L'illusion comme concept de défense

Fabrice Prigent

UT1 / RéSIST

RéSIST, 29 Janvier 2008

Un contexte mouvant

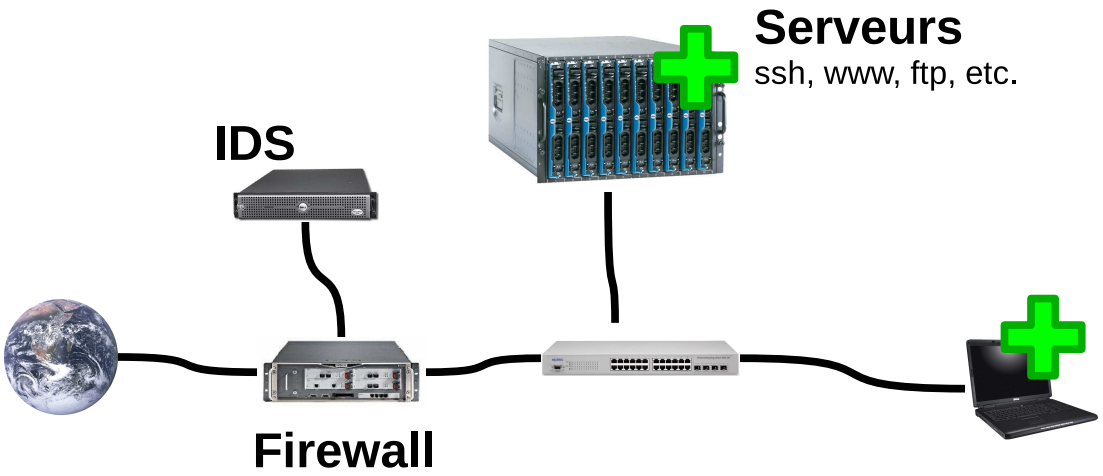
- Une industrialisation du piratage
  - De l'argent en grande quantité,
  - Objectifs d'efficacité (beaucoup de machines),
  - Une capacité de R & D très forte
    - Innovation
    - Reverse engineering
- Une utilisation renforcée de l'informatique
  - Des applications nombreuses généralement non contrôlées
  - Objectif : aller vite

Une ouverture très large

- Université en Sciences Sociales
- 16000 étudiants
- 2000 personnels
- WiFi en portail captif partout
- En quête d'informatisation permanente
- La sécurité n'est qu'un des aspects de la cellule réseau.

Comment protéger un réseau ?

- Un firewall avec de nombreuses règles (400 ouvertures, dont 30 en SSH)
- Des antivirus
- Un IDS



Problèmes de l'infrastructure standard

- Les firewall ne sont pas sensés gérer les contenus
  - Les attaques applicatives passent
- Les antivirus ont toujours un temps de retard
  - Au minimum 4 heures, mais 48 heures est fréquent
- Un IDS ne détecte que ce qu'il connait
  - Plus irréguliers encore que les antivirus
  - Des faux positifs très courants

Autres problèmes

- Le niveau de sécurité est rarement celui que l'on croit
  - antivirus pas à jour, même avec un déploiement centralisé,
  - des ouvertures erronées sur les firewalls,
  - des comptes avec des mots de passe ridicules (installation d'une appli).
- Un bon zéro-day (ou plutôt zéro signature day)
  - qui passe toute la sécurité
- etc.

Détecter l'inconnu

La solution serait de détecter les comportements déviants, sans signature. Ce qui revient à la problématique de détection de spams :

- Avoir suffisamment de critères (détecteurs), aucun n'étant absolu,
- Savoir ce que donnent les opérations normales (apprentissage ?),
- Faire une somme pondérée de ces critères,
- Réagir quand la somme dépasse un seuil.

Détecter l'inconnu : application

Ceci est déjà appliqué, de manière simplifié, sur les bloqueurs de bruteforce : sshdfilter ou autre.

- Bloquer les connexions sur un nombre d'échecs dépassé.

On généralise le processus aux autres applications

- Requêtes HTTP inconnues (404),
- Requêtes HTTP avec des paramètres inconnus,
- Comptes en ftp, pop, etc.

