Sécuriser les réseaux par la connaissance des usages

François Dagorn
30 août 2007

Résumé

Ce document est un tutoriel sur l'amélioration de la sécurité d'un réseau obtenue par une meilleure connaissance des usages. Il est basé sur un retour d'expérience dans la gestion au quotidien d'un petit réseau Ethernet TCP/IP (commuté et segmenté) protégé en amont par un pare-feu.
Commentaires et remarques à `prenom.nom at univ-rennes1 point fr`.

Table des matières

Chapitre 1

Introduction

La sécurisation d'un réseau Ethernet TCP/IP dans le monde académique repose en général sur la mise en œuvre d'un pare-feu ainsi que sur le cloisonnement des flux à l'aide de VLANs. Le pare-feu régule les flux depuis et vers l'extérieur, les VLANs gèrent les flux internes. La configuration des pare-feux ainsi que celle des VLANs s'effectue à l'aide de règles qui reposent sur l'idée que ce qui n'est pas explicitement autorisé est interdit. Au fil du temps, les administrateurs du réseau mettent à jour les règles et des doutes apparaissent sur l'étanchéité du dispositif :

- des portes ne sont-elles pas ouvertes à tort ?
- comment vérifier que l'utilisation du réseau est conforme à la charte d'usage signée par les utilisateurs ?
- et surtout, comment détecter les intrusions potentielles ?

Les pare-feux et les éléments responsables de la gestion des VLANs produisent des fichiers d'événements (*logs*) en général très volumineux. Les consulter est utile en phase de mise au point de nouvelles règles, ils sont par contre difficilement exploitables pour vérifier l'efficacité des règles de sécurité positionnées. Des dispositifs supplémentaires sont donc nécessaires pour :

- vérifier les ports ouverts sur l'ensemble des machines du réseau ;
- visualiser le trafic (en différents points) par machines, protocoles, ports ;
- vérifier la vulnérabilité des services installés ;
- détecter les tentatives d'attaques ;
- détecter les éventuels intrus.

Après avoir effectué le point sur les principales attaques provenant du réseau, nous présentons ici différents outils déployés pour répondre aux questions posées ci-dessus. Ces outils procurent à l'administrateur du réseau une bonne connaissance

des usages (souhaités ou non) et lui permettent de faire converger la politique de sécurité souhaitée et son implémentation effective.

Chapitre 2

Les principales attaques provenant du réseau

Pour protéger un réseau, il faut connaître les principales attaques qui peuvent l'affecter, la suite de ce chapitre les présente brièvement.

2.1 Les attaques sur les protocoles IP, TCP et UDP

Les attaques sur les protocoles TCP/IP tentent d'exploiter les concepts de base des couches réseau et transport. Elles ont pour but de bloquer le fonctionnement d'un réseau (déni de service) ou d'usurper l'identité d'une machine.

2.1.1 Le déni de service

Les attaques de type « déni de service » ont pour but de consommer toutes les ressources d'un réseau cible pour empêcher son fonctionnement normal. Ces attaques sont opérées par des groupes qui conjuguent leurs efforts pour le plaisir de nuire mais également par des organisations mafieuses qui demandent des rançons. Parmi un très grand nombre d'attaques en déni de service on peut citer les suivantes :

– **les TCP SYN flood** qui exploitent le mode connecté de TCP : pour établir une connexion, une machine source émet un paquet SYN (*synchronize*) auquel la machine destination répond par un paquet SYN-ACK (*synchronize acknowledment*). Dans une situation normale, la machine source émet alors un paquet ACK (*acknowledgment*) et la connexion est établie. Le schéma d'une attaque TCP SYN est de supprimer la dernière étape (envoi du paquet ACK), la machine cible l'attend alors pendant un certain temps. En répétant l'opération à grande échelle, on sature la machine cible en lui faisant atteindre

le nombre maximum de connexion TCP qu'elle peut supporter : elle ne peut plus fonctionner.

– **le ping of death** dont le principe est d'envoyer des paquets ICMP dont la taille est supérieure à la taille maximale d'un paquet IP, certains équipements se bloquent alors (ils sont en voie de disparition).

2.1.2 L'usurpation d'identité

L'usurpation d'identité IP (*IP spoofing*) est basée sur le fait que certains services utilisent l'adresse IP comme paramètre de confiance (*rlogin*, *allow from* d'apache, ...). Dès qu'une machine cliente dispose d'une connexion sur un serveur via une adresse IP de confiance, l'attaquant essaie d'usurper son adresse IP. L'attaquant doit au préalable réduire le vrai client au silence en le saturant au par une attaque de type « déni de service » (SYN flood par exemple). Il doit ensuite tenter d'exploiter une caractéristique du protocole TCP (numéros de séquences et d'acquittement qui ne sont pas toujours aléatoires) en ouvrant une session TCP usurpant l'adresse IP de la machine cliente. L'attaquant peut ensuite, s'il a réussi à deviner l'algorithme d'attribution des numéros de séquences, se faire passer la machine cliente.

Cette technique est évidemment très difficile à mettre en œuvre, certains systèmes qui n'utilisent pas un séquencement aléatoires y sont cependant toujours vulnérables.

2.2 Les attaques sur les protocoles applicatifs, les données qu'ils transportent et les applications qui les mettent en œuvre

Ces attaques sont les plus nombreuses car elles sont parfois très simples à déployer. Elles ciblent les vulnérabilités des protocoles applicatifs (SMTP, HTTP, ...) ou de leurs implémentations mais également les applications qu'ils permettent d'utiliser (les scripts CGI, les applications PHP, ...). On peut aussi classer ici les attaques sur les applications, notamment celles portant sur les navigateurs Web.

L'attaquant peut, en fonction des attaques, exécuter du code malveillant, prendre le contrôle du système ou engendrer un déni de service. Les trois exemples suivants mettent en évidence des failles exploitables du protocole HTTP, du serveur HTTP Apache et du navigateur FireFox. Les deux derniers exemples ont été choisis car ils sont relatifs à des outils très utilisés dans le monde universitaire. Il est évident que d'autres outils encore plus utilisés par ailleurs sont encore plus vulnérables (Microsoft Internet Explorer notamment) cf. *SANS Top-20 Internet Security Attack targets (2006 annual Update)*[1].

[1] http://www.sans.org/top20/

2.2.1 Attaque sur HTTP

La méthode TRACE du protocole HTTP/1.1 a été spécifiée pour faciliter le développement d'application au dessus de HTTP. Elle permet de visualiser dans le corps d'une réponse, toutes les en-têtes qui sont transmises lors d'une requète HTTP. Cette fonctionnalité est pratique dans une phase de mise au point pour vérifier ce que l'application distante reçoit. Malicieusement utilisée elle permet à un attaquant de visionner des *cookies*, des authentification de type `auth-type : basic` encodées en base 64 (en clair), ...

Cette faille peut être exploitée de la manière suivante :

- l'attaquant poste un mail à destination de l'attaqué (en usurpant l'adresse d'un collègue de celui-ci de préférence) ;
- l'attaqué consulte ce mail via une interface de type WebMail ;
- le courrier contient une pièce jointe qui va faire éxécuter un code JavaScript qui engendre une requète HTTP TRACE ;
- le script effectue une requète HTTP sur le serveur de l'attaquant en passant en paramètre le résultat de la requète TRACE effectuée sur le serveur de l'attaqué ;
- en consultant son `log` d'httpd, l'attaquant visualise les *cookies*, les champs `auth` en base 64, ...,

2.2.2 Attaques sur Apache

Le serveur HTTP Apache est composé d'un corps principal auquel s'ajoutent divers modules périphériques optionnels. De façon assez régulière, des failles sont répertoriées dans des modules, telles celles-ci qui permettent l'exécution de code arbitraire sur le serveur :

- le module mod_rewrite des branches 1.3.8, 2.0.46 et 2.2.0 permet en fonction des règles d'écritures (les Rewrite rules) d'exécuter du code arbitraire ;
- le module mod_tcl qui permet de faire exécuter des scripts developpés en TCL est vunérable dans sa version 1.0. Il autorise d'exécuter à distance du code arbitraire.

2.2.3 Attaque sur FireFox

Le problème le plus récurrent concerne l'activation des scripts JavaScript, celle-ci permet, dans certains cas, l'exécution de code arbiraire comme l'indique l'avis du CERTA du 13 novembre 2006 : « *il a été démontré qu'il était possible de modifier des objets scripts en cours d'exécution. Cette possibilité conduit potentiellement à*

l'exécution de code JavaScript arbitraire. Cette fonctionalité touche le moteur du navigateur qui est aussi utilisé dans le lecteur de mail qui devient ainsi vulnérable si la fonctionalité JavaScript est activée ».

2.3 Les attaques sur les systèmes d'exploitation ou leurs composants

Les systèmes d'exploitation et leurs composants peuvent également comporter des vulnérabilités qui peuvent être exploitées à distance. Plusieurs cas ont été signalés concernant des DLL de Windows, notamment à propos de la visualisation d'images WMF permettant l'exécution de code arbitraire. Pour exploiter il suffit de placer une image WMF corrompue sur un serveur Web et d'attendre les clients utilisant la DLL ciblée.

Les noyaux Linux peuvent également comporter des failles qui sont accessibles aux utilisateurs disposant d'un accès sur la machine cible. Combinées avec une attaque sur un service réseau (httpd par exemple), elles peuvent donner des accès privilégiés à un attaquant, ou provoquer un déni de service comme dans le cas de la *vulnérabilité du noyau Linux avec IPv6*[2] publiée par le CERTA le 07 novembre 2006.

2.4 Premières mesures à prendre

Pour protéger un réseau il convient de s'informer très régulièrement sur ses vulnérabilités potentielles. Il revient aux différentes communautés d'utilisateurs de l'Internet de s'organiser en fonction du niveau de risques encourus. En France, les universités disposent des services du CERT-RENATER[3] et de son réseau de correspondants de sécurité de sites. Plus généralement, en France, le CERTA[4] (Centre d'Expertise Gouvernemental de Réponse et de Traitement des Attaques informatiques) diffuse journellement des alertes et des avis de sécurité qui décrivent les risques ainsi que les mesures à prendre immédiatement.

S'agissant des trois exemples décrits ci-dessus, dans les avis du CERTA on découvre qu'il convient respectivement de ne plus autoriser la méthode TRACE, de mettre à jour Apache et d'installer la version `Firefox 1.5.0.8`.

[2] http://www.certa.ssi.gouv.fr/site/CERTA-2006-AVI-478/index.htm
[3] http://www.renater.fr/spip.php?rubrique19
[4] http ://www.certa.ssi.gouv.fr/site/index.htm

Chapitre 3

Les outils pour espionner

Pour comprendre le fonctionnement d'un réseau ou pour mesurer les risques liés à l'écoute, l'utilisation des renifleurs (*sniffers*) est recommandée. La suite de ce chapitre présente `tcpdump`[1] et `Ethereal`[2], deux des renifleurs les plus utilisés.

En fonction de l'endroit où le renifleur est positionné, l'écoute sera plus ou moins intéressante. Dans un réseau commuté, par défaut, on n'écoutera que le trafic destiné à la machine hébergeant le *sniffer*[3] (les échanges *unicast* où elle apparaît ainsi que les *broadcast* de son domaine de diffusion). Raccordé à un concentrateur (*hub*), le *sniffer* écoutera l'ensemble du trafic destiné à toutes les machines.

Pour fonctionner normalement, `tcpdump` et `Ethereal` nécessitent de placer l'interface d'écoute en mode indiscret (*promiscuous*), ceci requiert des privilèges d'administrateur sur la machine qui les héberge (par défaut une interface réseau ne s'intéresse qu'au trafic qui lui est destiné, en mode indiscret, tout le trafic est espionné).

`tcpdump` et `Ethereal` utilisent la bibliothèque portable `libpcap` qui permet de capturer des paquets sur une interface réseau. Le principe des 2 outils est le même : capturer des paquets en tenant compte d'un filtre de capture potentiellement positionné par l'utilisateur. Ils fonctionnent tous deux sous `Windows XP` et `Linux`, `tcpdump` est utilisable en mode ligne uniquement tandis que `Ethereal` procure une interface graphique agréable.

[1] http://www.tcpdump.org

[2] http://www.ethereal.com/

[3] Sur certains commutateurs, des ports peuvent être configurés pour répliquer le trafic d'autres ports (cf. *port monitoring* chez Cisco par exemple.

