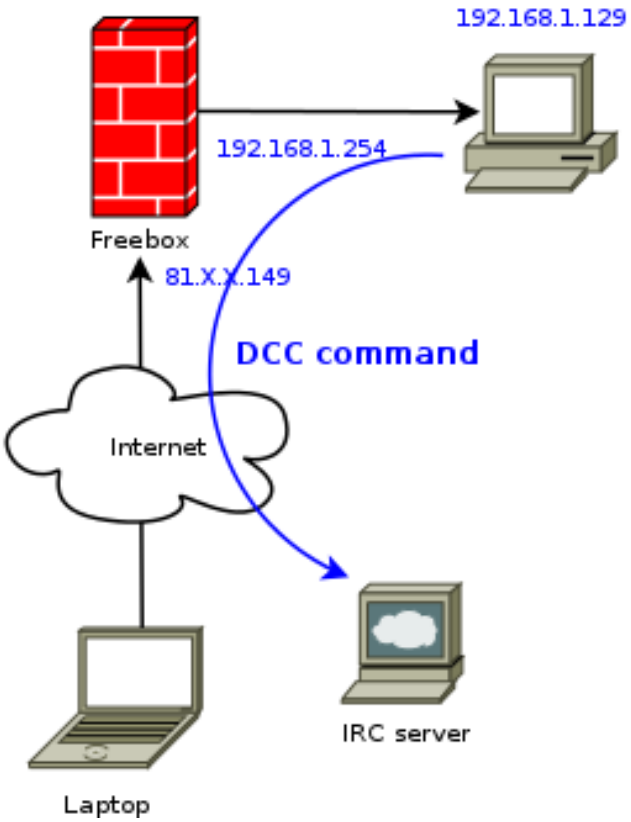
A mistake is simply another way of doing things.
(Katharine Graham)

Utilisation de la commande DCC



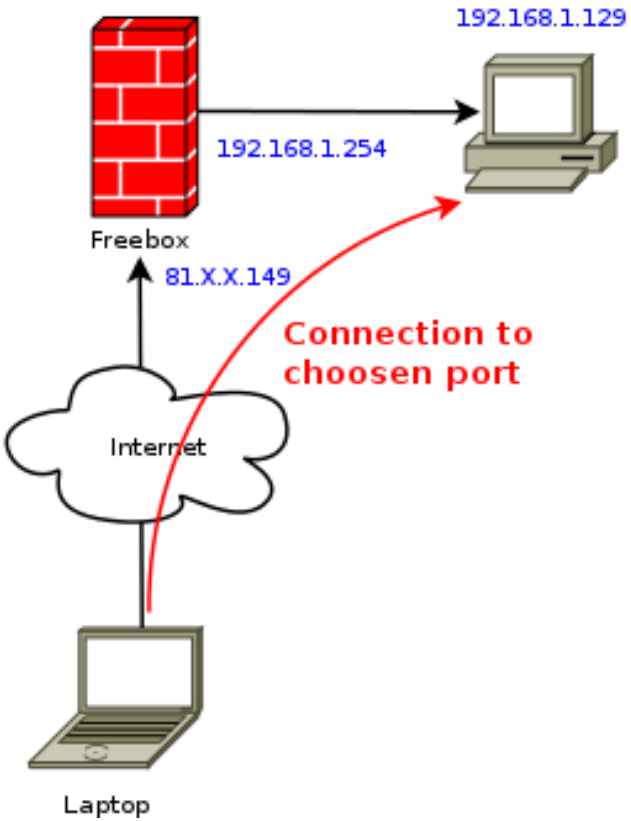
- Le client NATé derrière le pare-feu, le port *N* est fermé

Utilisation de la commande DCC



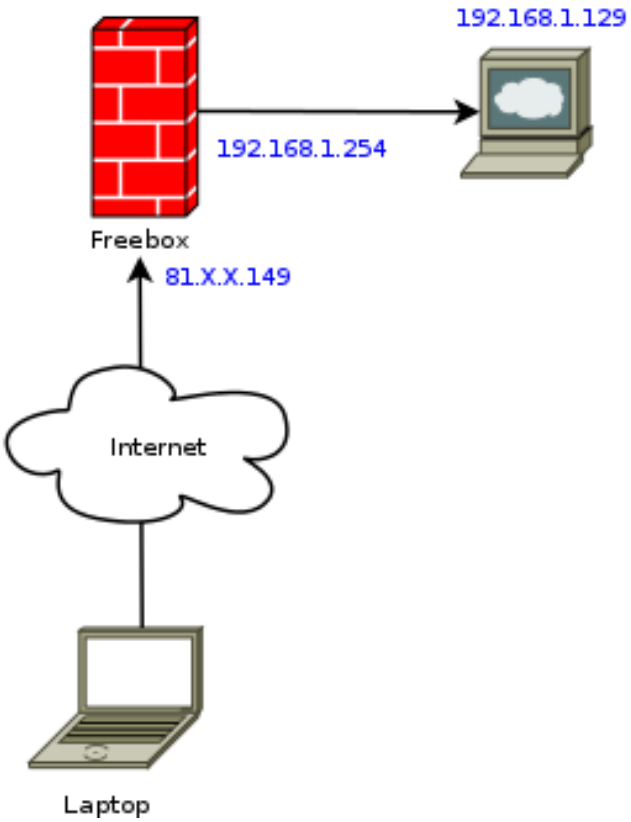
- Le client NATé derrière le pare-feu, le port *N* est fermé
- Le client envoie une commande DCC forgée vers un “serveur” IRC

Utilisation de la commande DCC



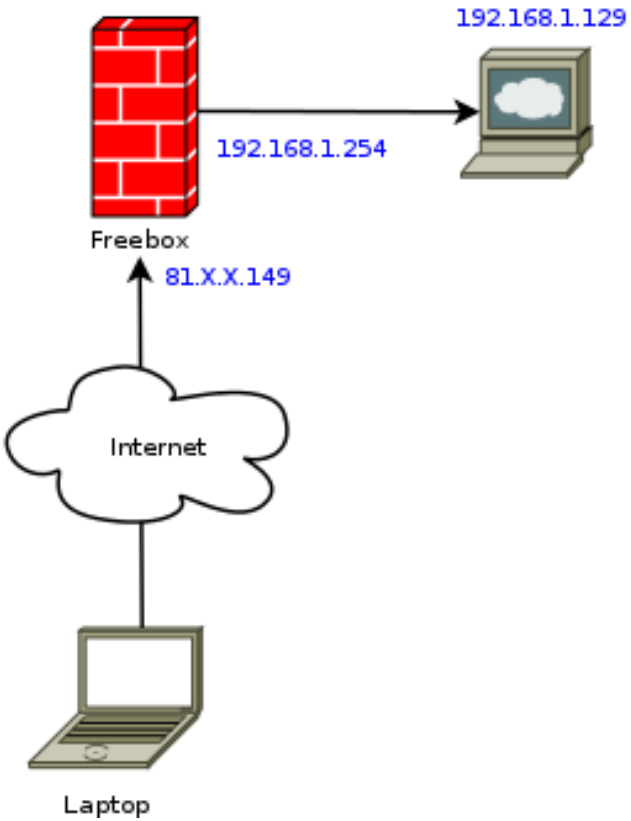
- Le client NATé derrière le pare-feu, le port *N* est fermé
- Le client envoie une commande DCC forgée vers un “serveur” IRC
- Le pare-feu crée une *expectation* et le client peut ouvrir une connexion

Démonstration de l'usage de DCC



Video

Démonstration de l'usage de DCC



Video

Connectons nous depuis Internet au port 6000 d'un client NATé.