Détecter l'inconnu : problèmes

Les difficultés sont connues :

- Si la faille est exploitable au premier coup, on ne bloque rien (cas vécu en ssh)
- Si les erreurs sont courantes, on bloque trop vite

Détecter l'inconnu : première solution

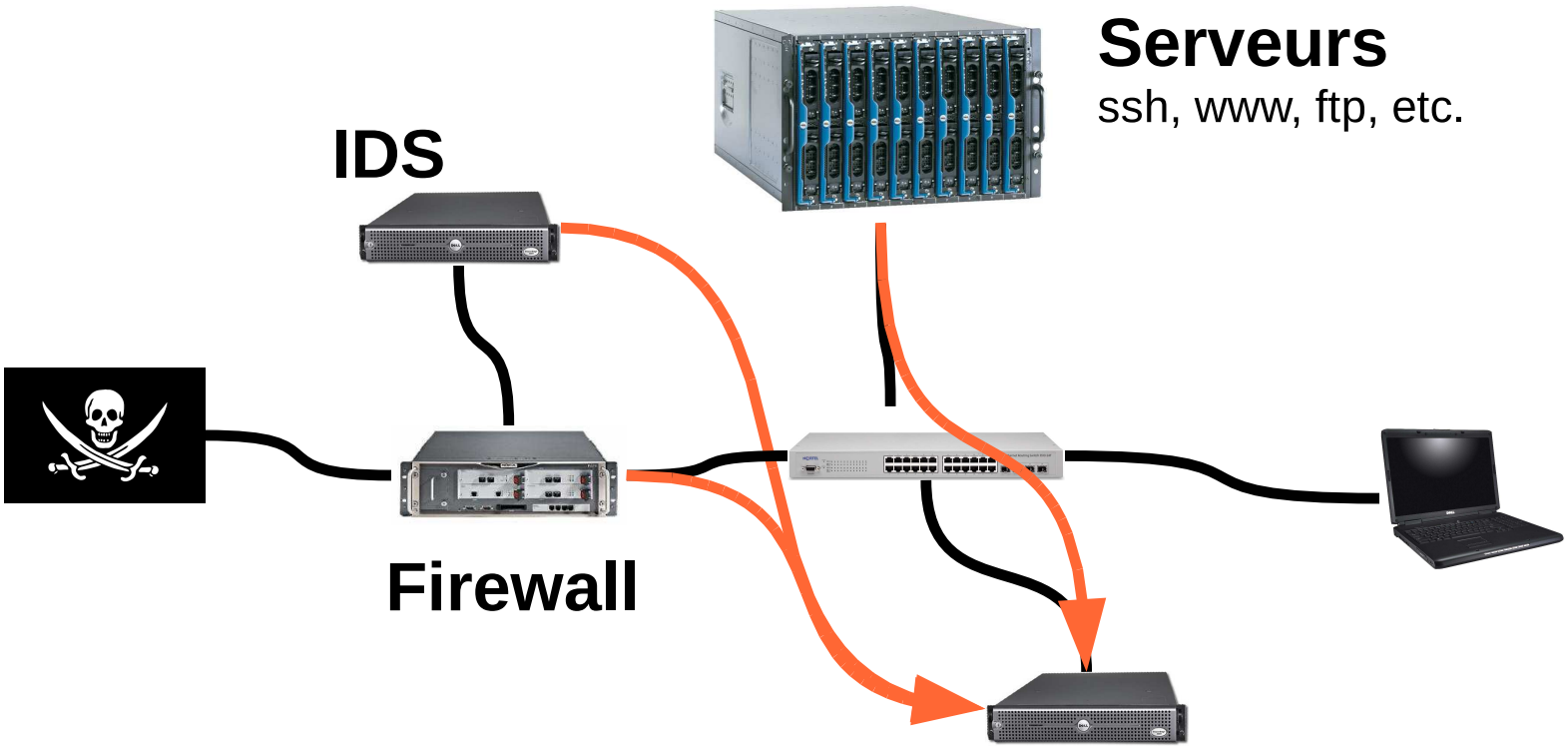
Centraliser au maximum :

- Centraliser les évènements étranges (la détection),
- Centraliser la décision,
- Diffuser l'information (voire l'action).

Ce qui permet :

- Si un bruteforce a lieu, les autres serveurs ssh bloqueront l'attaquant,
- D'élever le seuil de détection, et donc réduire les faux positifs.