3.1 Rédiger des filtres de capture pour `tcpdump` et `Ethereal`

Les filtres de captures permettent de restreindre le nombre de paquets capturés en ciblant précisément le type de recherche à effectuer. La notification utilisée pour spécifier des filtres de captures permet de sélectionner des réseaux, des noms de machines, des protocoles et des numéros de ports. Les différents éléments d'un filtre peuvent être combinés logiquement (`and`, `or`, `not`) comme dans les exemples suivants :

- `net 148.65.13.0 mask 255.255.255.0`
- `host www.machin.fr` : un nom de machine à résoudre ou un numéro IP ;
- `tcp` : on cible ici uniquement les paquets TCP. Les autres valeurs possibles incluent `udp`, `ip`, `icmp`, `broadcast`, `multicast`, `arp`, ...
- `port 80` : le port 80 en TCP ou UDP ;
- `tcp and port 80` : le port TCP 80 ;
- `tcp and not port 80` : tout TCP sauf le port 80 ;
- `not broadcast and not multicast and not icmp and not arp` : tout sauf les `broadcast`, le `multicast`, les paquets ICMP et les paquets ARP ;

3.2 tcpdump

Les options de démarrage de `tcpdump` sont les suivantes :

```
tcpdump [ -adeflnNOpqRStuvxX ] [ -c nombre de paquets ]
        [ -C taille du fichier de capture ] [ -F fichier ]
        [ -i interface ] [ -m module ] [ -r fichier ]
        [ -s nombre d'octets ] [ -T type ] [ -w fichier ]
        [ -E algo:secret ] [ expression ]
```

Parmi les options les plus usuelles de `tcpdump` on peut citer les suivantes :

`-c nombre de paquets` qui arrêtera la capture lorque le nombre de paquets indiqué sera atteint ;

`-w fichier` pour écrire les paquets capturés dans un fichier ;

`-x` pour imprimer les paquets capturés en codage hexadécimal ;

`-X` pour imprimer les paquets capturés en ASCII ;

`-s nombre d'octets` pris en compte dans chaque paquets (68 par défaut). En indiquant 0, on stipule qu'il faut utiliser le nombre d'octets nécessaires pour capturer la totalité du contenu des paquets.

`expression` correspond à un filtre rédigé comme indiqué en section 3.1

3.2.1 Un exemple d'utilisation de tcpdump

```
/usr/sbin/tcpdump -x -X -s 0 host www.google.fr
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
14:12:32.220589 IP zag.trucs.test-reseaux.fr.34904 > fg-in-f147.google.com.http:
   S 4099167971:4099167971(0) win 5840 <mss 1460,sackOK,timestamp 357988976 0,
   nop,wscale 2>
        0x0000:  000b 4668 a600 0007 e9eb aadc 0800 4500  ..Fh..........E.
        0x0010:  003c 2c93 4000 4006 4a35 943c 0a16 480e  .<,.@.@.J5.<..H.
        0x0020:  dd93 8858 0050 f454 56e3 0000 0000 a002  ...X.P.TV.......
        0x0030:  16d0 099a 0000 0204 05b4 0402 080a 1556  ...............V
        0x0040:  7a70 0000 0000 0103 0302                 zp........
14:12:32.240613 IP fg-in-f147.google.com.http > zag.trucs.test-reseaux.fr.34904:
    S 2805043872:2805043872(0) ack 4099167972 win 8190 <mss 1380>
        0x0000:  0007 e9eb aadc 000b 4668 a600 0800 4500  ........Fh....E.
        0x0010:  002c 367b 0000 ef06 d15c 480e dd93 943c  .,6......H....<
        0x0020:  0a16 0050 8858 a731 92a0 f454 56e4 6012  ...P.X.1...TV.‘.
        0x0030:  1ffe a6c0 0000 0204 0564 0000            .........d..
14:12:32.240651 IP zag.trucs.test-reseaux.fr.34904 > fg-in-f147.google.com.http:
    ack 1 win 5840
        0x0000:  000b 4668 a600 0007 e9eb aadc 0800 4500  ..Fh..........E.
        0x0010:  0028 2c95 4000 4006 4a47 943c 0a16 480e  .(,.@.@.JG.<..H.
        0x0020:  dd93 8858 0050 f454 56e4 a731 92a1 5010  ...X.P.TV..1..P.
        0x0030:  16d0 c75b 0000                           ...[..
14:12:32.243247 IP zag.trucs.test-reseaux.fr.34904 > fg-in-f147.google.com.http:
    P 1:503(502) ack 1 win 5840
        0x0000:  000b 4668 a600 0007 e9eb aadc 0800 4500  ..Fh..........E.
        0x0030:  16d0 c604 0000 4745 5420 2f20 4854 5450  ......GET./.HTTP
        0x0040:  2f31 2e31 0d0a 486f 7374 3a20 7777 772e  /1.1..Host:.www.
        0x0050:  676f 6f67 6c65 2e66 720d 0a55 7365 722d  google.fr..User-
        0x0060:  4167 656e 743a 204d 6f7a 696c 6c61 2f35  Agent:.Mozilla/5
        0x0070:  2e30 2028 5831 313b 2055 3b20 4c69 6e75  .0.(X11;.U;.Linu
        0x0080:  7820 6936 3836 3b20 656e 2d55 533b 2072  x.i686;.en-US;.r
        0x0090:  763a 312e 372e 3132 2920 4765 636b 6f2f  v:1.7.12).Gecko/
        0x00a0:  3230 3035 3039 3230 0d0a 4163 6365 7074  20050920..Accept
        0x00b0:  3a20 7465 7874 2f78 6d6c 2c61 7070 6c69  :.text/xml,appli
```

L'exemple précédent est une trace des échanges entre un navigateur WWW et un serveur HTTP. Le premier paquet est l'établissement de la connexion TCP entre le client et le serveur (émission d'un paquet SYN marqué du drapeau S par `tcpdump`). Le second paquet est l'acceptation de la connexion par le serveur qui émet un paquet SYN-ACK à destination du client. Le troisième paquet est l'émission d'un paquet ACK par le client pour valider son acceptation de la connexion. Le quatrième paquet TCP marque le début de la première requête HTTP (GET /).

3.3 Ethereal

En dehors de son interface d'usage graphique (très riche), `Ethereal` se distingue de `tcpdump` par le très grand nombre de protocoles qu'il interprète (plus de 500 protocoles, dont BitTorrent, CUPS, HTTP, ICQ, IMAP, LDAP, NTP ... cf. http://www.ethereal.com/docs/dfref/). Parmi les principales fonctionnalités d'`Ethereal` on peut citer les suivantes :

- adaptation à différentes technologie de réseaux locaux (Ethernet, FDDI, Token Ring, 802.11, ...) ;
- filtrage des paquets en capture et impression (avec personnalisation de l'utilisation de la couleur) ;
- suivi de session TCP ;
- création de statistiques et de graphes (nombre de paquets, nombre de requêtes, ...).

3.3.1 Un exemple d'utilisation d'`Ethereal`

Les figures suivantes présentent `Ethereal` en fonctionnement, l'utilisateur a demandé de tracer à l'aide d'un filtre de capture (port 80). À l'issue de la capture, il a selectionné un paquet TCP qui transporte une requête HTTP. Il a en outre demandé l'interprétation du protocole HTTP (figure 3.1). La figure 3.2 montre le rendu l'option *Follow TCP Stream*.