Utilisation sûre des *helpers* Netfilter

Désactivation des *helpers* par défaut

Chargement du helper avec `ports=0` ou entrée de *proc* (Linux > 3.5):

```
modprobe nf_conntrack_ftp ports=0
```

Utilisation sûre des *helpers* Netfilter

Désactivation des *helpers* par défaut

Chargement du helper avec `ports=0` ou entrée de *proc* (Linux > 3.5):

```
modprobe nf_conntrack_ftp ports=0
```

Utilisation de la cible CT

Activation du *helper* pour autoriser de manière explicite le trafic relatif :

```
iptables -A PREROUTING -t raw -p tcp --dport 21 \\
  -d $MY_FTP_SERVER -j CT --helper ftp
iptables -A FORWARD -m conntrack --ctstate RELATED \\
         -m helper --helper ftp -d $MY_FTP_SERVER \\
         -p tcp --dport 1024: -j ACCEPT
```

Utilisation sûre des *helpers* Netfilter

Désactivation des *helpers* par défaut

Chargement du helper avec `ports=0` ou entrée de *proc* (Linux > 3.5):

```
modprobe nf_conntrack_ftp ports=0
```

Utilisation de la cible CT

Activation du *helper* pour autoriser de manière explicite le trafic relatif :

```
iptables -A PREROUTING -t raw -p tcp --dport 21 \\
  -d $MY_FTP_SERVER -j CT --helper ftp
iptables -A FORWARD -m conntrack --ctstate RELATED \\
         -m helper --helper ftp -d $MY_FTP_SERVER \\
         -p tcp --dport 1024: -j ACCEPT
```

Plus d'informations

Voir `https://home.regit.org/netfilter-en/secure-use-of-helpers/`

Objectif

- Est-il possible en tant que client de déclencher des comportements non souhaités?
  - Peut-on ouvrir des flux "arbitraires" sur un pare-feu ?
  - Peut-on ouvrir des ports sur un serveur ?
  - Peut-on accéder ainsi à des services mal protégés?
    - Comme une base de données interne
    - Comme des services vulnérables

Objectif

- Est-il possible en tant que client de déclencher des comportements non souhaités?
  - Peut-on ouvrir des flux "arbitraires" sur un pare-feu ?
  - Peut-on ouvrir des ports sur un serveur ?
  - Peut-on accéder ainsi à des services mal protégés?
    - Comme une base de données interne
    - Comme des services vulnérables
- L'étude des *helpers* a montré que c'était impossible sans malice :
  - Les possibilités du client sont toujours limitées.
  - Les extensions dangereuses sont désactivées par défaut.

Objectif

- Est-il possible en tant que client de déclencher des comportements non souhaités?
  - Peut-on ouvrir des flux "arbitraires" sur un pare-feu ?
  - Peut-on ouvrir des ports sur un serveur ?
  - Peut-on accéder ainsi à des services mal protégés?
    - Comme une base de données interne
    - Comme des services vulnérables
- L'étude des *helpers* a montré que c'était impossible sans malice :
  - Les possibilités du client sont toujours limitées.
  - Les extensions dangereuses sont désactivées par défaut.
- Une approche alternative doit donc être trouvée.

Idée basique

- Seul le serveur peut envoyer des messages intéressants.

Idée basique

- Seul le serveur peut envoyer des messages intéressants.
- Nous pouvons forcer le serveur à envoyer un message forgé.

Idée basique

- Seul le serveur peut envoyer des messages intéressants.
- Nous pouvons forcer le serveur à envoyer un message forgé.
- Nous ne pouvons forcer le serveur à le faire.

Idée basique

- Seul le serveur peut envoyer des messages intéressants.
- Nous pouvons forcer le serveur à envoyer un message forgé.
- Nous ne pouvons forcer le serveur à le faire.
- Il est possible sous certaines conditions d'envoyer un message pour le serveur.
  - Un ordinateur peut forger un paquet IP arbitraire
  - et l'envoyer à la passerelle
  - si l'ordinateur est sur le même réseau Ethernet que la passerelle.

Idée basique

- Seul le serveur peut envoyer des messages intéressants.
- Nous pouvons forcer le serveur à envoyer un message forgé.
- Nous ne pouvons forcer le serveur à le faire.
- Il est possible sous certaines conditions d'envoyer un message pour le serveur.
  - Un ordinateur peut forger un paquet IP arbitraire
  - et l'envoyer à la passerelle
  - si l'ordinateur est sur le même réseau Ethernet que la passerelle.
- Un attaquant **sur un réseau directement connecté** peut envoyer des paquets :

Idée basique

- Seul le serveur peut envoyer des messages intéressants.
- Nous pouvons forcer le serveur à envoyer un message forgé.
- Nous ne pouvons forcer le serveur à le faire.
- Il est possible sous certaines conditions d'envoyer un message pour le serveur.
  - Un ordinateur peut forger un paquet IP arbitraire
  - et l'envoyer à la passerelle
  - si l'ordinateur est sur le même réseau Ethernet que la passerelle.
- Un attaquant **sur un réseau directement connecté** peut envoyer des paquets :
  - à l'adresse Ethernet du pare-feu

Idée basique

- Seul le serveur peut envoyer des messages intéressants.
- Nous pouvons forcer le serveur à envoyer un message forgé.
- Nous ne pouvons forcer le serveur à le faire.
- Il est possible sous certaines conditions d'envoyer un message pour le serveur.
  - Un ordinateur peut forger un paquet IP arbitraire
  - et l'envoyer à la passerelle
  - si l'ordinateur est sur le même réseau Ethernet que la passerelle.
- Un attaquant **sur un réseau directement connecté** peut envoyer des paquets :
  - à l'adresse Ethernet du pare-feu
  - avec comme source l'adresse IP du serveur

Idée basique

- Seul le serveur peut envoyer des messages intéressants.
- Nous pouvons forcer le serveur à envoyer un message forgé.
- Nous ne pouvons forcer le serveur à le faire.
- Il est possible sous certaines conditions d'envoyer un message pour le serveur.
  - Un ordinateur peut forger un paquet IP arbitraire
  - et l'envoyer à la passerelle
  - si l'ordinateur est sur le même réseau Ethernet que la passerelle.
- Un attaquant **sur un réseau directement connecté** peut envoyer des paquets :
  - à l'adresse Ethernet du pare-feu
  - avec comme source l'adresse IP du serveur
- Essayons cette méthode.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.
3. Il récupère un paquet venant du serveur.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.
3. Il récupère un paquet venant du serveur.
   - Inverse source et destination au niveau Ethernet.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.
3. Il récupère un paquet venant du serveur.
   - Inverse source et destination au niveau Ethernet.
   - Modifie la charge du paquet en forgeant une commande serveur avec des paramètres choisis.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.
3. Il récupère un paquet venant du serveur.
   - Inverse source et destination au niveau Ethernet.
   - Modifie la charge du paquet en forgeant une commande serveur avec des paramètres choisis.
   - Incrémente l'identifiant IP.
   - Positionne le numéro de séquence TCP grâce à sa connaissance des paquets échangés.
   - Met à jour les sommes de contrôles et les longueurs.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.
3. Il récupère un paquet venant du serveur.
   - Inverse source et destination au niveau Ethernet.
   - Modifie la charge du paquet en forgeant une commande serveur avec des paramètres choisis.
   - Incrémente l'identifiant IP.
   - Positionne le numéro de séquence TCP grâce à sa connaissance des paquets échangés.
   - Met à jour les sommes de contrôles et les longueurs.
4. L'attaquant envoie alors ce paquet forgé sur le réseau.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.
3. Il récupère un paquet venant du serveur.
   - Inverse source et destination au niveau Ethernet.
   - Modifie la charge du paquet en forgeant une commande serveur avec des paramètres choisis.
   - Incrémente l'identifiant IP.
   - Positionne le numéro de séquence TCP grâce à sa connaissance des paquets échangés.
   - Met à jour les sommes de contrôles et les longueurs.
4. L'attaquant envoie alors ce paquet forgé sur le réseau.
5. Le pare-feu traite la requête forgée qui est valide au niveau IP et au niveau Ethernet

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.
3. Il récupère un paquet venant du serveur.
   - Inverse source et destination au niveau Ethernet.
   - Modifie la charge du paquet en forgeant une commande serveur avec des paramètres choisis.
   - Incrémente l'identifiant IP.
   - Positionne le numéro de séquence TCP grâce à sa connaissance des paquets échangés.
   - Met à jour les sommes de contrôles et les longueurs.
4. L'attaquant envoie alors ce paquet forgé sur le réseau.
5. Le pare-feu traite la requête forgée qui est valide au niveau IP et au niveau Ethernet
6. Le pare-feu crée une *expectation* avec les paramètres **"donnés"** par le serveur.