Schéma



Détecter l'inconnu : Insuffisant

Mais c'est insuffisant :

- Si le firewall est correctement configuré : peu de services accessibles.
- Donc, peu de détecteurs actifs,
- Donc, si la faille est exploitable au premier coup, on ne bloque toujours rien dans la majorité des cas.

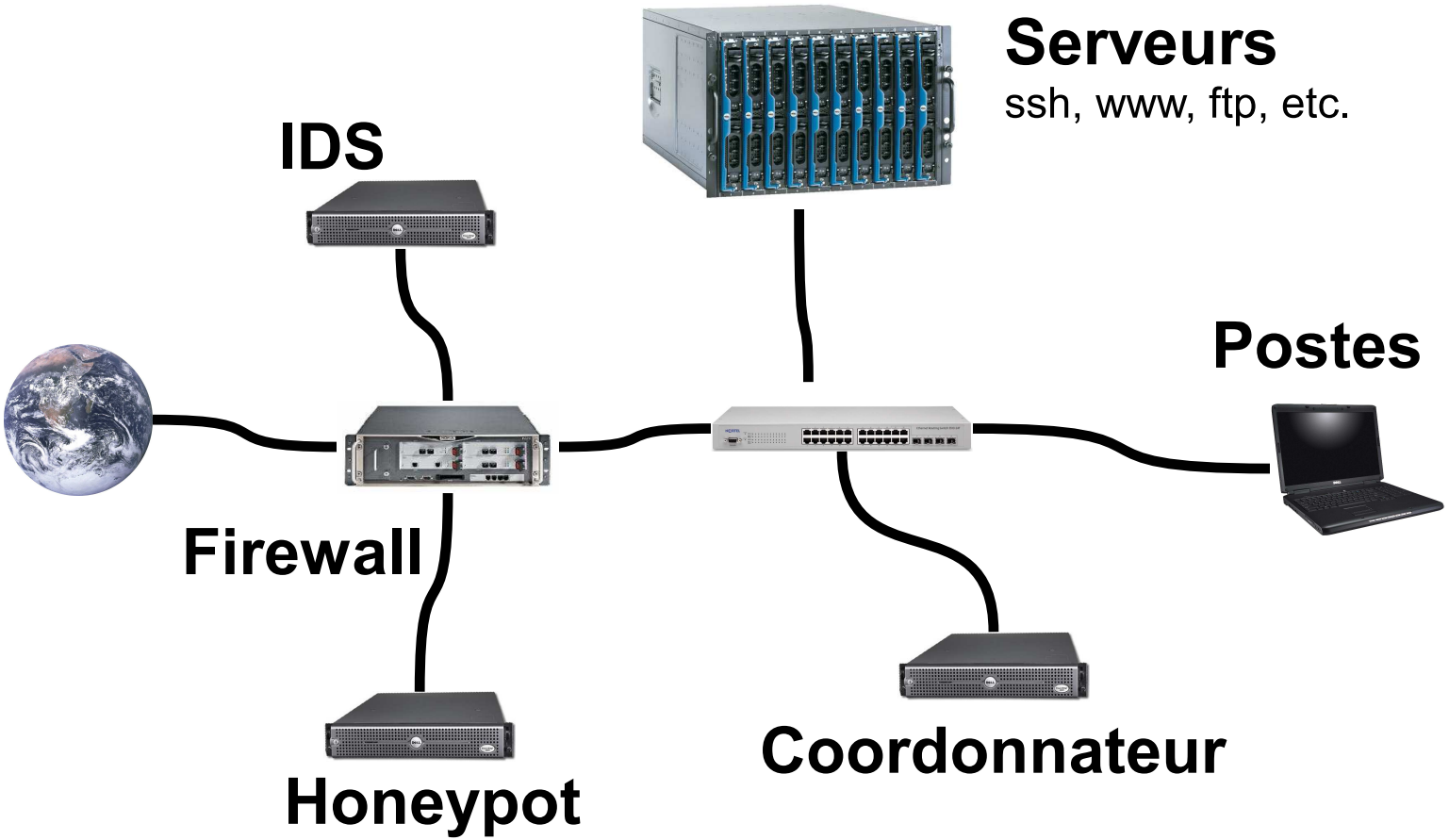
Détecter l'inconnu : deuxième solution

L'idéal serait d'avoir :

- Plein de serveurs non sensibles,
- Avec plein d'applications.. non sensibles aussi,
- Pour permettre l'extension du périmètre de détection.