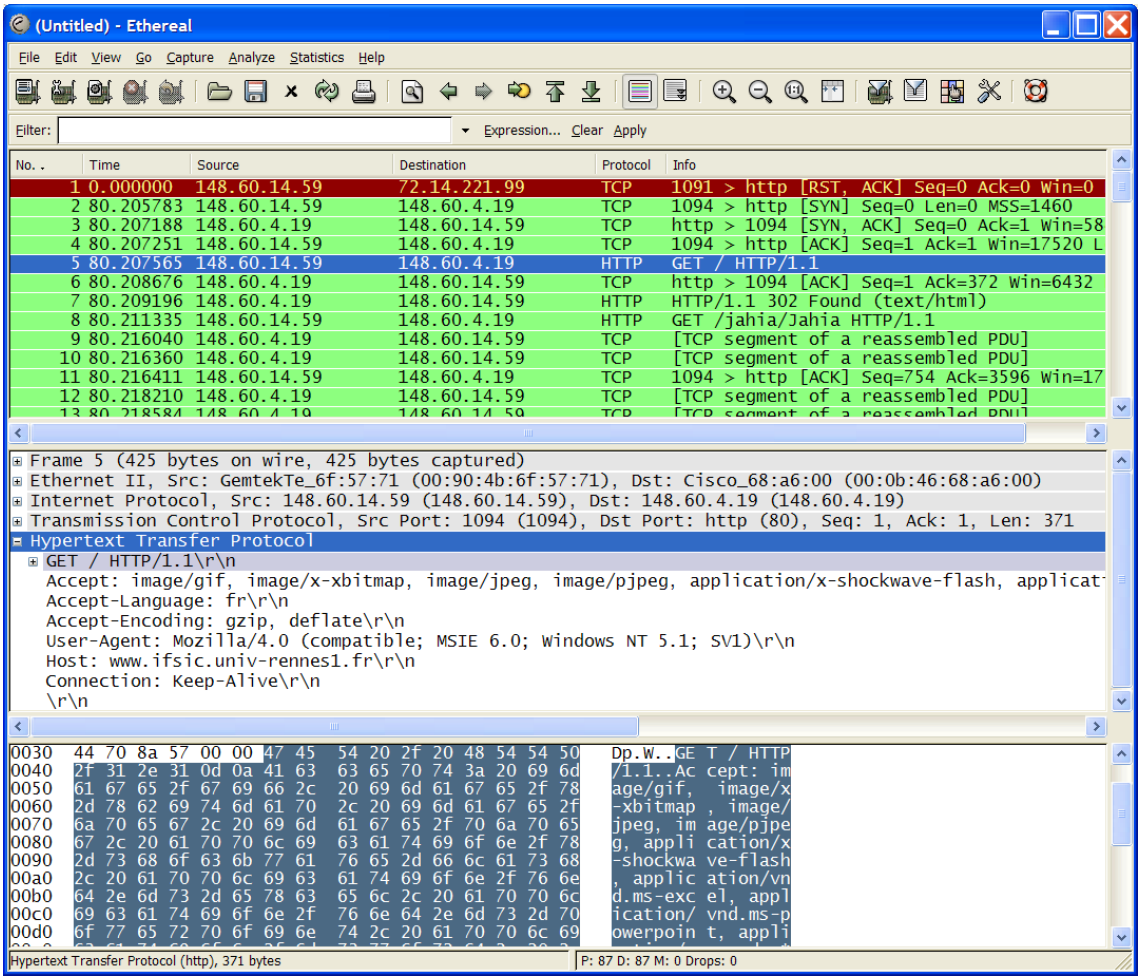


Figure 3.1: `Ethereal` capturant avec un filtre *port 80*

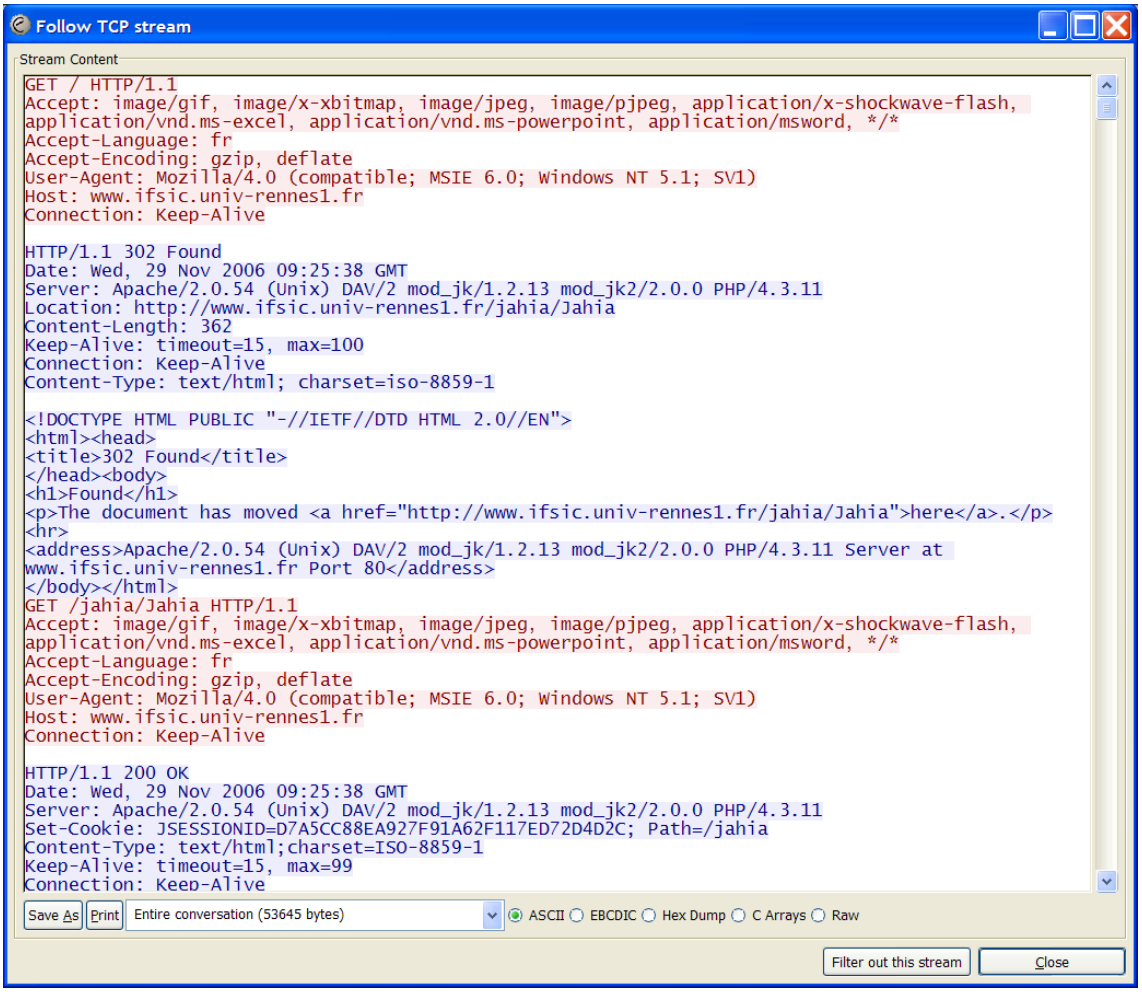


Figure 3.2: l'option *Follow TCP stream* d'`Ethereal`

3.4 Conclusion

L'intérêt de ces outils pour un administrateur de réseau est manifeste. Ils sont par contre très dangereux lorsque des personnes mal intentionnées les utilisent. Compte tenu de leur capacité à interpréter en clair des protocoles aussi répandu que HTTP, POP, IMAP, LDAP, ... il est évident que l'usage des versions sécurisées HTTPS, POPS, IMAPS, LDAPS, ... est fondamental.

On rappelera aussi qu'en France, la CNIL (Commission Nationale de l'Informatique et des Libertés) encadre le rôle des administrateurs informatiques, le texte ci-dessous est extrait du *guide pratique pour les employeurs* de la CNIL[4] :

« *De même, les administrateurs de réseaux et systèmes, généralement tenus au*

[4]http://www.cnil.fr/fileadmin/documents/La_CNIL/publications/CNIL_GuideTravail.pdf

secret professionnel ou à une obligation de discrétion professionnelle, ne doivent pas divulguer des informations qu'ils auraient été amenés à connaître dans le cadre de leurs fonctions, et en particulier lorsque celles-ci sont couvertes par le secret des correspondances ou relèvent de la vie privée des utilisateurs et ne mettent en cause ni le bon fonctionnement technique des applications, ni leur sécurité, ni l'intérêt de l'entreprise. Ils ne sauraient non plus être contraints de le faire, sauf disposition législative particulière en ce sens. »

Chapter 4

Scanner les ports

Compte tenu de la vulnérabilité potentielle des protocoles, des applications et des systèmes d'exploitation (cf section 2.2 et section 2.3), il est très important qu'un administrateur ait une parfaite connaissance des services qui fonctionnent sur l'ensemble du parc de machines qu'il gère. Les scanners de ports sont des outils de découverte automatique des services disponibles. Ils permettent d'examiner l'ensemble des machines d'un réseau et produisent des rapports de synthèses très précis.

Comme dans le cas des renifleurs (cf. chapitre 3), ces outils sont également très intéressants pour les personnes mal intentionnées. Ils permettent à distance, de tenter de découvrir quels sont les services installés, quels sont les systèmes d'exploitation déployés, ... Le scanner de ports est un des outils de base des pirates informatiques.

La suite de ce chapitre est consacrée à `Nmap`[1] (Network Mapper) un des scanners de ports les plus utilisés.

4.1 `Nmap`

`Nmap` peut être utilisé pour scanner une machine, un sous-ensemble ou la totalité d'un réseau, et fournit un rapport d'analyse dont la table des *ports intéressants* est l'élément essentiel. Lorsqu'une machine est accessible, `Nmap` affecte un statut aux ports scannés, il peut prendre les valeurs suivantes :

- **fermé** : le port est accessible mais n'est pas en service (il n'y a pas d'application à l'écoute) ;
- **filtré** : le port est filtré par un dispositif de contrôle (pare-feu, ACLs de routeurs). `Nmap` ne peut pas déterminer s'il y a effectivement un service à

[1] http://insecure.org/nmap/

l'écoute ;

- **ouvert** : un service utilise ce port et répond aux sollicitations ;
- **ouvert ou filtré** : `Nmap` est incapable de déterminer si le port est ouvert ou filtré. Cet état se présente avec certains types d'analyses (cf. section 4.1.1) et dépend aussi de la configuration des pare-feux rencontrés (que font-ils lorsque un paquet est indésirable ? en général il ne répondent pas (filtré donc), dans certains cas ils émettent un paquet `RST`, qui a émis le paquet `RST` une application ou un pare-feu ?) ;
- **non filtré** : `Nmap` est incapable de déterminer si le port est ouvert ou fermé. Cet état se présente uniquement avec le type d'analyse `ACK` (cf. section 4.1.1) ;
- **fermé ou filtré** : `Nmap` est incapable de déterminer si le port est fermé ou filtré. Cet état se présente uniquement avec le type d'analyse `Idle` (cf. section 4.1.1).

4.1.1 Les différents types de *scan* de `Nmap`

`Nmap` permet d'effectuer des *scans* en utilisant différentes techniques issues de l'étude du comportement des machines respectant le RFC 793[2] (TCP). Parmi la douzaine de techniques de *scan* connues, on peut citer les suivantes :

- **SYN Stealth scan** : le *scan* SYN furtif est celui proposé par défaut. Il est très rapide car il n'effectue pas une connexion TCP complète. `Nmap` émet un paquet sur le port ciblé et attend la réponse qui peut être :
  - un paquet SYN/ACK qui indique que le port est ouvert ;
  - un paquet RST qui indique que le port est fermé ;
  - pas de réponse si le port est filtré.

  Dans tous les cas `Nmap` n'effectue pas la connexion complète et émet un paquet RST pour fermer la connexion. C'est pourquoi ce type de *scan* est également appelé demi-ouvert (*half open scan*).
- **Connect scan** : le *scan* le plus anodin. Il effectue un appel système *connect()* et se comporte donc comme un client TCP normal. Il ne nécessite pas de droits particuliers (contrairement au **SYN Stealth scan**) mais son exécution est plus lente ;
- **UDP port scan** : le *scan* des ports UDP ;
- consulter la documentation de tous les types de *scan* sur http://insecure.org/nmap/man/fr/man-port-scanning-techniques.html.

[2] http://www.rfc-editor.org/rfc/rfc793.txt

4.1.2 Quelques exemples de *scans*

Les exemples suivants utilisent `nmapfe`, l'interface X-Window de la commande `Nmap`.

4.1.2.1 Un *scan* sur un serveur ouvert

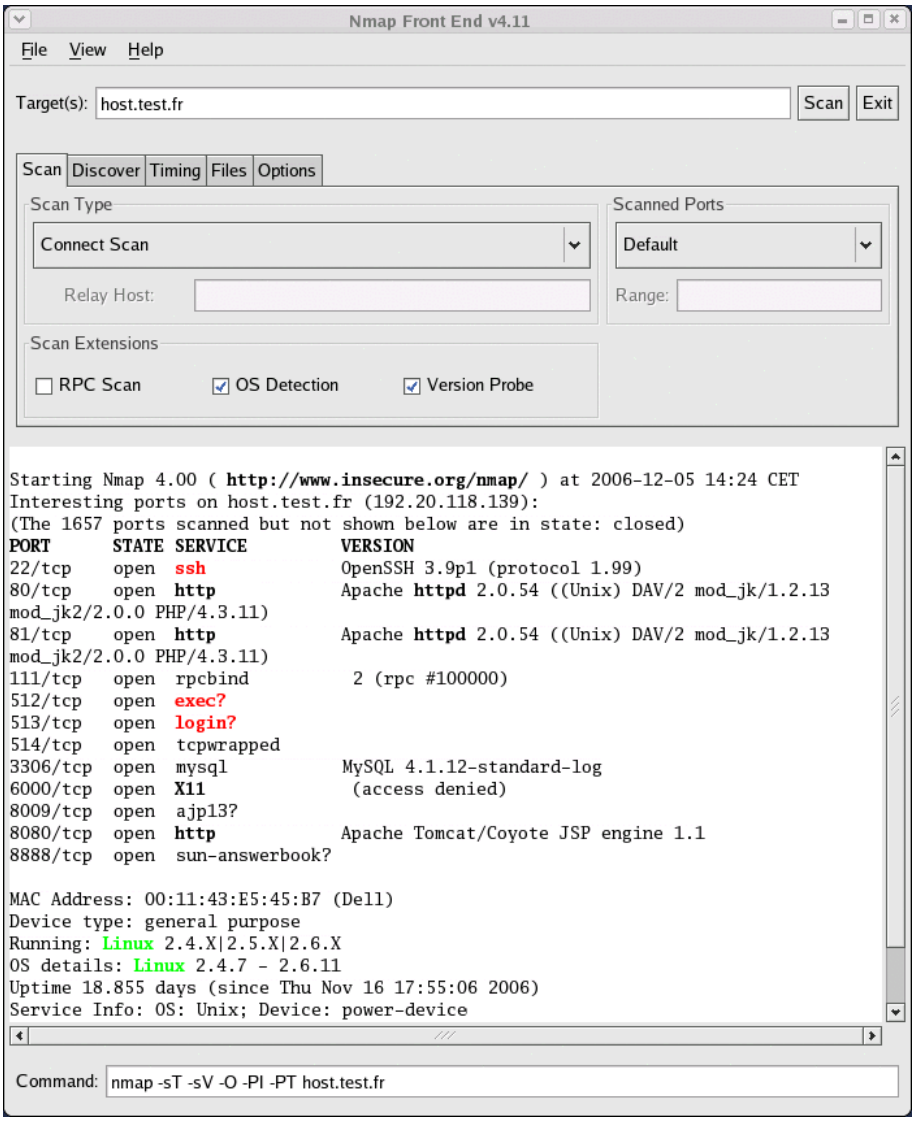


Figure 4.1: Un *scan* sur un serveur ouvert

Le résultat de ce **Connect scan** sur un serveur non protégé par un pare-feu révèle la version d'Apache utilisée (il est donc nécessaire de rendre Apache plus

furtif[3]). Par ailleurs on peut constater qu'un service SSH existe et que les services `rsh`, `rlogin` ne sont pas désarmés (il faut le faire).