Description de l'attaque

1. L'attaquant ouvre une connexion réseau légitime pour un protocole donné.
2. Il sniffe le trafic réseau pour ce protocole.
3. Il récupère un paquet venant du serveur.
   - Inverse source et destination au niveau Ethernet.
   - Modifie la charge du paquet en forgeant une commande serveur avec des paramètres choisis.
   - Incrémente l'identifiant IP.
   - Positionne le numéro de séquence TCP grâce à sa connaissance des paquets échangés.
   - Met à jour les sommes de contrôles et les longueurs.
4. L'attaquant envoie alors ce paquet forgé sur le réseau.
5. Le pare-feu traite la requête forgée qui est valide au niveau IP et au niveau Ethernet
6. Le pare-feu crée une *expectation* avec les paramètres "donnés" par le serveur.
7. L'attaquant peut alors ouvrir une connexion avec les paramètres qu'il a choisi.

Étude du protocole FTP

Connexion dynamique

Des connexions dynamiques peuvent être ouvertes vers le serveur en mode passif :

Étude du protocole FTP

Connexion dynamique

Des connexions dynamiques peuvent être ouvertes vers le serveur en mode passif :

- Le serveur envoie un message au client pour indiquer quelle IP et quel port utiliser.
- Le client se connecte alors avec les paramètres fournis.

Étude du protocole FTP

Connexion dynamique

Des connexions dynamiques peuvent être ouvertes vers le serveur en mode passif :

- Le serveur envoie un message au client pour indiquer quelle IP et quel port utiliser.
- Le client se connecte alors avec les paramètres fournis.

Cas d'IPv4

- Demande d'une connexion cliente à 192.168.2.2 sur le port 3306:
  `227 Entering Passive Mode (192,168,2,2,12,234)\r\n`
- Le format est simple (si on sait que $12 * 256 + 334 = 3306$).

Étude du protocole FTP

Connexion dynamique

Des connexions dynamiques peuvent être ouvertes vers le serveur en mode passif :

- Le serveur envoie un message au client pour indiquer quelle IP et quel port utiliser.
- Le client se connecte alors avec les paramètres fournis.

Cas d'IPv4

- Demande d'une connexion cliente à 192.168.2.2 sur le port 3306:

  `227 Entering Passive Mode (192,168,2,2,12,234)\r\n`

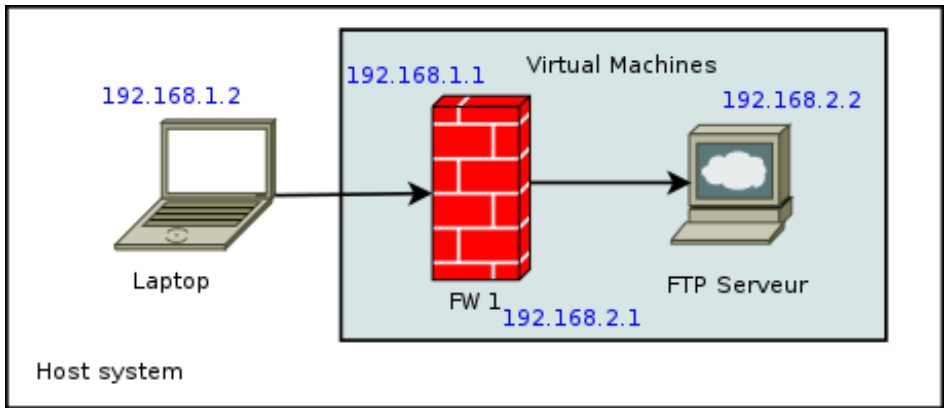
- Le format est simple (si on sait que $12 * 256 + 334 = 3306$).

Cas d'IPv6

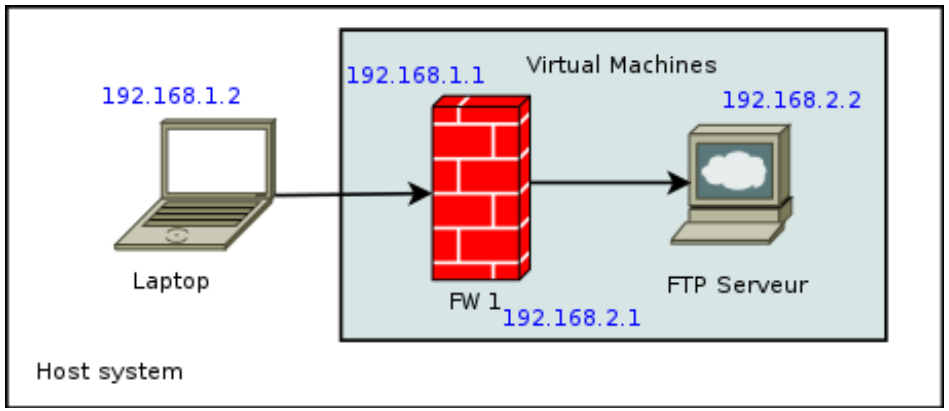
- Demande d'une connexion cliente sur le port 3306:

  `229 Extended Passive Mode OK (|||3306|)\r\n`

Essai sur Netfilter



Video



Video

Prenons un pare-feu avec une politique de filtrage autorisant seulement le port 21 et ouvrons une connexion vers le port 22 du serveur FTP.

Violation de la politique de sécurité

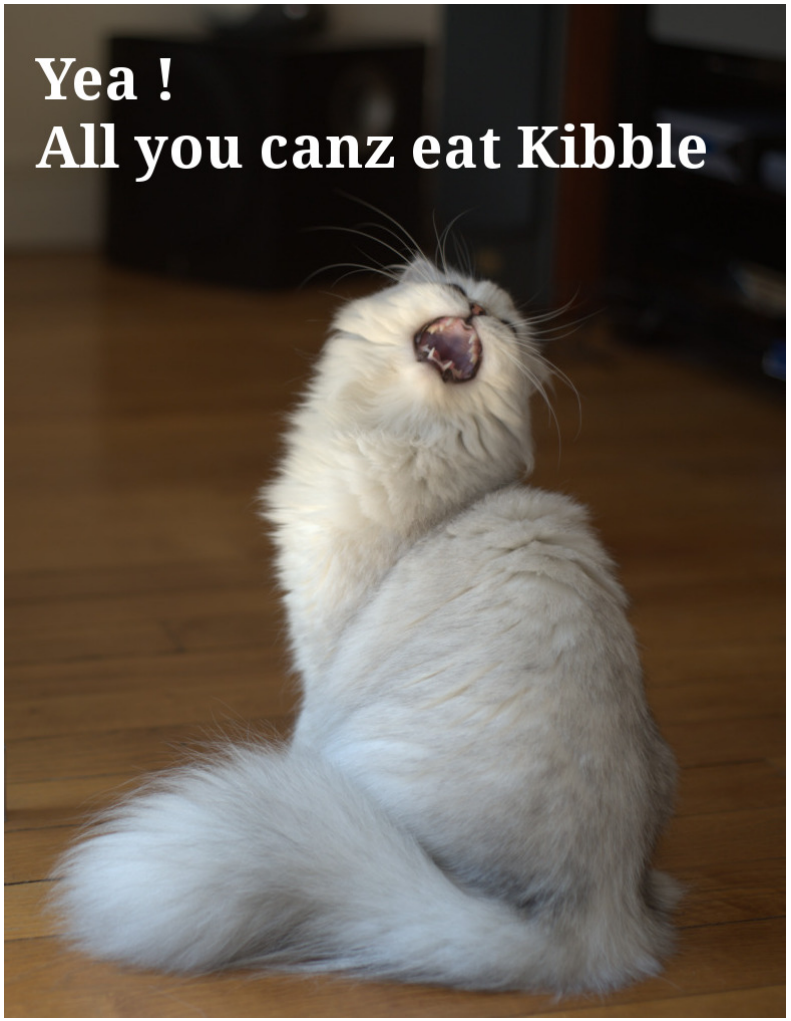
- Nous sommes parvenus à ouvrir une connexion sur le port 22.