Schéma

Cette solution s'appelle un honeypot :



Configuration du Firewall

- Le firewall redirige tous les paquets normalement interdits vers le honeypot,
- Le firewall interdit au honeypot de communiquer avec l'extérieur hormis,
  - Les mises à jours système (sur un serveur interne),
  - La journalisation (vers le coordonnateur),
  - La résolution DNS (vers notre serveur DNS).

Précautions sur le Firewall

- Le honeypot est physiquement séparé du réseau,
- Le firewall est optimisé pour gérer de très nombreuses connexions,
- Le honeypot n'a droit qu'à une faible bande passante (au cas où..).

Configuration du honeypot

Le honeypot est configuré comme suit :

- Les services réellement offerts par l'établissement sont fournis par de VRAIS serveurs, sauf si...,
  - Ils sont trop compliqués,
  - Trop dangereux,
  - Trop lourds.
  - Ils sont alors simplement simulés par un honeypot à faible interaction (mode sensor de xinetd).
- Les autres services sont systématiquement simulés par le même principe,
- Si un service n'est pas simulé, on annonce son indisponibilité (REJECT au lieu de DROP)

Configuration du honeypot : cas du WWW

Le WWW fait l'objet d'un traitement particulier

- Un logiciel dédié (HiP Hop) propose un certain nombre d'applications vulnérables,
- Il est virtualhost pour tout nom de domaine,
- Pour des raisons que nous verrons plus tard, il renvoie un message particulier si le virtualhost correspond à un vrai virtualhost.
- Les modules suhosin et modsecurity sont installés et leur logs "intéressants" remontés.

Journalisation du honeypot

Le honeypot journalise à tout va.

- Un snort avec la quasi totalité des règles repère les agressions,
- Les services sont en mode forte journalisation,
  - Pour repérer les actions offensives (et éviter le reste),
  - Pour obtenir l'adresse IP de l'attaquant,
- Même les simples connexions sont remontées (mais pas les scans SYN ou UDP),
- Les logs sont envoyés au coordonnateur.

Optimisation du honeypot

Le honeypot n'a pas besoin de rendre "efficacement" le service.

- Les paramètres TCP sont tunés pour ne pas prendre trop de ressources,
- La taille mémoire dévolue aux sessions de connexions est agrandie.

Ce que le honeypot n'est pas

Le Honeypot dans ce cas là n'est pas destiné à :

- à étudier la psychologie pirates : c'est le rôle des chercheurs
- à résister à une détection par la NSA ou le GRU : les bots et les scripts kiddies me suffisent
- à attirer les hackers de tout bord : qu'ils restent chez eux,
- à analyser de nouvelles techniques d'agression : c'est le rôle de sociétés spécialisées.

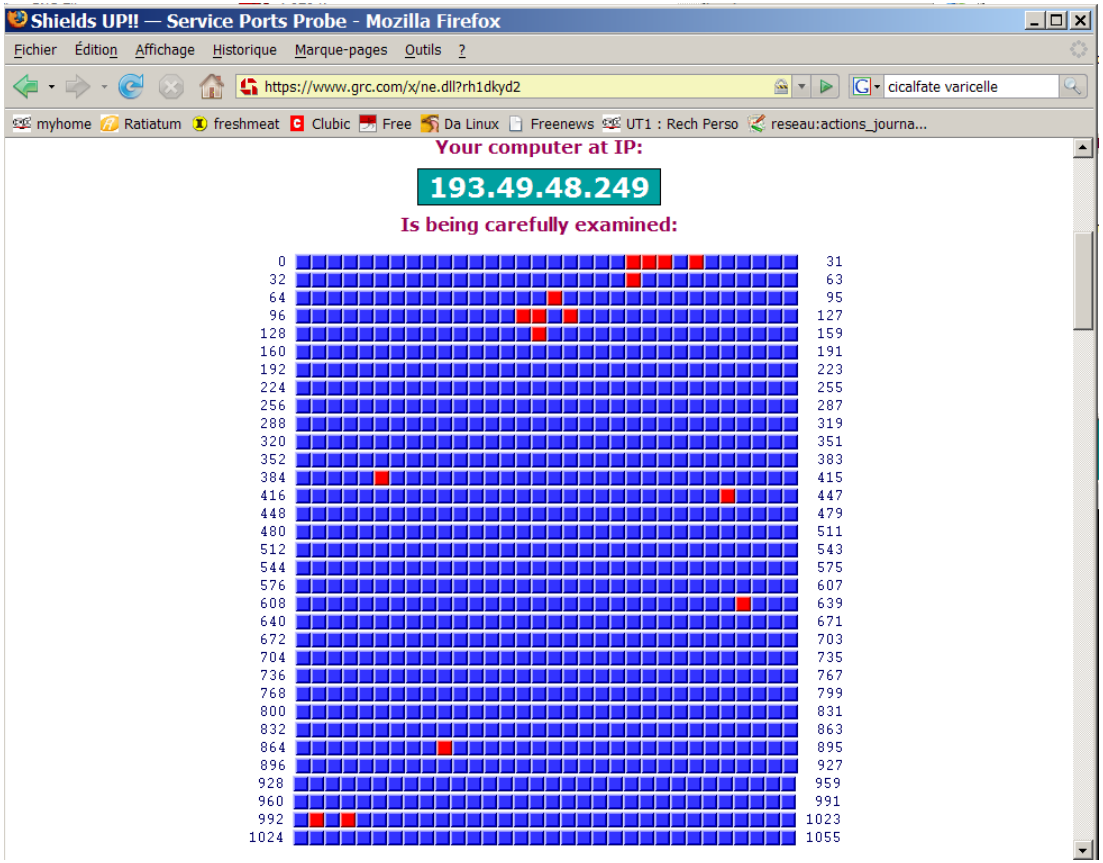
Indice : Je suis fonctionnaire et informaticien. Selon la définition de Larry Wall ma qualité première est la fainéantise.