4.1.2.2 Un *scan* sur un serveur protégé par un pare-feu

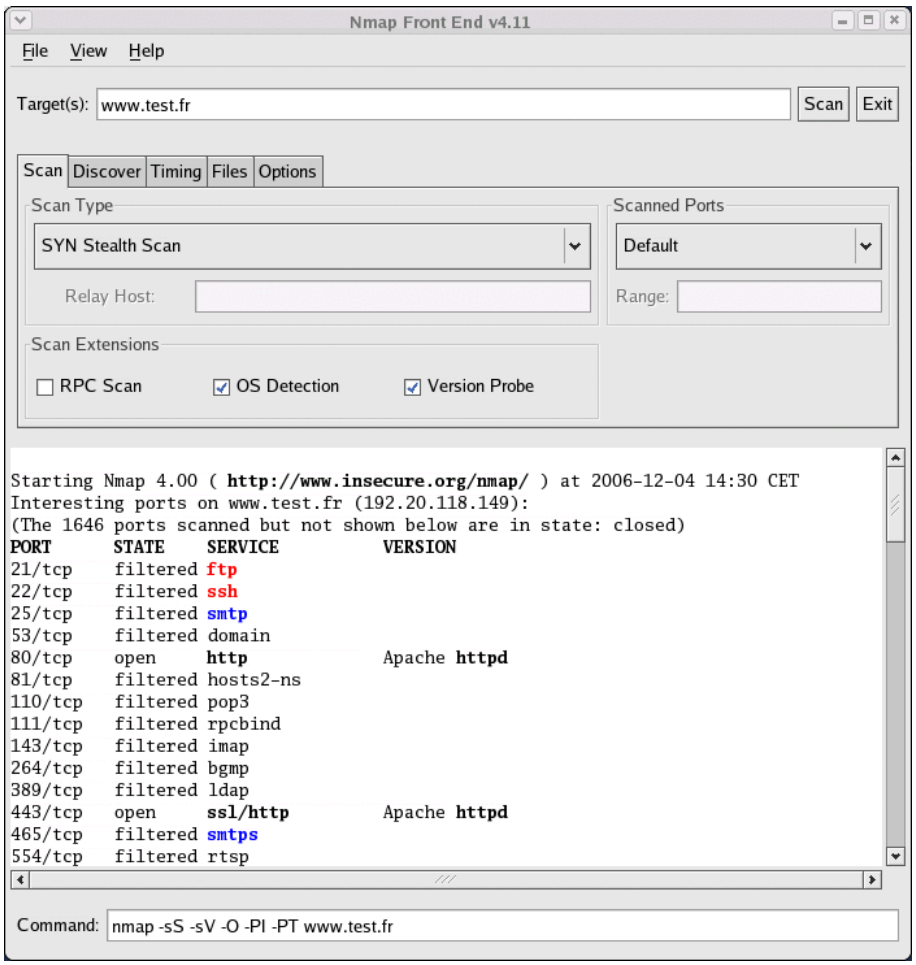


Figure 4.2: Un *scan* sur un serveur protégé par un pare-feu

L'exemple de la figure 4.2 montre le résultat d'un **SYN Stealth scan** sur un serveur qui héberge un service Web ouvert et d'autres services filtrés. On notera que malgré la sélection de l'option *Version probe* (afficher les versions de serveurs), Nmap ne peut indiquer quelle est la version d'Apache utilisée. Dans un bon réflexe, l'administrateur du serveur WWW a configuré de masquer cette information.

[3] `ServerTokens Prod` et `ServerSignature off` dans le fichier `httpd.conf`

4.2 Conclusion

Les scanners de ports sont des outils de base pour les pirates, il doivent donc l'être aussi pour les administrateurs de réseaux. Ceux-ci doivent soumettre régulièrement les réseaux qu'ils gèrent à une analyse complète. Notamment après l'installation de toute nouvelle machine (des services inutiles peuvent être ouverts) ou pour vérifier que les bonnes résolutions initiales sont toujours effectives.

L'usage d'un scanner de ports (tel `Nmap`) améliore la sécurité des réseaux car il permet aux administrateurs de suivre très précisémment l'état des services ouverts, c'est un élément important pour maîtriser les usages.

Chapter 5

Visualiser l'utilisation du réseau avec `ntop`

Une fois la politique de sécurité déployée (pare-feu, *access-lists* de VLAN, ...), après que les règles d'usages aient été maintes fois rappelées aux utilisateurs, il est important de pouvoir visualiser l'usage courant du réseau. La suite de ce chapitre présente ntop[1] qui répond à cet impératif.

5.1 Présentation générale

`ntop` est un outil de mesure et de caractérisation du trafic sur un réseau. Il fonctionne en temps réel et montre l'usage courant du réseau via une interface Web. `ntop` utilise la bibliothèque portable `libpcap` (comme `ethereal` et `tcpdump`) pour capturer les paquets sur une ou plusieurs interfaces réseaux.

`ntop` utilise les couches 2 et 3 pour présenter par machines, une synthèse de tous les protocoles utilisés (`TCP`, `UDP`, `ICMP`, `ICMPv6`, `DLC`, `IPX`, `Decnet`, `(R)ARP`, `AppleTalk`, `NetBios`, `OSI`, `IPv6`, `STP`, `IPSEC`, `OSPF`, `IGMP`). Dans le cas spécifique du protocole IP, le trafic est caractérisé par protocoles applicatifs incluant les suivants : `FTP`, `HTTP`, `DNS`, `Telnet`, `NBios-IP`, `Mail`, `DHCP-BOOTP`, `SNMP`, `NNTP`, `NFS/AFS`, `VoIP`, `X11`, `SSH`, `Gnutella`, `Kazaa`, `WinMX`, `DC++`, `eDonkey`, `BitTorrent`, `Messenger`.

Parmi les fonctionnalités de `ntop`, on peut citer les suivantes :

- graphes du débit global du réseau pour le mois en cours, les dernières 24 heures, les dernières 60 minutes, les dernières 10 minutes ;
- caractérisation du trafic par machine et protocole en séparant éventuellement

[1] http://www.ntop.org/

les machines locales et les machines distantes. Par défaut, les résultats sont classés par ordre décroissant de trafic constaté ;

- détails de la consommation de chaque machine sur les dernières 24 heures permettant de mettre en évidence les usages (détails des connexions, sessions TCP et UDP actives, ...) ;
- statistiques d'usage par domaines avec caractérisation du trafic par protocoles ;
- ...

5.2 Déployer `ntop`

`ntop` capture et analyse les paquets qu'il voit passer. Son positionnement determine donc les flux qui seront *monitorés*. Pour la mise en place d'un service `ntop` (une sonde) il est conseillé d'utiliser deux interfaces : l'une pour l'accès à la machine et au service WWW de consultation, l'autre pour capturer le trafic. Par ailleurs, il est indispensable de configurer l'interface utilisée pour capturer le trafic dans une plage d'adresse IP non routable.

5.2.1 Quelques options de démarrage de `ntop`

Parmi les options de démarrage de ntop, on peut mentionner les suivantes :

- [**-i interfaces**] : une liste d'interface (séparées par des virgules) de capture (`-i eth1, eth2`) ;
- [**-m reseaux locaux**] : une liste de réseaux (séparés par des virgules) pour indiquer quels sont les réseaux locaux afin de procéder à la séparation du trafic (*local*, *distant*). Par défaut `ntop` trouve cette information en interrogeant la configuration de l'interface de capture, si elle configurée dans une plage d'adresse non routable, alors en l'absence de précision, tout le trafic est considéré comme *distant* ;
- [**-o**] : dans certains cas, notamment lorsque l'interface de capture reçoit du trafic issu d'un port configuré en SPAN (cf. section 5.2.2), il est nécessaire de positionner cette option (*Situations which may require this option include port/VLAN mirror, ...*) ;

5.2.2 Comment placer la sonde

Dans le cadre d'un réseau commuté, par défaut, une sonde ne peut avoir accès qu'au trafic destiné à la machine qui l'héberge. Ce n'est évidemment pas ce qui

est en général recherché. Pour élargir le champ de vision, il est nécessaire de choisir un des trois dispositifs suivants :

- utiliser un concentrateur (*hub*) comme point de passage obligé du trafic à analyser. Cette méthode est simple, elle peut servir pour effectuer quelques essais. Elle ne peut évidemment pas être utilisée en production, car outre les problèmes de sécurité liés au partage du média, le mode de fonctionnement *half-duplex* peut rendre inopérants certains services (diffusion de vidéos en *streaming*, ...) ;
- configurer le port utilisé par l'interface de capture en mode SPAN (*Switch Port ANalyser*). Dans ce cas, le port configuré va recevoir un duplicata du trafic des ports à analyser. La mise en œuvre de cette fonctionnalité est très simple, elle dépend évidemment des commutateurs utilisés. L'exemple suivant montre les commandes nécessaires pour configurer des commutateurs CISCO (dans les deux cas l'interface `Fa0/11` reçoit un duplicata du trafic des ports `Fa0/12` et `Fa0/13`) :
  - **Commutateurs 2950**
    ```
    monitor session 1 source interface Fa0/11
    monitor session 1 destination interface Fa0/12
    monitor session 1 destination interface Fa0/13
    ```
  - **Commutateurs 3500**
    ```
    !
    interface FastEthernet0/11
    port monitor FastEthernet0/12
    port monitor FastEthernet0/13
    switchport access vlan x
    !
    ```
- utiliser un dispositif qui duplique le signal (*network tap*)[2] pour permettre de réaliser une dérivation du réseau sur laquelle la sonde pourra être installée. Cette solution autorise de placer des sondes totalement furtives (invisibles du reste du réseau), elle est à privilégier dans les environnements très sensibles.

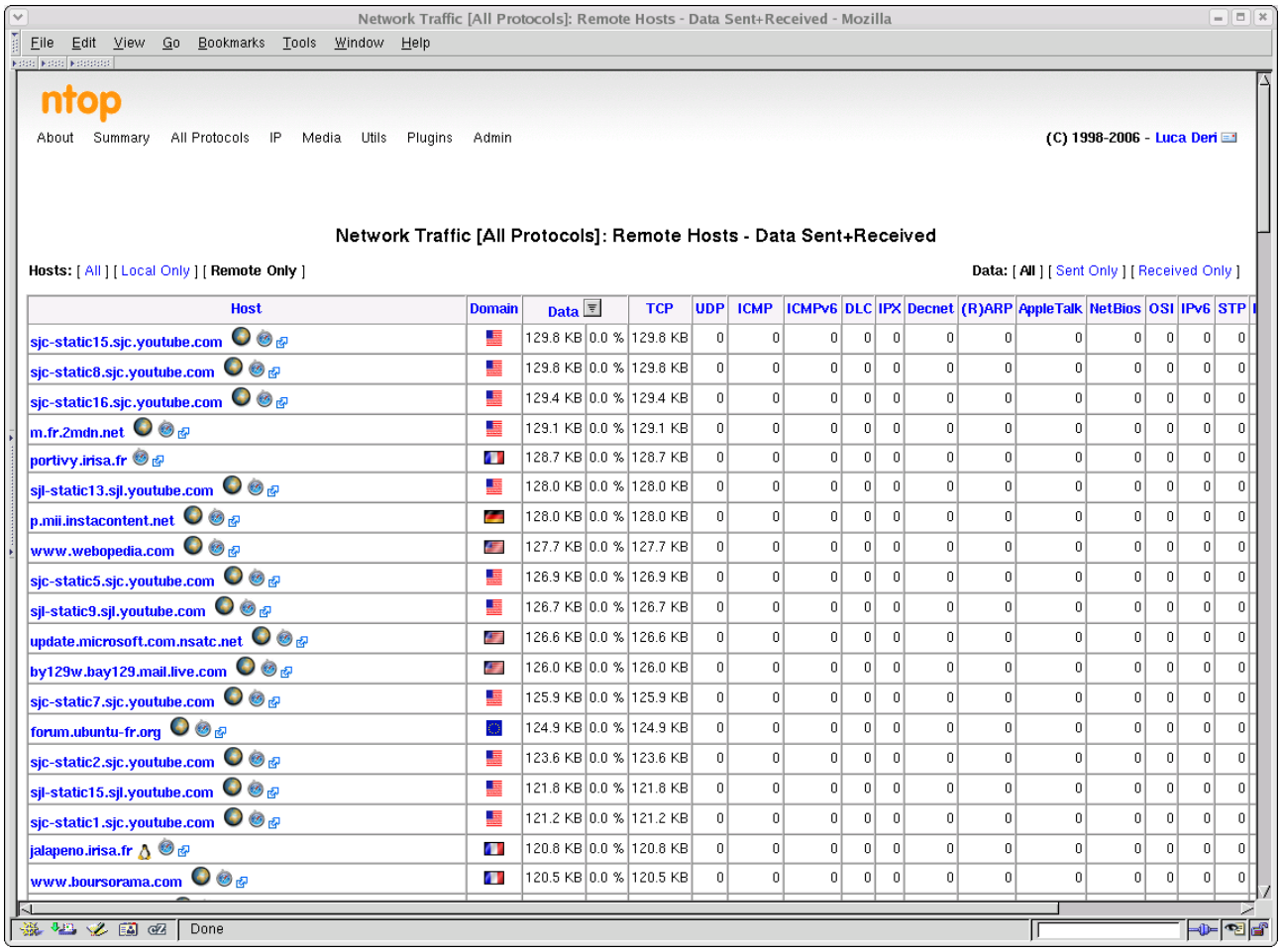
5.2.3 Quelques exemples d'utilisation de `ntop`

Les figures suivantes montrent les résultats obtenus par une sonde placée à l'extrémité d'un réseau (juste avant le pare-feu). Le niveau de détail obtenu permet d'atteindre l'objectif d'une meilleure connaissance des usages.