Violation de la politique de sécurité

- Nous sommes parvenus à ouvrir une connexion sur le port 22.
- Avec une politique de filtrage ne le permettant pas.

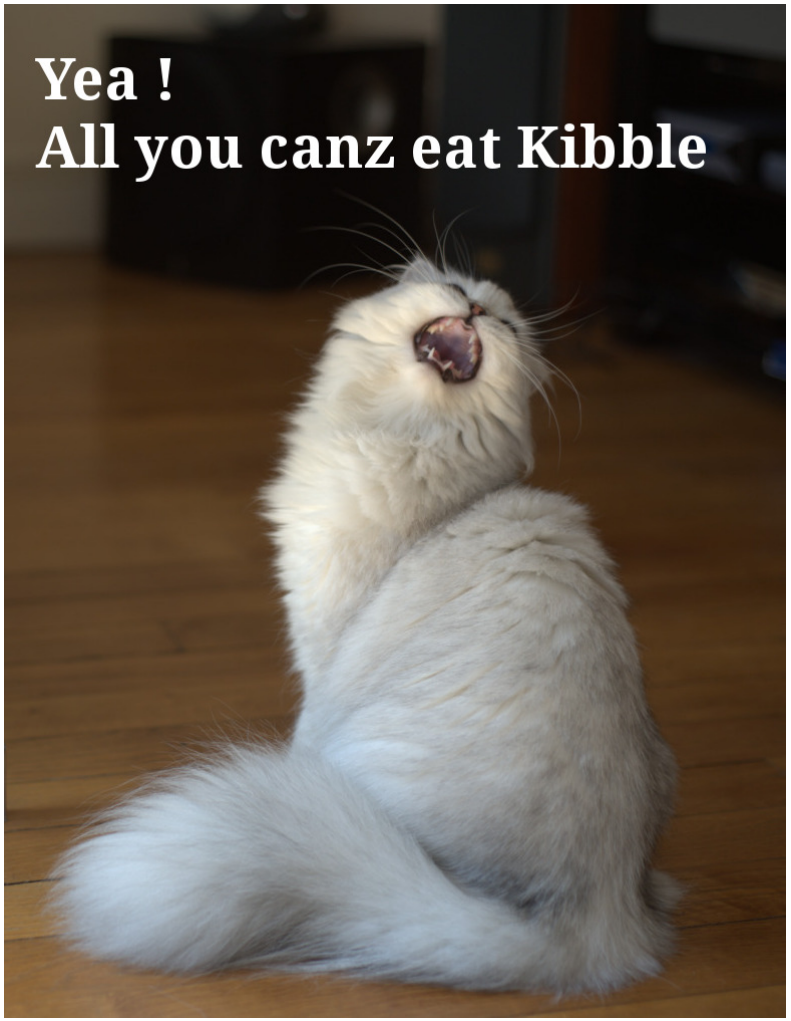
Violation de la politique de sécurité

- Nous sommes parvenus à ouvrir une connexion sur le port 22.
- Avec une politique de filtrage ne le permettant pas.



Violation de la politique de sécurité

- Nous sommes parvenus à ouvrir une connexion sur le port 22.
- Avec une politique de filtrage ne le permettant pas.
- Tout doux, chaton, tout doux!



Contre-mesures

- L'anti-spoofing est suffisant pour bloquer l'attaque.
- *Reverse path filtering* est notre ami:
  - Accepter seulement les paquet arrivant sur une interface si on a une route vers cette source passant par cette interface.
  - Le noyau ne traitera alors pas le paquet d'attaque.
- C'est si facile d'être protégé?

Contre-mesures

- L'anti-spoofing est suffisant pour bloquer l'attaque.
- *Reverse path filtering* est notre ami:
  - Accepter seulement les paquet arrivant sur une interface si on a une route vers cette source passant par cette interface.
  - Le noyau ne traitera alors pas le paquet d'attaque.
- C'est si facile d'être protégé? **Oui**

Contre-mesures

- L'anti-spoofing est suffisant pour bloquer l'attaque.
- *Reverse path filtering* est notre ami:
  - Accepter seulement les paquet arrivant sur une interface si on a une route vers cette source passant par cette interface.
  - Le noyau ne traitera alors pas le paquet d'attaque.
- C'est si facile d'être protégé? Oui
- Mais il reste quelques surprises.

Autres protocoles

IRC

- Le *helper* IRC donne déjà tous les pouvoirs au client.
- Mais peut on agir contre le client ?

Autres protocoles

IRC

- Le *helper* IRC donne déjà tous les pouvoirs au client.
- Mais peut on agir contre le client ?
- La même technique s'applique avec les conditions suivantes :
  - Attaquant et client séparé par le pare-feu.
  - L'attaquant est sur un réseau directement connecté sur un pare-feu.
  - Le trafic IRC peut être sniffé par l'attaquant (MITM ou serveur).

Autres protocoles

IRC

- Le *helper* IRC donne déjà tous les pouvoirs au client.
- Mais peut on agir contre le client ?
- La même technique s'applique avec les conditions suivantes :
  - Attaquant et client séparé par le pare-feu.
  - L'attaquant est sur un réseau directement connecté sur un pare-feu.
  - Le trafic IRC peut être sniffé par l'attaquant (MITM ou serveur).
- Plus contraignant, pas vraiment intéressant.

Autres protocoles

IRC

- Le *helper* IRC donne déjà tous les pouvoirs au client.
- Mais peut on agir contre le client ?
- La même technique s'applique avec les conditions suivantes :
  - Attaquant et client séparé par le pare-feu.
  - L'attaquant est sur un réseau directement connecté sur un pare-feu.
  - Le trafic IRC peut être sniffé par l'attaquant (MITM ou serveur).
- Plus contraignant, pas vraiment intéressant.

SIP

- Le serveur envoie des paramètres de port d'une façon similaire à FTP.
- La même attaque est possible.
- Seule le contenu de l'attaque doit être adapté.

Protection de Netfilter

- Il suffit d'utiliser la fonctionnalité **rp_filter**.
- Elle est disponible depuis le siècle dernier dans tout les noyaux Linux.

Protection de Netfilter

- Il suffit d'utiliser la fonctionnalité rp_filter.
- Elle est disponible depuis le siècle dernier dans tout les noyaux Linux.
- **Désactivée par défaut**.

Protection de Netfilter

- Il suffit d'utiliser la fonctionnalité rp_filter.
- Elle est disponible depuis le siècle dernier dans tout les noyaux Linux.
- Désactivée par défaut. Activée par tous les scripts de filtrage décents

Protection de Netfilter

- Il suffit d'utiliser la fonctionnalité rp_filter.
- Elle est disponible depuis le siècle dernier dans tout les noyaux Linux.
- Désactivée par défaut. Activée par tous les scripts de filtrage décents
- Pour l'activer:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

Protection de Netfilter

- Il suffit d'utiliser la fonctionnalité rp_filter.
- Elle est disponible depuis le siècle dernier dans tout les noyaux Linux.
- Désactivée par défaut. Activée par tous les scripts de filtrage décents
- Pour l'activer:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

- **Hummmmm**

Protection de Netfilter

- Il suffit d'utiliser la fonctionnalité rp_filter.
- Elle est disponible depuis le siècle dernier dans tout les noyaux Linux.
- Désactivée par défaut. Activée par tous les scripts de filtrage décents
- Pour l'activer:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

- **Hummmmm** et pour IPv6 ?