Premiers résultats

Point de vue pirate, ver ou bot :

- 2560 serveurs web,
- 2560 serveurs ssh,
- 2560 serveurs netbios,
- ...

Premiers résultats : vision de "Shields UP ! !"



Premiers résultats : nmap avant la mise en place

```
[root@zaurus root]# nmap -P0 -O 127.102.0.1
Starting Nmap 4.02 ( http://www.insecure.org/nmap/ ) at 2007-11-01 19:34 CEST
Interesting ports on 127.102.0.1 (127.102.0.1):
(The 1649 ports scanned but not shown below are in state: filtered)
PORT     STATE SERVICE
80/tcp   open  http
Device type: general purpose|media device
Running: Linux 2.4.X, Pace embedded
OS details: Linux 2.4.18 - 2.4.27, Pace digital cable TV receiver
Uptime 128.907 days (since Tue Aug 1 13:21:31 2007)

Nmap finished: 1 IP address (1 host up) scanned in 1051.202 seconds
```

Premiers résultats : nmap après la mise en place

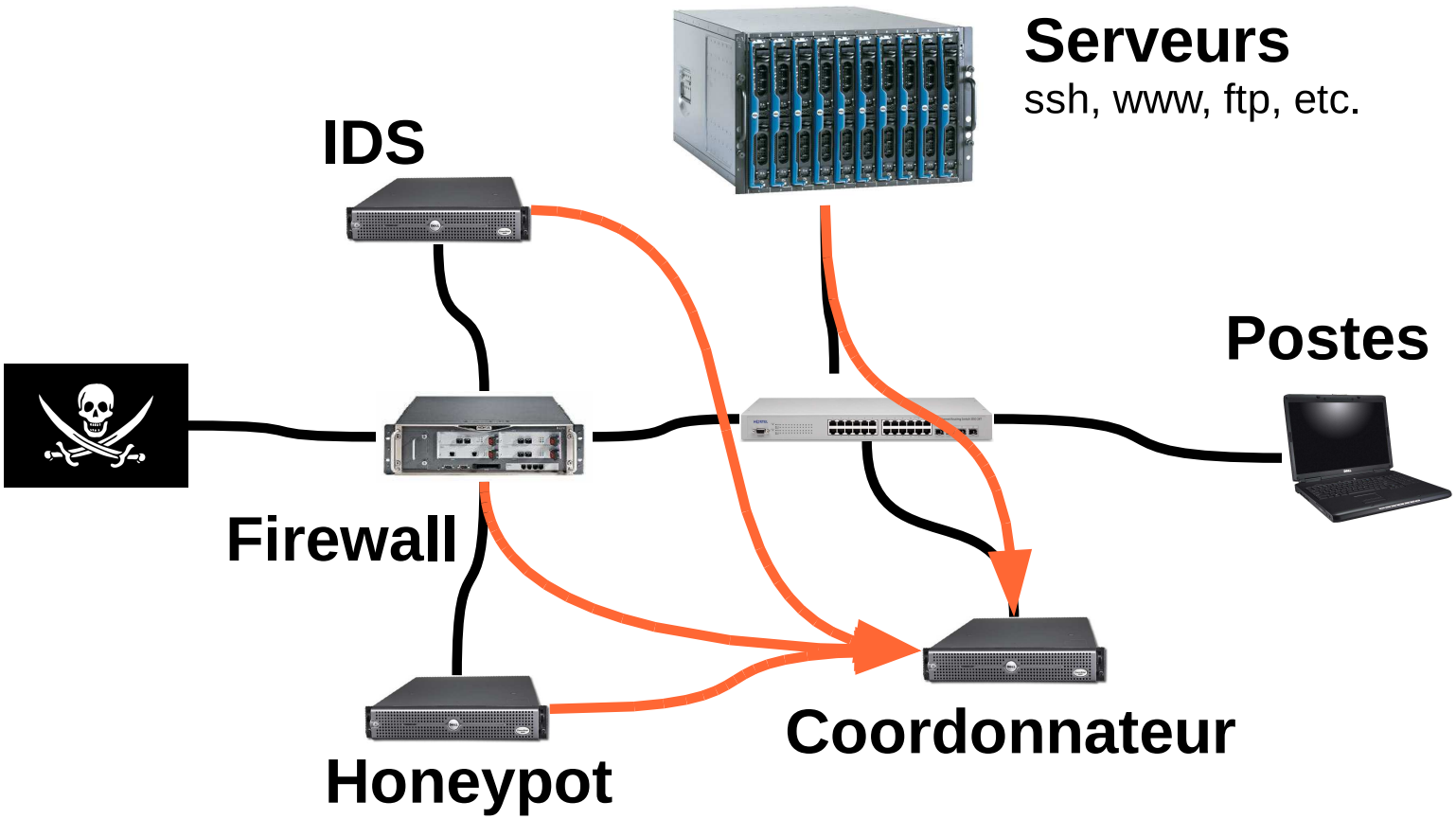
```
[root@zaurus root]# nmap -O 127.102.0.1
Starting Nmap 4.02 ( http://www.insecure.org/nmap/ ) at 2007-11-01 21:25 CEST
Interesting ports on 127.102.0.1 (127.102.0.1):
(The 1649 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
80/tcp   open  http