Les figures 5.1 et 5.2 montrent le trafic avec des machines distantes caractérisé par protocoles. Les figures 5.3 et 5.4 illustrent le détail qui est obtenu lorsqu'un

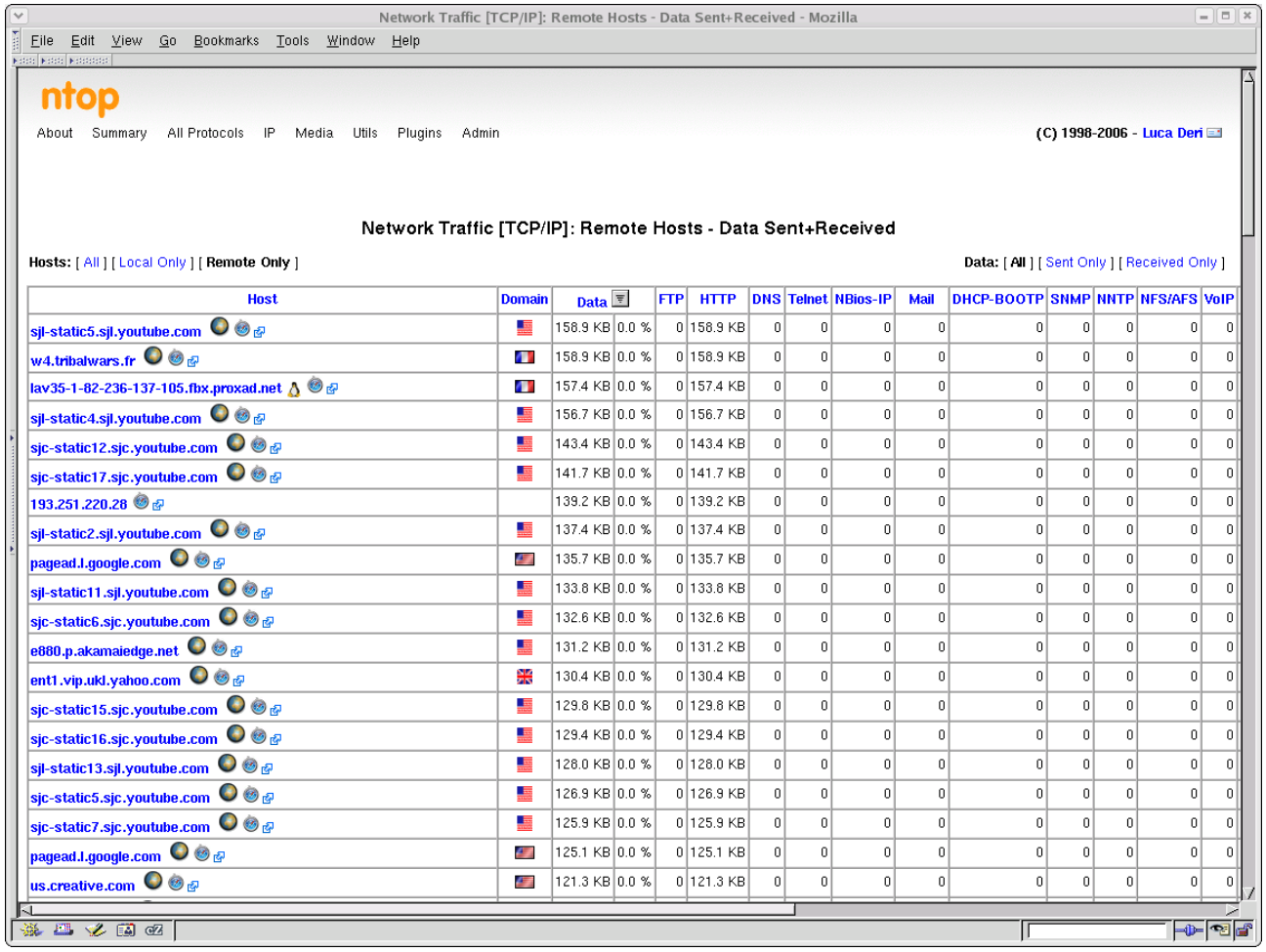
[2] cf. http://www.netopics.com

administrateur s'intéresse particulièrement à l'activité (présente et passée) d'une machine. La figure 5.5 permet de constater si l'activité est conforme aux prévisions, notamment la nuit.



Host	Domain	Data		TCP	UDP	ICMP	ICMPv6	DLC	IPX	Decnet	(R)ARP	AppleTalk	NetBios	OSI	IPv6	STP
sjc-static15.sjc.youtube.com		129.8 KB	0.0 %	129.8 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjc-static8.sjc.youtube.com		129.8 KB	0.0 %	129.8 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjc-static16.sjc.youtube.com		129.4 KB	0.0 %	129.4 KB	0	0	0	0	0	0	0	0	0	0	0	0
m.fr.2mdn.net		129.1 KB	0.0 %	129.1 KB	0	0	0	0	0	0	0	0	0	0	0	0
portivy.irisa.fr		128.7 KB	0.0 %	128.7 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjl-static13.sjl.youtube.com		128.0 KB	0.0 %	128.0 KB	0	0	0	0	0	0	0	0	0	0	0	0
p.mii.instacontent.net		128.0 KB	0.0 %	128.0 KB	0	0	0	0	0	0	0	0	0	0	0	0
www.webopedia.com		127.7 KB	0.0 %	127.7 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjc-static5.sjc.youtube.com		126.9 KB	0.0 %	126.9 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjl-static9.sjl.youtube.com		126.7 KB	0.0 %	126.7 KB	0	0	0	0	0	0	0	0	0	0	0	0
update.microsoft.com.nsatc.net		126.6 KB	0.0 %	126.6 KB	0	0	0	0	0	0	0	0	0	0	0	0
by129w.bay129.mail.live.com		126.0 KB	0.0 %	126.0 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjc-static7.sjc.youtube.com		125.9 KB	0.0 %	125.9 KB	0	0	0	0	0	0	0	0	0	0	0	0
forum.ubuntu-fr.org		124.9 KB	0.0 %	124.9 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjc-static2.sjc.youtube.com		123.6 KB	0.0 %	123.6 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjl-static15.sjl.youtube.com		121.8 KB	0.0 %	121.8 KB	0	0	0	0	0	0	0	0	0	0	0	0
sjc-static1.sjc.youtube.com		121.2 KB	0.0 %	121.2 KB	0	0	0	0	0	0	0	0	0	0	0	0
jalapeno.irisa.fr		120.8 KB	0.0 %	120.8 KB	0	0	0	0	0	0	0	0	0	0	0	0
www.boursorama.com		120.5 KB	0.0 %	120.5 KB	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.1: ntop caractérise les protocoles utilisés



Host	Domain	Data	FTP	HTTP	DNS	Telnet	NBios-IP	Mail	DHCP-BOOTP	SNMP	NNTP	NFS/AFS	VoIP
sjl-static5.sjl.youtube.com		158.9 KB 0.0 %	0	158.9 KB	0	0	0	0	0	0	0	0	0
w4.tribalwars.fr		158.9 KB 0.0 %	0	158.9 KB	0	0	0	0	0	0	0	0	0
lav35-1-82-236-137-105.fbx.proxad.net		157.4 KB 0.0 %	0	157.4 KB	0	0	0	0	0	0	0	0	0
sjl-static4.sjl.youtube.com		156.7 KB 0.0 %	0	156.7 KB	0	0	0	0	0	0	0	0	0
sjc-static12.sjc.youtube.com		143.4 KB 0.0 %	0	143.4 KB	0	0	0	0	0	0	0	0	0
sjc-static17.sjc.youtube.com		141.7 KB 0.0 %	0	141.7 KB	0	0	0	0	0	0	0	0	0
193.251.220.28		139.2 KB 0.0 %	0	139.2 KB	0	0	0	0	0	0	0	0	0
sjl-static2.sjl.youtube.com		137.4 KB 0.0 %	0	137.4 KB	0	0	0	0	0	0	0	0	0
pagead.l.google.com		135.7 KB 0.0 %	0	135.7 KB	0	0	0	0	0	0	0	0	0
sjl-static11.sjl.youtube.com		133.8 KB 0.0 %	0	133.8 KB	0	0	0	0	0	0	0	0	0
sjc-static6.sjc.youtube.com		132.6 KB 0.0 %	0	132.6 KB	0	0	0	0	0	0	0	0	0
e880.p.akamaiedge.net		131.2 KB 0.0 %	0	131.2 KB	0	0	0	0	0	0	0	0	0
ent1.vip.ukl.yahoo.com		130.4 KB 0.0 %	0	130.4 KB	0	0	0	0	0	0	0	0	0
sjc-static15.sjc.youtube.com		129.8 KB 0.0 %	0	129.8 KB	0	0	0	0	0	0	0	0	0
sjc-static16.sjc.youtube.com		129.4 KB 0.0 %	0	129.4 KB	0	0	0	0	0	0	0	0	0
sjl-static13.sjl.youtube.com		128.0 KB 0.0 %	0	128.0 KB	0	0	0	0	0	0	0	0	0
sjc-static5.sjc.youtube.com		126.9 KB 0.0 %	0	126.9 KB	0	0	0	0	0	0	0	0	0
sjc-static7.sjc.youtube.com		125.9 KB 0.0 %	0	125.9 KB	0	0	0	0	0	0	0	0	0
pagead.l.google.com		125.1 KB 0.0 %	0	125.1 KB	0	0	0	0	0	0	0	0	0
us.creative.com		121.3 KB 0.0 %	0	121.3 KB	0	0	0	0	0	0	0	0	0

Figure 5.2: ntop caractérise les protocoles applicatifs utilisés

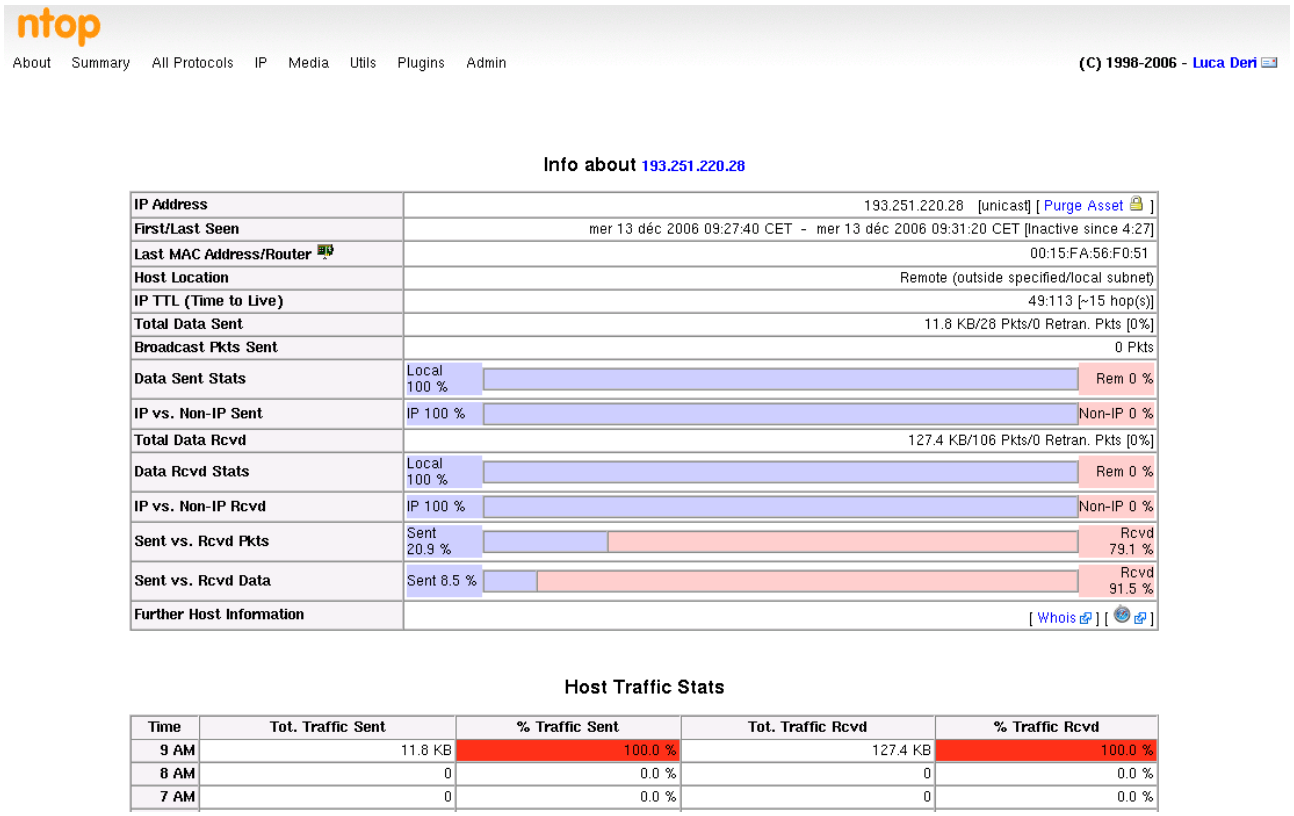


Figure 5.3: ntop détaille l'activité d'une machine (1)