Protection de Netfilter

- Il suffit d'utiliser la fonctionnalité rp_filter.
- Elle est disponible depuis le siècle dernier dans tout les noyaux Linux.
- Désactivée par défaut. Activée par tous les scripts de filtrage décents
- Pour l'activer:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

- **Hummmmm** et pour IPv6 ?
- Trop facile, positionnons la valeur dans `/proc`:

```
echo "1"> /proc/sys/net/ipv6/conf/all/rp_filter
  /proc/sys/net/ipv6/conf/all/rp_filter: No such file or directory
```

Protection de Netfilter

- Il suffit d'utiliser la fonctionnalité rp_filter.
- Elle est disponible depuis le siècle dernier dans tout les noyaux Linux.
- Désactivée par défaut. Activée par tous les scripts de filtrage décents
- Pour l'activer:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

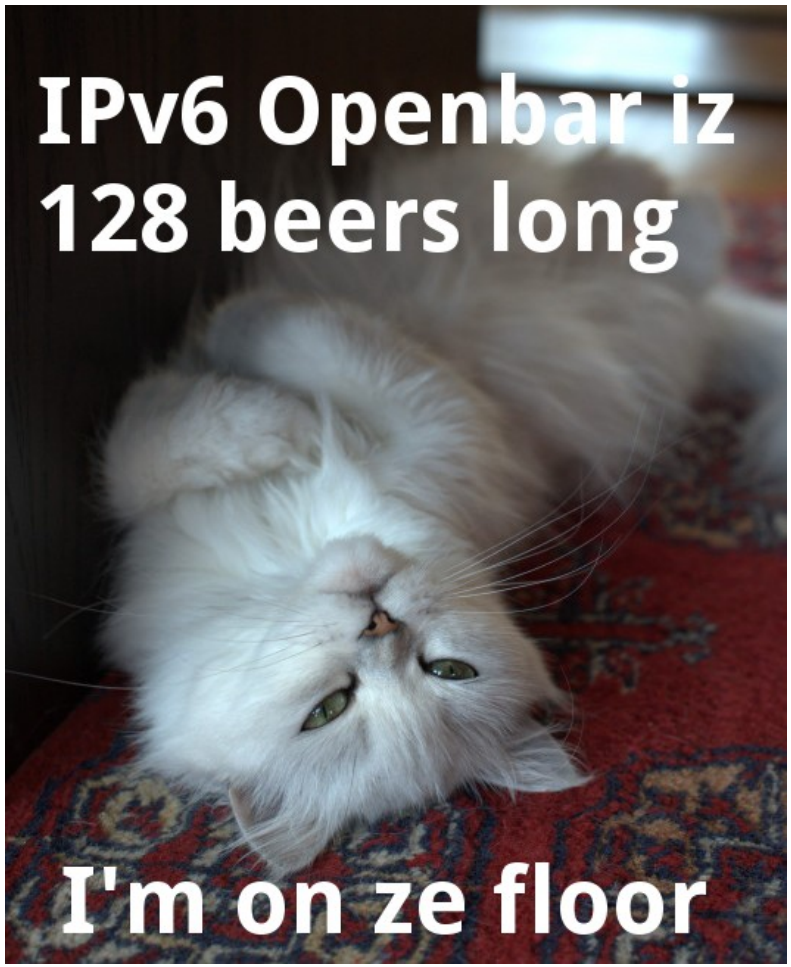
- **Hummmmm** et pour IPv6 ?
- Trop facile, positionnons la valeur dans `/proc`:

```
echo "1"> /proc/sys/net/ipv6/conf/all/rp_filter
  /proc/sys/net/ipv6/conf/all/rp_filter: No such file or directory
```

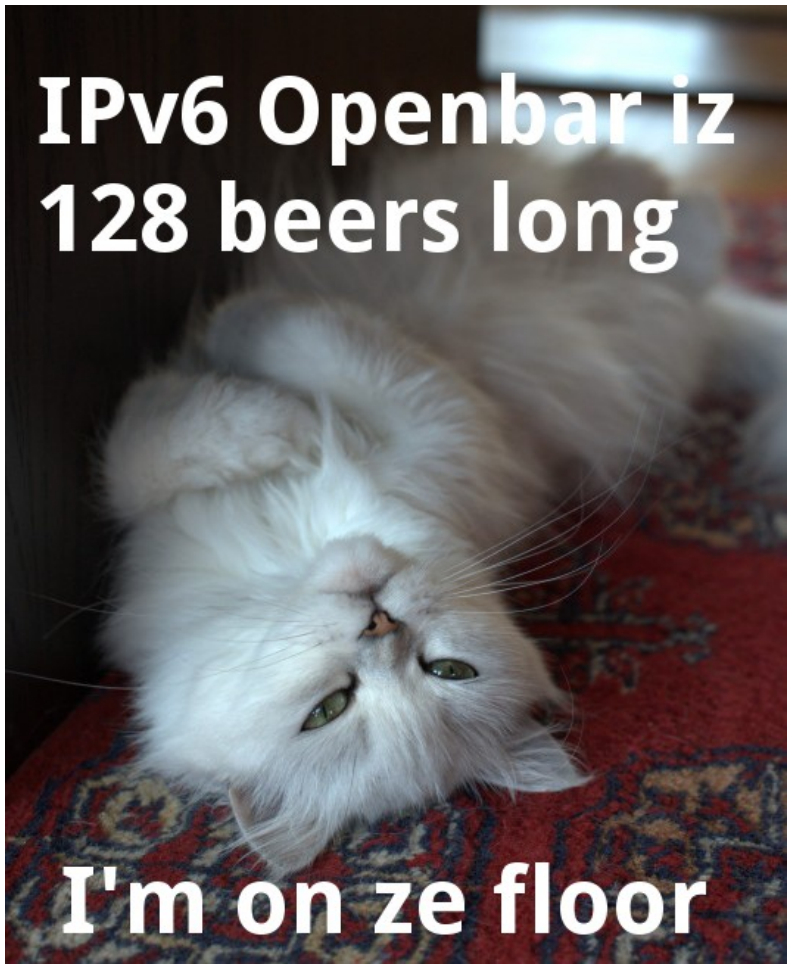
Okay, Houston, we've had a problem here.

(Jack Swigert)