110/tcp  open  pop3
111/tcp  open  rpcbind
113/tcp  open  auth
137/tcp  open  netbios-ns
138/tcp  open  netbios-dgm
143/tcp  open  imap
389/tcp  open  ldap
445/tcp  open  microsoft-ds
636/tcp  open  ldapssl
873/tcp  open  rsync
993/tcp  open  imaps
995/tcp  open  pop3s
1433/tcp open  ms-sql-s
1434/tcp open  ms-sql-m
1494/tcp open  citrix-ica
3306/tcp open  mysql
3389/tcp open  ms-term-serv
5900/tcp open  vnc
Device type: general purpose|media device
Running: Linux 2.4.X, Pace embedded
OS details: Linux 2.4.18 - 2.4.27, Pace digital cable TV receiver
Uptime 129.191 days (since Tue Aug 1 13:21:31 2007)

Nmap finished: 1 IP address (1 host up) scanned in 3.989 seconds
```

Encore faut-il décider

Le coordonnateur joue alors son rôle de décision :

- pour chaque "agression" il augmente le compteur de l'IP associée dans une base de donnée,
- chaque 2 secondes il regarde les IP qui ont dépassé un seuil de 10 agressions en 5 minutes,
- chaque 5 minutes, il regarde les IP qui ont dépassé un seuil de 20 agressions en 1 heure,
- ces IP sont alors qualifiées d'attaquantes,
- la liste de ces IP est propagée à chaque élément du réseau.