TCP/UDP Service/Port Usage

IP Service	Port	# Client Sess.	Last Client Peer	# Server Sess.	Last Server Peer
http	80	134/132.1 KB	148.60.4.19		

TCP/UDP - Traffic on Other Ports

Client Port	Server Port
• 51532	

TCP/UDP Recently Used Ports

Client Port	Server Port
• 51532	

2124 Active TCP/UDP Sessions

Client	Server	Data Sent	Data Rcvd	Active Since	Last Seen	Duration	Inactive	Latency	L7 Proto	Note
193.251.220.28 :48949	brutus :http	861	47.4 KB	mer 13 déc 2006 09:27:40 CET	mer 13 déc 2006 09:28:21 CET	41 sec	9:26			ACK PUSH
193.251.220.28 :49470	brutus :http	166	1014	mer 13 déc 2006 09:28:31 CET	mer 13 déc 2006 09:28:31 CET	0 sec	9:16			ACK PUSH
193.251.220.28 :49546	brutus :http	875	12.3 KB	mer 13 déc 2006 09:28:34 CET	mer 13 déc 2006 09:28:51 CET	17 sec	8:56			ACK PUSH
193.251.220.28 :49566	brutus :http	454	19.3 KB	mer 13 déc 2006 09:28:34 CET	mer 13 déc 2006 09:28:55 CET	21 sec	8:52			ACK PUSH
193.251.220.28 :49898	brutus :http	436	2.5 KB	mer 13 déc 2006 09:29:08 CET	mer 13 déc 2006 09:29:08 CET	0 sec	8:39			ACK PUSH
193.251.220.28 :50016	brutus :http	3.1 KB	11.4 KB	mer 13 déc 2006 09:29:16 CET	mer 13 déc 2006 09:29:39 CET	23 sec	8:08			ACK PUSH
193.251.220.28 :50152	brutus :http	2.2 KB	5.1 KB	mer 13 déc 2006 09:29:25 CET	mer 13 déc 2006 09:29:37 CET	12 sec	8:10			ACK PUSH
193.251.220.28 :51352	brutus :http	904	828	mer 13 déc 2006 09:31:10 CET	mer 13 déc 2006 09:31:18 CET	8 sec	6:29			ACK PUSH
193.251.220.28 :51353	brutus :http	461	440	mer 13 déc 2006 09:31:10 CET	mer 13 déc 2006 09:31:10 CET	0 sec	6:37			ACK PUSH
193.251.220.28 :51361	brutus :http	904	828	mer 13 déc 2006 09:31:10 CET	mer 13 déc 2006 09:31:18 CET	8 sec	6:29			ACK PUSH
193.251.220.28 :51530	brutus :http	461	440	mer 13 déc 2006 09:31:20 CET	mer 13 déc 2006 09:31:20 CET	0 sec	6:27			ACK PUSH
193.251.220.28 :51532	brutus :http	461	440	mer 13 déc 2006 09:31:20 CET	mer 13 déc 2006 09:31:20 CET	0 sec	6:27			ACK PUSH

The color of the host link indicates how recently the host was FIRST seen
0 to 5 minutes 5 to 15 minutes 15 to 30 minutes 30 to 60 minutes 60+ minutes

Figure 5.4: ntop détaille l'activité d'une machine (2)

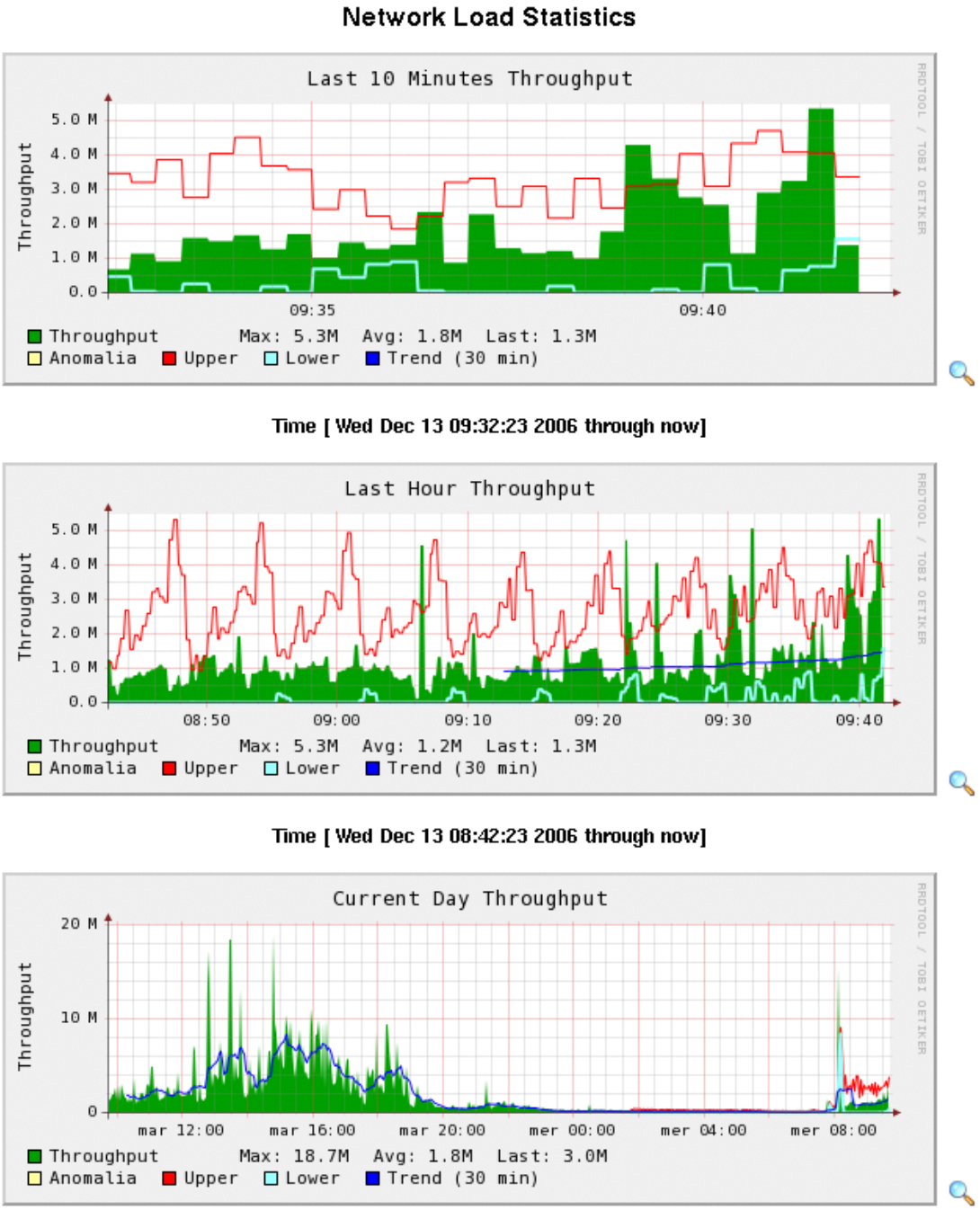


Figure 5.5: ntop présente des statistiques d'utilisation

Chapter 6

Détecter les vulnérabilités

Comme indiqué au chapitre 2, les protocoles, les applications et les systèmes d'exploitation peuvent présenter des vulnérabilités. Dès qu'une vulnérabilité est recensée, un bulletin d'alerte est diffusé (cf. section 2.4), la disponibilité d'*exploits*[1] peut alors être très rapide.

Des outils permettent de détecter les vulnérabilités, ils peuvent même être configurés pour être intrusifs, ils sont à la disposition des administrateurs de réseaux pour évaluer les risques encourus. Ces outils sont également accessibles aux personnes mal intentionnées, il est donc très important de les connaître et de les utiliser régulièrement.

La suite de ce chapitre présente le scanner de vulnérabilités `Nessus` ainsi que l'outil d'exploitation de vulnérabilités `Metasploit`.

6.1 Scanner les vulnérabilités avec `Nessus`

`Nessus`[2] est un outil qui automatise la découverte de vulnérabilités connues sur les machines d'un réseau. C'est un produit commercial diffusé par la société `TENABLE Network Security`. Il peut toutefois être utilisé gratuitement en utilisant une base des vulnérabilités dont la mise à jour est décalée[3] d'une semaine. `Nessus` est organisé en mode client/serveur : le serveur effectue les tests tandis que les clients sont chargés de la configuration et de la présentation des résultats. Les principales fonctionnalités de `Nessus` sont les suivantes :

- mise à jour automatique de la base des vulnérabilités ;

[1] traduction de l'anglais *sploit* : *shortname for exploit or the use of a software bug to accomplish something not intended by the designers.*

[2] cf. http://www.nessus.org

[3] les administrateurs apprécieront en fonction de la sensibilité des sites gérés si un délai est envisageable ou pas.

- disponibilités des tests sous formes de *plugins* développés en NASL (Nessus Attack Scripting Language). NASL permet de développer rapidement des scripts de tests de vulnérabilités, il propose des méta-instructions pour interroger des serveurs HTTP, FTP, ... ou pour fabriquer des paquets IP, TCP, UDP ou ICMP, ... ;
- reconnaissance automatique des services attachés aux ports scannés ;
- sélection par l'utilisateur des types de détection à mettre en œuvre (avec ou sans activation des options intrusives, ...).

6.1.1 Quelques exemples d'utilisation avec `Nessus`

Le client `Nessus` est disponible dans tous les environnements (Windows, Mac OS X, mode ligne sous Unix, X11), les tests suivants ont été effectués à l'aide de `NessusClient` (le client `Nessus` pour X11).

La figure 6.1 montre l'écran d'accueil de `NessusClient` et notamment les options générales utilisées pour les *scans* (*safe checks* ici pour ne pas risquer de corrompre une machine). La figure 6.2 montre le résultat d'une analyse sur un serveur où un service FTP très vulnérable est installé. `Nessus` indique la solution (mettre à jour) ainsi que les références des bulletins d'alertes qui correspondent et qu'il est nécessaire de consulter[4] :

```
National Vulnerability DataBase
Original release date: 6/18/2001
Last revised: 10/20/2005
Source: US-CERT/NIST

Overview

Heap overflow in FTP daemon in Solaris 8 allows remote attackers to
execute arbitrary commands by creating a long pathname and calling
the LIST command, which uses glob to generate long strings.

Impact

CVSS Severity: 10.0 (High) Approximated
Range: Remotely exploitable
Impact Type: Provides administrator access
```