Protection IPv6 pour Netfilter

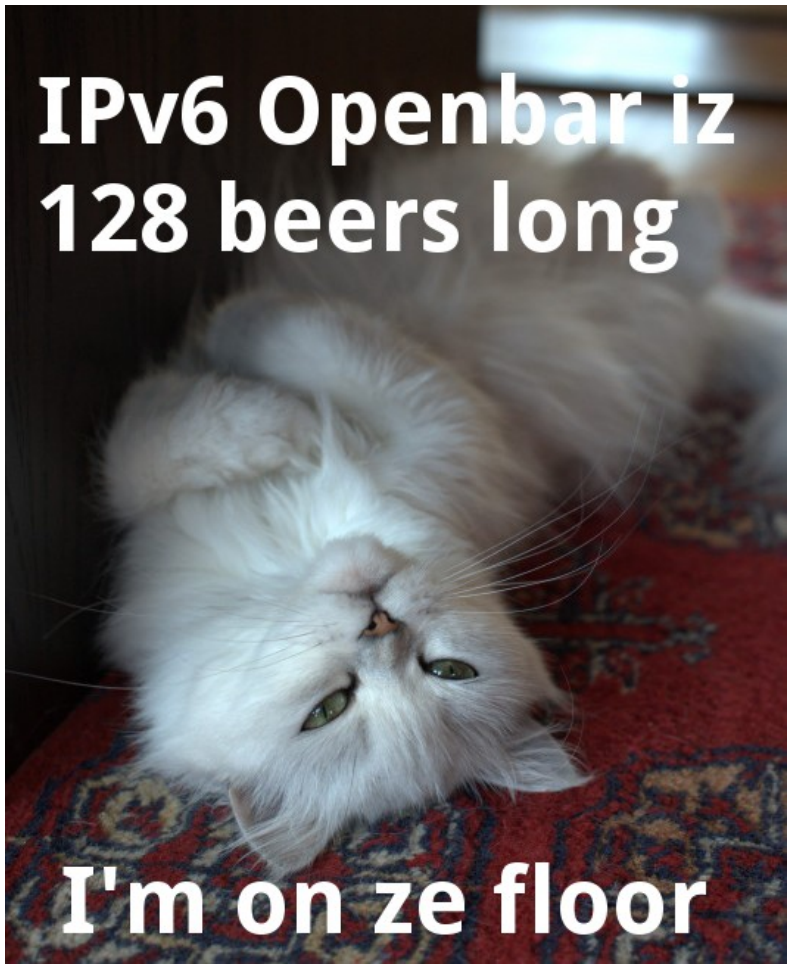


Protection IPv6 pour Netfilter



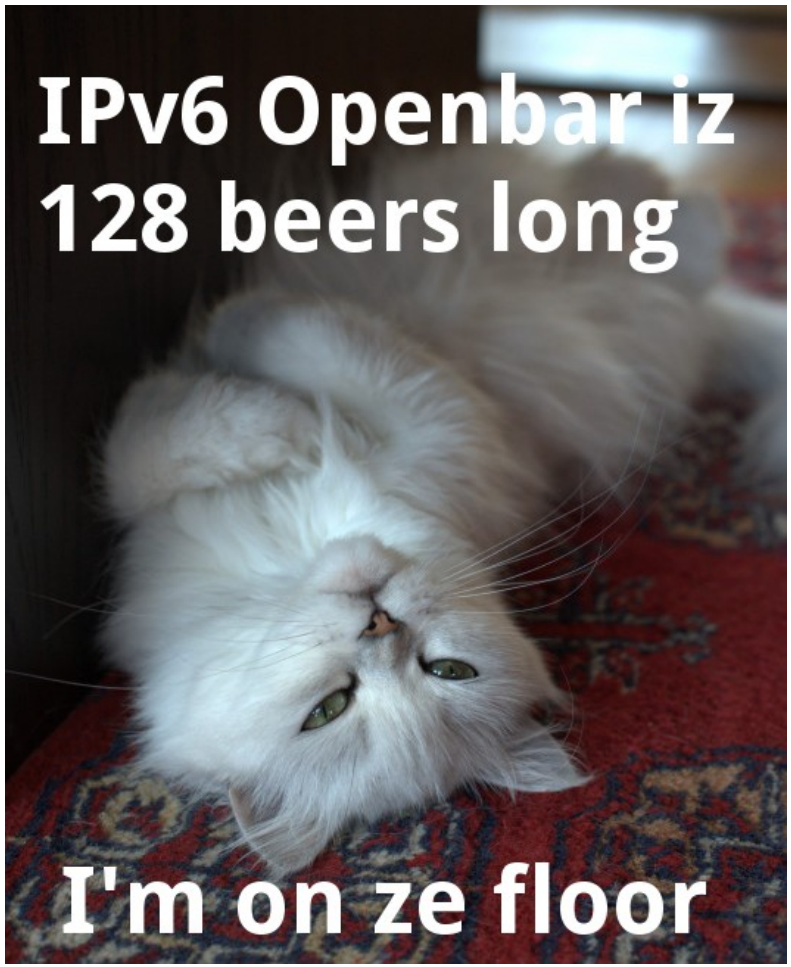
- Une configuration manuelle est nécessaire.

Protection IPv6 pour Netfilter



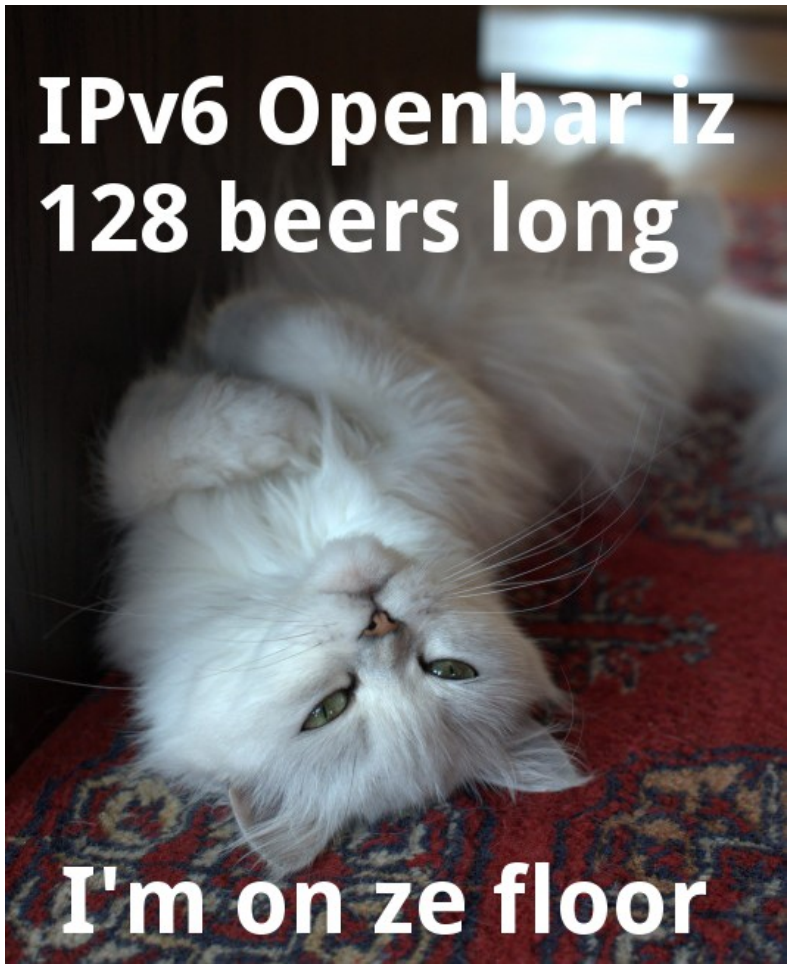
- Une configuration manuelle est nécessaire.
- Des règles *ip6tables* dédiées doivent être ajoutées.

Protection IPv6 pour Netfilter



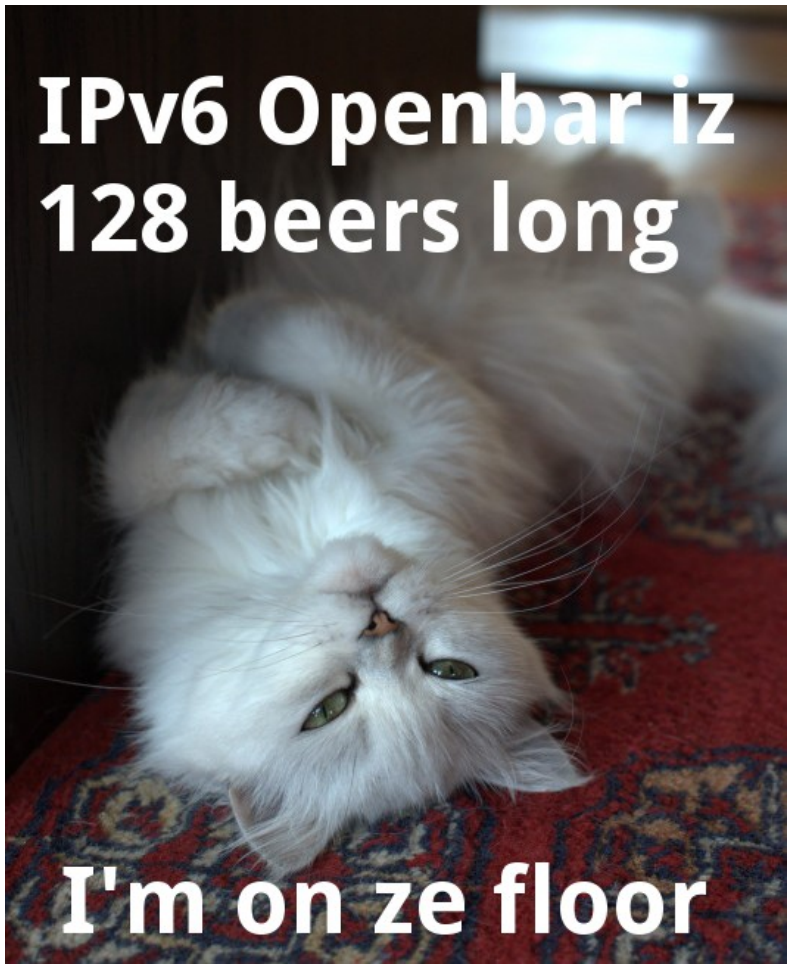
- Une configuration manuelle est nécessaire.
- Des règles *ip6tables* dédiées doivent être ajoutées.
- La topologie réseau doit être connue.