Détection d'une attaque



Statistiques d'IP agressives

Rapidement, la nouvelle architecture donne un résultat tout à fait intéressant. Voici l'exemple d'une semaine standard (Hors mssql et netbios) :

Détections à UT1

Une journée "intéressante" à UT1 :

Quarantaine Journalière

DNS	IP	Attaque	Nb
3-10-251-64.serverpronto.com	64.251.10.3	Echec PROFTPD	331611
203.252.53.222	203.252.53.222	Echec PROFTPD	154957
218.189.243.246	218.189.243.246	Echec PROFTPD	102451
61.40.129.197	61.40.129.197	Echec SSH	17219
125.128.228.126	125.128.228.126	Scan de port vnc	3184
83-65-75-114.hebragasse-ii.xdsl-line.inode.at	83.65.75.114	Echec SSH	1581
211.99.196.208	211.99.196.208	Echec SSH	1202
211.151.95.193	211.151.95.193	Echec SSH	1035
121.156.65.219	121.156.65.219	Echec SSH	779
138.16.102.121.dy.bbexcite.jp	121.102.16.138	Scan de port vnc	547
138.16.102.121.dy.bbexcite.jp	121.102.16.138	POLICY VNC server response	470
83.244.81.252	83.244.81.252	Echec SSH	458
wsip-24-234-56-32.lv.lv.cox.net	24.234.56.32	Echec SSH	369
221.144.173.15	221.144.173.15	Echec SSH	358
pc110.lyon.inserm.fr	194.57.165.199	Echec SSH	320
218.249.210.161	218.249.210.161	Echec SSH	306
220.207.191.1	220.207.191.1	Scan de port vnc	283
p3082-ipbffx02osakakita.osaka.ocn.ne.jp	61.113.215.82	Scan de port vnc	277
12.164.100.229	12.164.100.229	Scan de port vnc	257
195.207.160.93	195.207.160.93	Scan de port rdesktop	247
61.138.6.86	61.138.6.86	Scan de port actif telnet	246
220.207.191.1	220.207.191.1	POLICY VNC server response	238
61.138.6.86	61.138.6.86	TELNET Solaris login environment variable authenti	229
p3082-ipbffx02osakakita.osaka.ocn.ne.jp	61.113.215.82	POLICY VNC server response	221

Utilisation du Honeypot

La décision de classification est prise. On peut bien sûr appliquer cette décision de plusieurs manières :

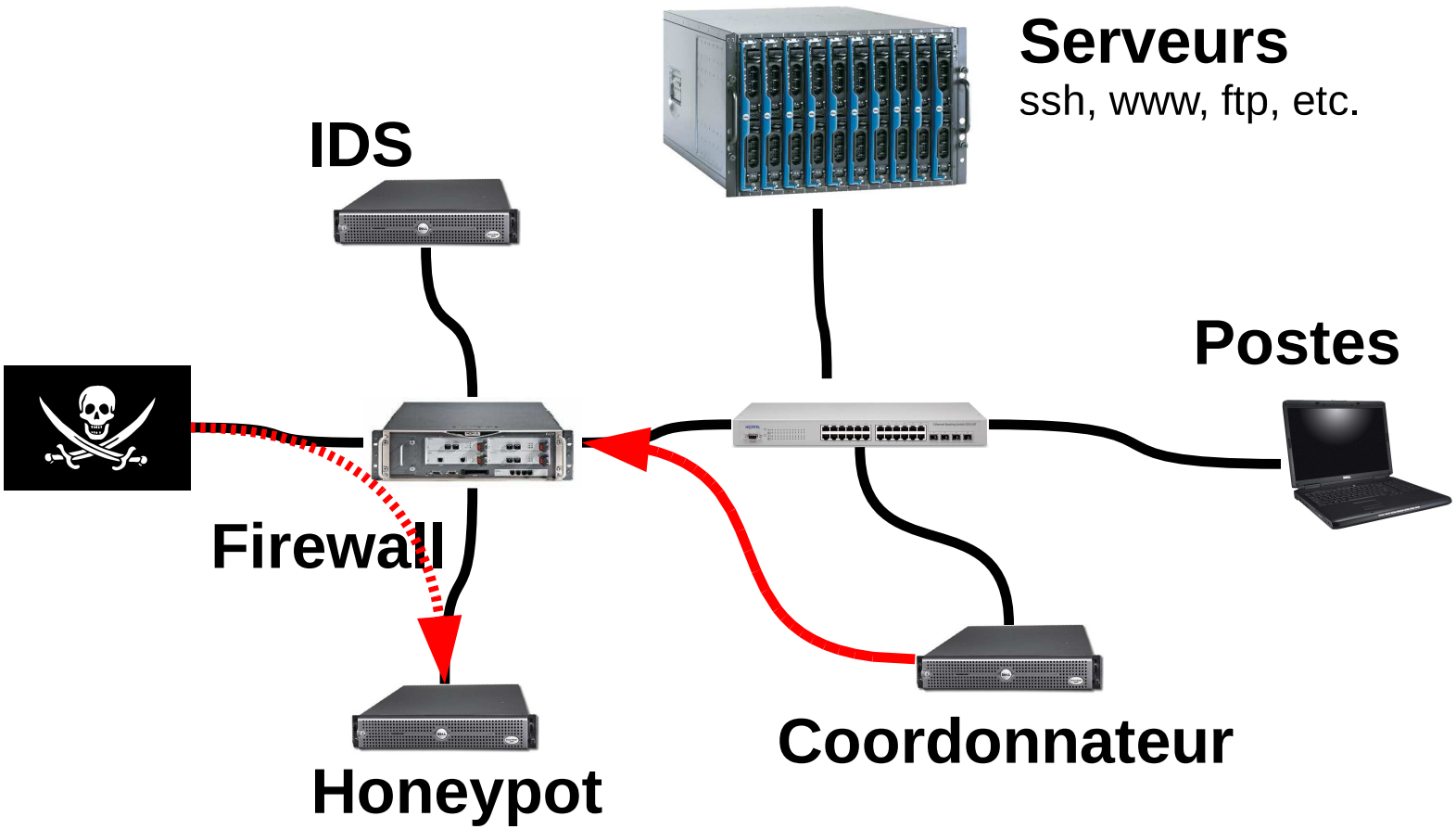
- bloquer le pirate : qui reprend son action dès que possible
- le ralentir : ce qui ne sert pas à grand chose
- signaler à son FAI ses actions : long, compliqué et souvent inutile
- ou bien utiliser notre honeypot