[4] http://nvd.nist.gov/nvd.cfm?cvename=CVE-2001-0249

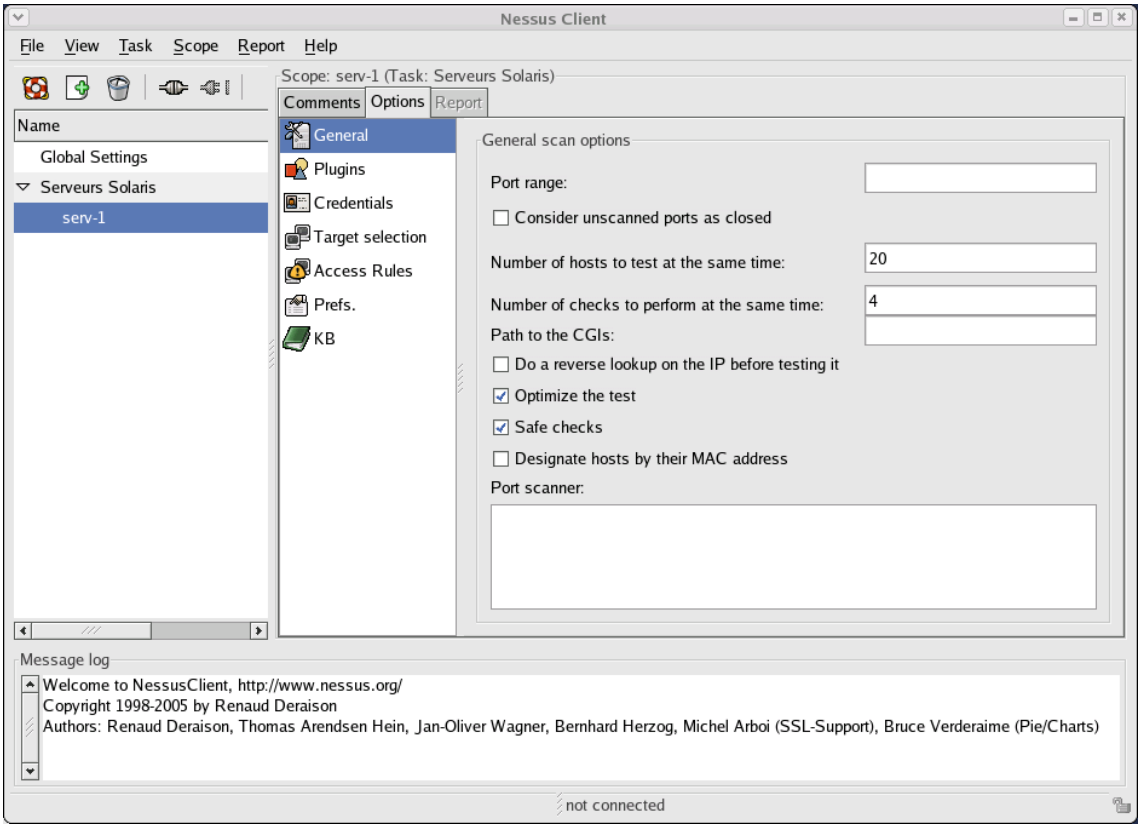


Figure 6.1: écran d'accueil de `NessusClient`

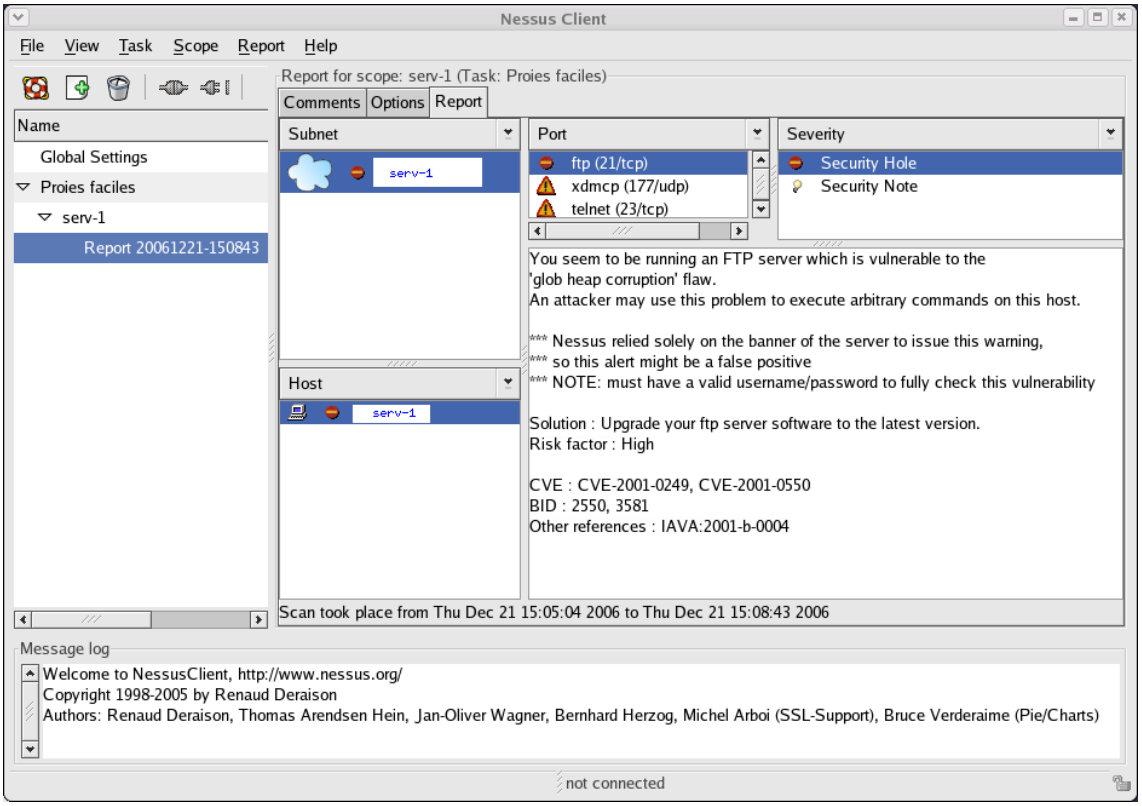


Figure 6.2: Le résultat d'un *scan* `Nessus`

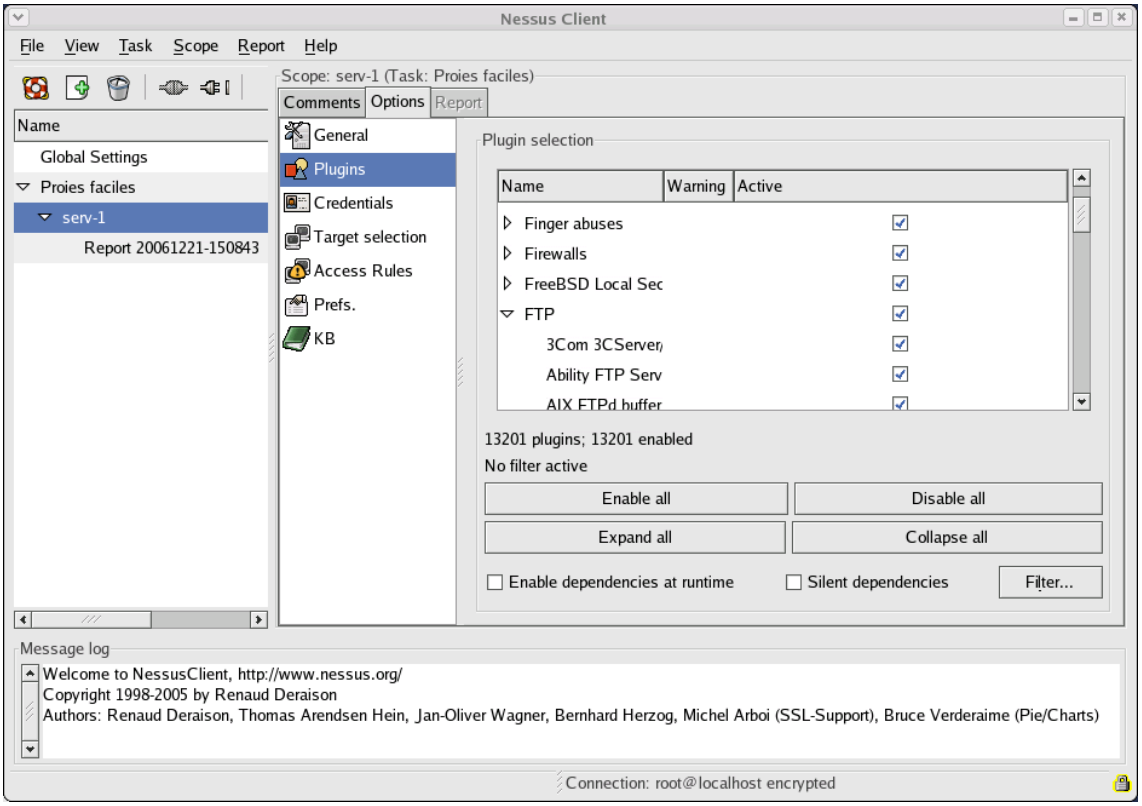


Figure 6.3: Le choix des vulnérabilités à tester avec `Nessus`

6.1.2 Conclusion

`Nessus` est très largement utilisé. Il recense plus de 13000 *plugins* dont le code est assez simple à comprendre (cf. */opt/nessus/lib/nessus/plugins* sous Linux). Comme tous les scanners, `Nessus` peut engendrer des faux positifs (des alertes alors que le produit est sain), une connaissance de NASL est alors utile pour déterminer si l'alerte est justifiée. Dans ce cas il convient d'effectuer les seuls tests qui ont généré l'alerte en y plaçant des traces. La commande `nasl` permet de soumettre un test rapidement comme dans l'exemple suivant :

```
nasl -T /tmp/trace -t serv-1 /opt/nessus/lib/nessus/plugins/mod_auth_any.nasl

Ouf, ce serveur n'est pas vulnerable
```

Le code modifié de `mod_auth_any.nasl` est le suivant :

```
#
# The script code starts here
#
include("http_func.inc");
include("http_keepalive.inc");
include("global_settings.inc");

port = get_http_port(default:80);

if ( report_paranoia < 2 )
{
 banner = get_http_banner(port port);
 if ( ! banner || "Apache" >!< banner ) exit(0);
}

pages = get_kb_list(string("www/", port, "/content/auth_required"));
if(isnull(pages)) {
   display ("Ouf, ce serveur n'est pas vulnerable");
   exit(0);
}
pages = make_list(pages);

foreach file (pages)
{
 req = http_get(item file, port port);
 auth = egrep(pattern:"^Authorization", string:req);
 if(auth) req = req - auth;

 res = http_keepalive_send_recv(port:port, data:req);
 ....
```

L'utilisation de `Nessus` est très intéressante, les premiers *scans* sont en général surprenants, il faut prévoir beaucoup de temps pour les analyser. Pour cette raison, il est recommandé d'effectuer les analyses en classifiant les serveurs et en évitant de lancer des *scans* sur la globalité d'un réseau (l'administrateur serait alors noyé sous un déluge d'informations).

6.2 Exploiter les vulnérabilités

Bien qu'il puisse être configuré pour être intrusif, `Nessus` reste surtout un outil de détection de vulnérabilités. Des outils nettement plus agressifs sont également disponibles, ils recensent et diffusent des méthodes pour exploiter les failles connues. La suite de cette section présente `Metasploit` et `SecWatch.org` qui peuvent

sembler ambigus car ils vulgarisent des techniques d'intrusion. Le concept de base de ces outils est que si des pirates disposent d'outils simples pour attaquer, il est alors très important que les administrateurs de réseaux soient informés et réactifs

6.2.1 Metasploit

Metasploit[5] est une base de tests d'intrusions. Les *exploits* sont développés sous forme de modules Perl, ils partagent un même cadre comme dans l'exemple suivant :

```
package Msf::Exploit::mssql2000_resolution;
use base "Msf::Exploit";
use strict;
use Pex::Text;

my $advanced = { };
my $info =
  {
        'Name'    => 'MSSQL 2000/MSDE Resolution Overflow',
        'Version' => '$Rev: 3818 $',
        'Authors' => [ 'H D Moore <hdm [at] metasploit.com>', ],
        'Arch'  => [ 'x86' ],
        'OS'    => [ 'win32', 'win2000' ],
        'Priv'  => 1,
        'AutoOpts' => { 'EXITFUNC' => 'process' },
        'UserOpts'  =>
          {
                'RHOST' => [1, 'ADDR', 'The target address'],
                'RPORT' => [1, 'PORT', 'The target port', 1434],
          },
        'Payload'  =>
          {
                'Space'  => 512,
                'BadChars'  => "\x00\x3a\x0a\x0d\x2f\x5c",
          },
        'Description'  => Pex::Text::Freeform(qq{
        This is an exploit for the SQL Server 2000 resolution
        service buffer overflow. This overflow is triggered by
        sending a udp packet to port 1434 which starts with 0x04 and
        is followed by long string terminating with a colon and a
        number. This module should work against any vulnerable SQL
        Server 2000 or MSDE install (pre-SP3).
}),
```

[5] http://www.metasploit.org

La dernière version de `Metasploit` comporte 164 *exploits*, ils sont classés par architecture, système d'exploitation et application. Les paramètres principaux d'un *exploit* sont la machine cible (RHOST) et le port cible (RPORT), il convient en outre de stipuler une sentence (*payload*) à exécuter si la tentative d'intrusion est réussie. Les sentences sont développées indépendamment des *exploits*, il existe donc, en général, plusieurs sentences pour un même *exploit*. Parmi les sentences disponibles pour un *exploit* en environnement Linux, on peut citer les suivantes :

- exécution d'un interpréteur de commandes ;
- ajout d'un utilisateur ;
- exécution d'une commande ;
- démarrage d'un tunnel SSH (*reverse shell*).