Protection IPv6 pour Netfilter



- Une configuration manuelle est nécessaire.
- Des règles *ip6tables* dédiées doivent être ajoutées.
- La topologie réseau doit être connue.
- Les bonnes implémentations utilisent déjà ce genre de règles.

Protection IPv6 pour Netfilter



- Une configuration manuelle est nécessaire.
- Des règles *ip6tables* dédiées doivent être ajoutées.
- La topologie réseau doit être connue.
- Les bonnes implémentations utilisent déjà ce genre de règles.
- Mais protègent elles contre l'attaque ?

Protection IPv6 pour Netfilter

Le mauvais exemple

```
ip6tables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -i $CLIENT_IFACE ! -s $CLIENT_NET -j DROP
```

- Le paquet appartient à une connexion établie.
- Accepté par la première règle, il n'est pas vu par l'anti-spoofing.

Protection IPv6 pour Netfilter

Le mauvais exemple

```
ip6tables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -i $CLIENT_IFACE ! -s $CLIENT_NET -j DROP
```

- Le paquet appartient à une connexion établie.
- Accepté par la première règle, il n'est pas vu par l'anti-spoofing.

La bonne méthode

```
ip6tables -A PREROUTING -t raw -i $CLIENT_IFACE ! -s $CLIENT_NET -j DROP
```

Le paquet est bloqué avant d'atteindre le suivi de connexions.

Protection IPv6 pour Netfilter

Le mauvais exemple

```
ip6tables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -i $CLIENT_IFACE ! -s $CLIENT_NET -j DROP
```

- Le paquet appartient à une connexion établie.
- Accepté par la première règle, il n'est pas vu par l'anti-spoofing.

La bonne méthode

```
ip6tables -A PREROUTING -t raw -i $CLIENT_IFACE ! -s $CLIENT_NET -j DROP
```

Le paquet est bloqué avant d'atteindre le suivi de connexions.

La nouvelle méthode (Linux > 3.3)

```
ip6tables -A PREROUTING -t raw -m rpfilter --invert -j DROP
```

Paramétrage de Checkpoint

Débutant Checkpoint

- Je n'ai pas lu la documentation.
- Pourquoi devrais je le faire ? Je travaille avec des pare-feu depuis des années.

Paramétrage de Checkpoint

Débutant Checkpoint

- Je n'ai pas lu la documentation.
- Pourquoi devrais je le faire ? Je travaille avec des pare-feu depuis des années.
- Qui n'a jamais fait pas pareil ?

Paramétrage de Checkpoint

Débutant Checkpoint

- Je n'ai pas lu la documentation.
- Pourquoi devrais je le faire ? Je travaille avec des pare-feu depuis des années.
- Qui n'a jamais fait pas pareil ?

Logiciel utilisé

- Version d'évaluation.
- Installation du minimum de fonctionnalités.

Paramétrage de Checkpoint

Débutant Checkpoint

- Je n'ai pas lu la documentation.
- Pourquoi devrais je le faire ? Je travaille avec des pare-feu depuis des années.
- Qui n'a jamais fait pas pareil ?

Logiciel utilisé

- Version d'évaluation.
- Installation du minimum de fonctionnalités.
- Installation par défaut **en suivant les wizards**.

Paramétrage de l'environnement

Paramétrage de l'environnement

- Créons une politique de filtrage avec une unique règle autorisant FTP ;

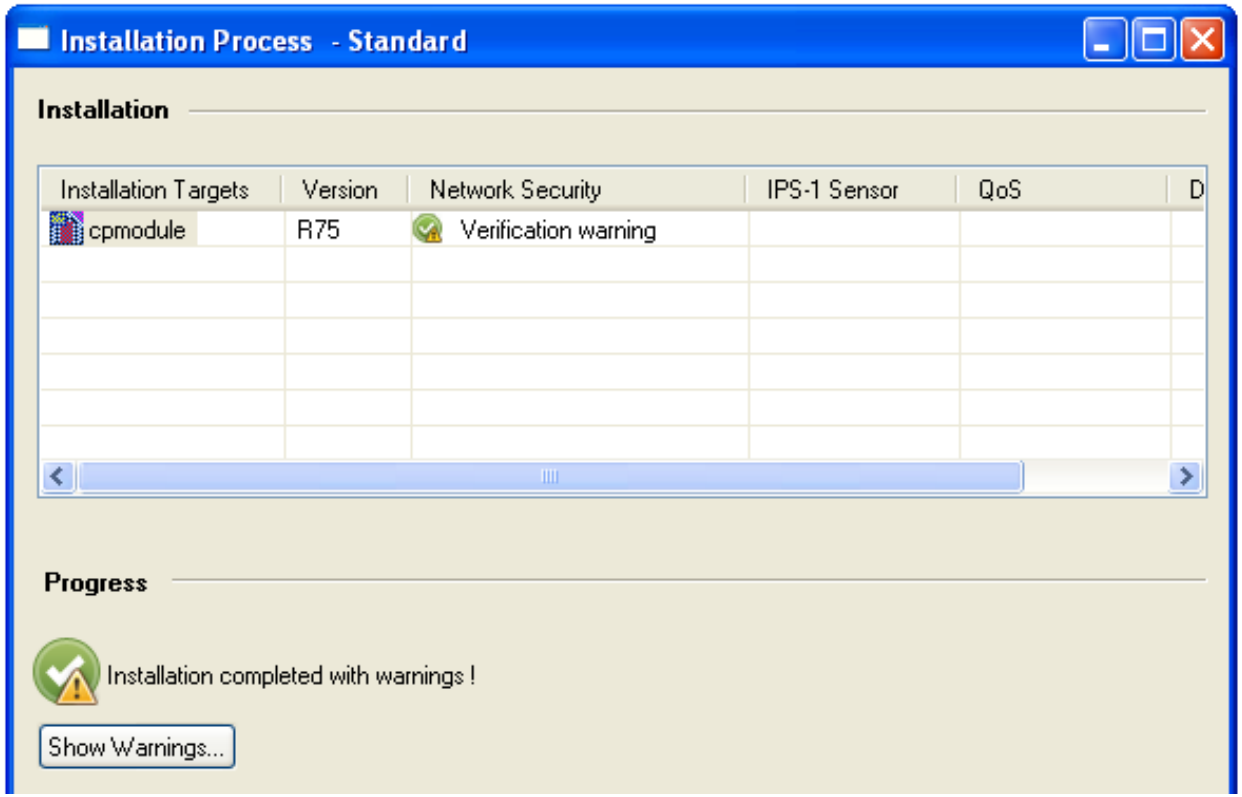
SOURCE	DESTINATION	VPN	SERVICE	ACTION	TRACK	INSTALL ON	TIME
* Any	* Any	* Any Traffic	TCP ftp	accept	- None	* Policy Targets	* Any

Paramétrage de l'environnement

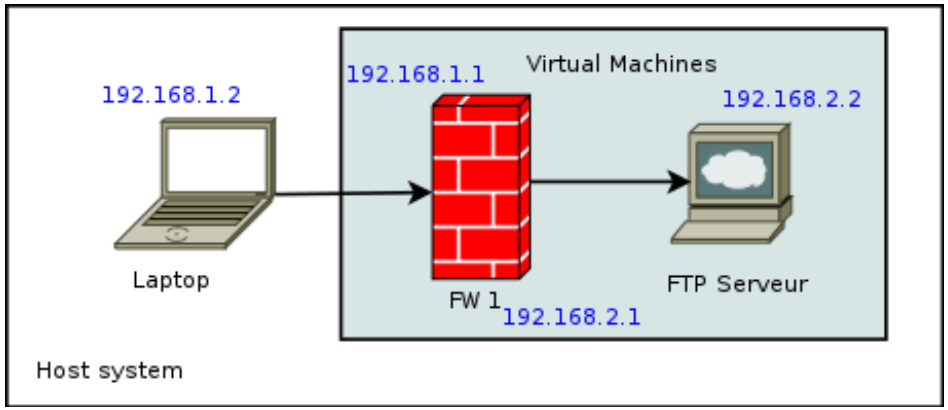
- Créons une politique de filtrage avec une unique règle autorisant FTP ;

SOURCE	DESTINATION	VPN	SERVICE	ACTION	TRACK	INSTALL ON	TIME
* Any	* Any	* Any Traffic	TCP ftp	accept	- None	* Policy Targets	* Any

- Et installons la politique de filtrage.