Honeypot : une quarantaine illusionniste

Le détournement de l'attaquant sur le honeypot est avantageux sur plusieurs points

- il ne touche plus aux serveurs de production,
- si son attaque est automatique, elle va s'essouffler sur le honeypot,
- tant qu'il continue, on considère son IP comme dangereuse,
- s'il utilise une faille inconnue
  - il prendra possession du honeypot,
  - il tentera de récupérer son rootkit,
  - et se fera donc repérer.

Honeypot : schéma de la quarantaine



Résultats à UT1

Plusieurs résultats tangibles :

- à 99,9% plus aucune attaque brute force n'atteint nos serveurs,
- les attaques web référencées (style Code Rouge) sont bloquées à 99,9% (en fait aucun cas depuis la mise en place),
- les attaques web "inconnues" sont parfois bloquées avant d'arriver sur nos serveurs (à 30%).

Une charge raisonnable

- le honeypot est une machine comme une autre, et même plus facile à administrer que les autres (vive le yum update)
- la charge CPU est faible (mais attention au load)
- le coût est ridicule : pour nous c'est un pentium 933 avec 1Go de RAM.

Honeypot : une solution universelle ?

Si la solution est efficace, elle n'est bien évidemment pas universelle :

- les universités ont beaucoup d'adresses IP disponibles, ce qui est plutôt rare ailleurs,
- nous ne sommes quasiment jamais la cible d'attaque spécifique,
- les pirates ne sont pas prêts à consommer des ressources.

Mais elle correspond bien à notre besoin

- peu de moyens pour la sécurité,
- une compétence nécessaire assez faible,
- une administration simplifiée.

Honeypot : le détecter ?

C'est assez facile, pour peu que l'on s'y attarde :

- les timestamp,
- les TTL,
- les bannières avec le nom de la machine,
- etc.

Mais cela obligera malgré tout le pirate à revoir ses procédures d'automatisation

Améliorer le principe

Plusieurs axes d'amélioration sont possibles

- virtualiser plusieurs honeypots,
- modifier les communications pour les rendres indiscernables (timestamp, TTL, etc.),
- renforcer la simulation WWW (indexation par google).

Conclusion

Cette méthode est très efficace pour notre environnement, sans présenter de handicaps lourds. Elle garde de plus une bonne marge de progression, pour s'adapter à de nouvelles agressions.
L'avènement prochain d'IPV V6 devrait réduire l'intérêt d'une telle méthode.

Conclusion

Des questions ?