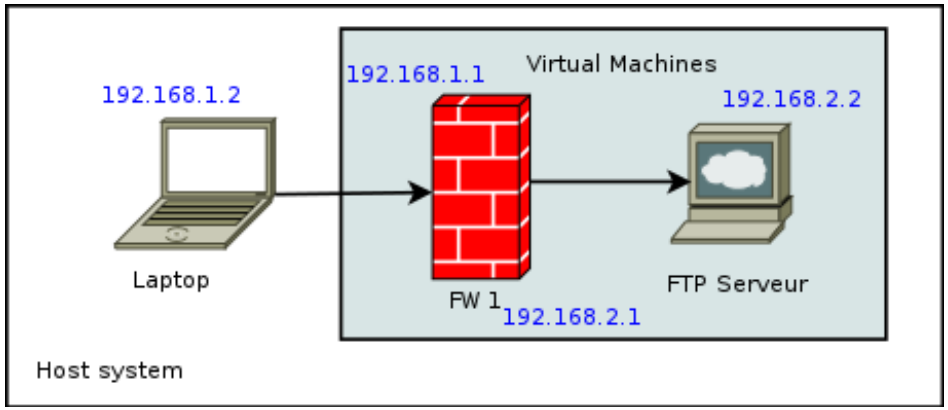


Démonstration



Video

Démonstration



Video

Essayons d'ouvrir une connexion vers le port 22.

Détournement de la politique

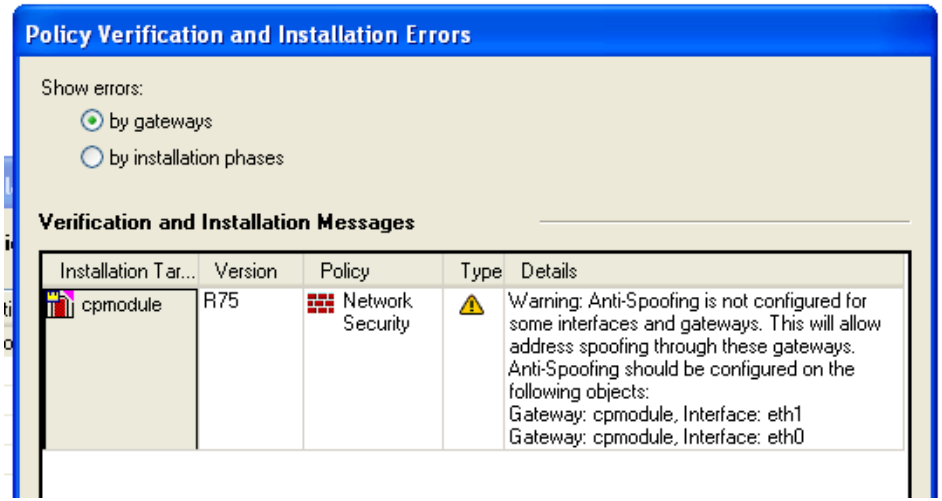
- Une connexion a été établie sur le port 22
- violant ainsi la politique de filtrage

Détournement de la politique

- Une connexion a été établie sur le port 22
- violant ainsi la politique de filtrage
- Mais la connexion a été bloqué après quelques paquets.

Détournement de la politique

- Une connexion a été établie sur le port 22
- violant ainsi la politique de filtrage
- Mais la connexion a été bloqué après quelques paquets.
- L'interface de Checkpoint affiche un avertissement à propos de l'anti-spoofing.



Où est le problème ?

Une réaction rapide de l'équipe de sécurité de Checkpoint.

Configuring anti-spoofing is a basic requirement.

Them

Are you planning some action regarding this issue?

Me

Anti-spoofing exists exactly for such issues. So [we] don't think that we need to do anything.

Them

Où est le problème ?

Une réaction rapide de l'équipe de sécurité de Checkpoint.

Configuring anti-spoofing is a basic requirement.

Them

Are you planning some action regarding this issue?

Me

Anti-spoofing exists exactly for such issues. So [we] don't think that we need to do anything.

Them

Recommandation de base

Bien choisir son prestataire: le niveau de sécurité dépend de ses compétences.

Autres produits

Une attaque générique

- L'attaque peut impacter tout pare-feu utilisant des *ALGs*:
  - Pare-feu "propriétaire",
  - Interface Iptables,
  - Script Iptables maison.

Tests

- Facile à faire avec *opensvp*.
- Disponible sous GPL: `https://home.regit.org/software/opensvp/`
- Tout retour est le bienvenu.

Les attaques bas niveaux ne sont pas mortes

Le cas d'IPv6 sous Linux est caractéristique

- rp_filter pour IPv6 n'a pas été développé car trop couteux en CPU.
- Deux patches l'iméplementant ont été refusés.
- Une implémentation Netfilter du Reverse Path filtering est disponible dans Linux 3.3.

La configuration par défaut de Checkpoint

- L'utilisabilité a entrainé le choix de valeur par défaut non sure.
- L'anti-spoofing des cluster Checkpoint semble problématique à gérer.
- Voir : `http://rivald.blogspot.com/2011/01/checkpoint-utm-firewall-clusters-part-2.html`

Considérations génériques

S'élever est dangereux

- Considérer des couches OSI plus élevés est dangereux.
- Même des vieux protocoles comme FTP sont dangereux.
- Les "nouveaux" comme SIP héritent de cette tradition.

Considérations génériques

S'élever est dangereux

- Considérer des couches OSI plus élevés est dangereux.
- Même des vieux protocoles comme FTP sont dangereux.
- Les "nouveaux" comme SIP héritent de cette tradition.

À propos du niveau de sécurité

- La sécurité par défaut est un mythe:
  - Les configurations par défauts peuvent être vulnérables à des attaques.
  - Ne laisser aucun avertissements de l'interface impuni.
- La défense en profondeur ne doit pas rester un mythe:
  - Protéger des services "internes" même si ils sont derrière un pare-feu.
  - La séparation des routeurs et des pare-feu était une bonne idée.

Gestion d'une attaque générique

Une tache très difficile

- Il est impossible pour un individu
  - d'obtenir la liste des produits potentiellement vulnérables.
  - de contacter les personnes concernées.
- C'est encore pire si des scripts iptables maison sont vulnérables.

Source d'aide

- Contacter les CERT
  - Si vous n'avez pas de réponse, envoyez un second courriel.
  - Essayez de contacter le CERT Luxembourg, CERT Finland. La mise en concurrence peut augmenter les chances de prises en compte.
  - Microsoft Vulnerability Research (MSVR) est une alternative au CERT.
- Pour ce qui est des logiciels Open Source, contactez la liste de diffusion OSS security.

Questions

Avez-vous des questions ?

Merci à

- Pablo Neira, Patrick McHardy: les développeurs noyaux peuvent être sympas.
- Florian Westphal: pour son implémentation Netfilter de RP filter.

Plus d'information

- Mon blog : `https://home.regit.org`
- Secure use of Iptables and connection tracking helpers: `http://home.regit.org/netfilter-en/secure-use-of-helpers/`

Me joindre

- Courriel: eric@regit.org
- Twitter: @Regiteric