`Metasploit` propose plusieurs interfaces d'utilisation : `msfconsole` et `msfcli` en mode lignes de commandes et `msfweb` qui est adaptée à l'utilisation d'un navigateur Web. Les 3 exemples des figures 6.4, 6.5 et 6.6 ont été réalisés avec `msfweb`, ils montrent respectivement la sélection des *exploits* de l'environnement Solaris (figure 6.4), l'introduction des paramètres d'un *exploit* (figure 6.5) et la preuve de l'intrusion (figure 6.6).

L'*exploit* choisi ici à titre d'exemple (*Arbitrary Read File*) ne propose pas de choix de sentence car il ne s'agit pas d'une intrusion complète, mais simplement d'un accès illicite à des fichiers protégés (*/etc/shadow* ici).

`Metasploit` permet d'ajouter de développer de nouveaux *exploits*, pour plus d'informations consulter le contenu du répertoire `sdk` d'une installation.

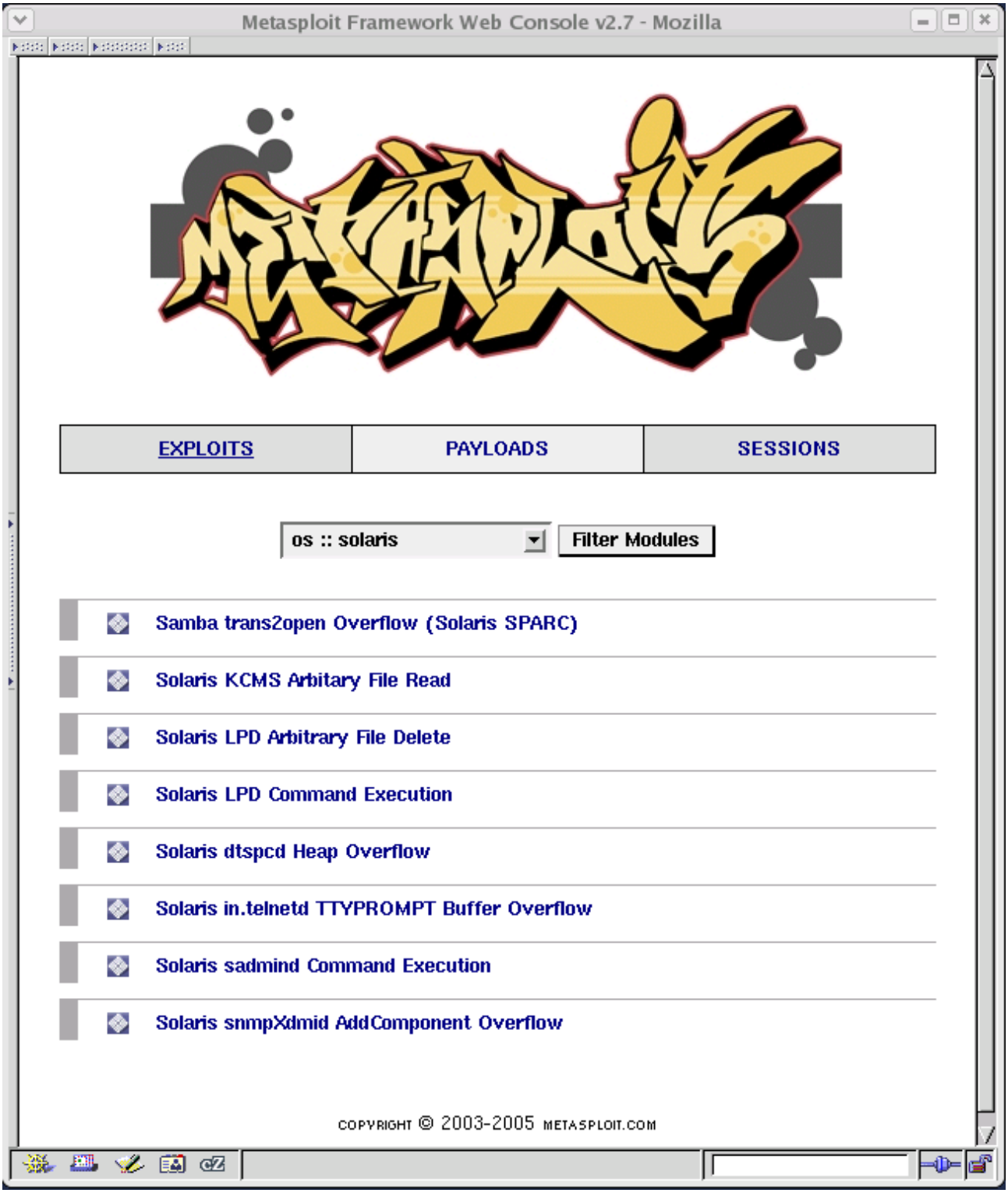


Figure 6.4: Le choix des exploits avec `msfweb`

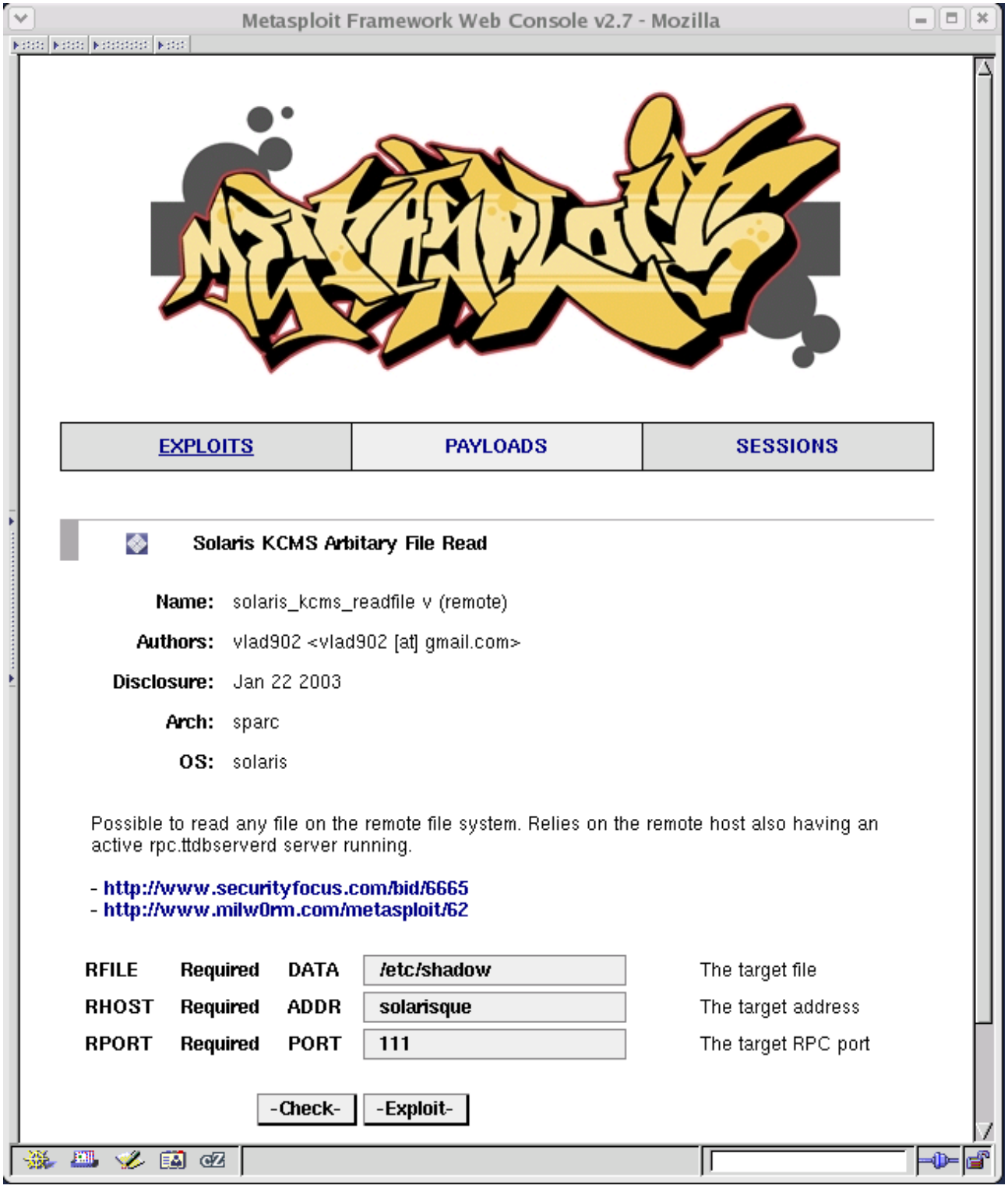


Figure 6.5: Les paramètres de l'exploit `solaris_kcms_readfile`

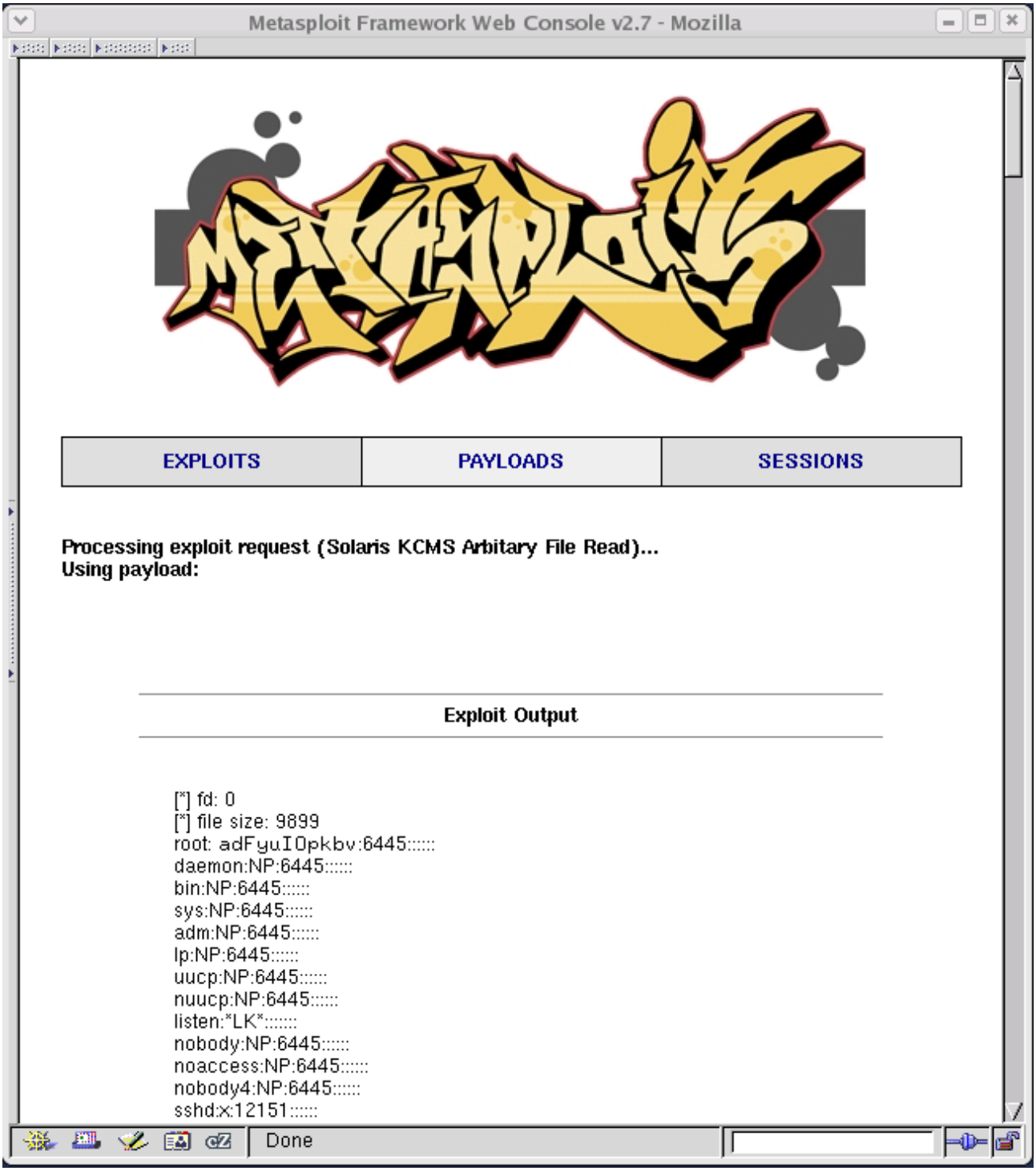


Figure 6.6: Le résultat de l'exploit `solaris_kcms_readfile`

6.2.2 SecWatch.org

http://www.secwatch.org propose une archive des vulnérabilités et des *exploits* connus (cf. figure 6.7).

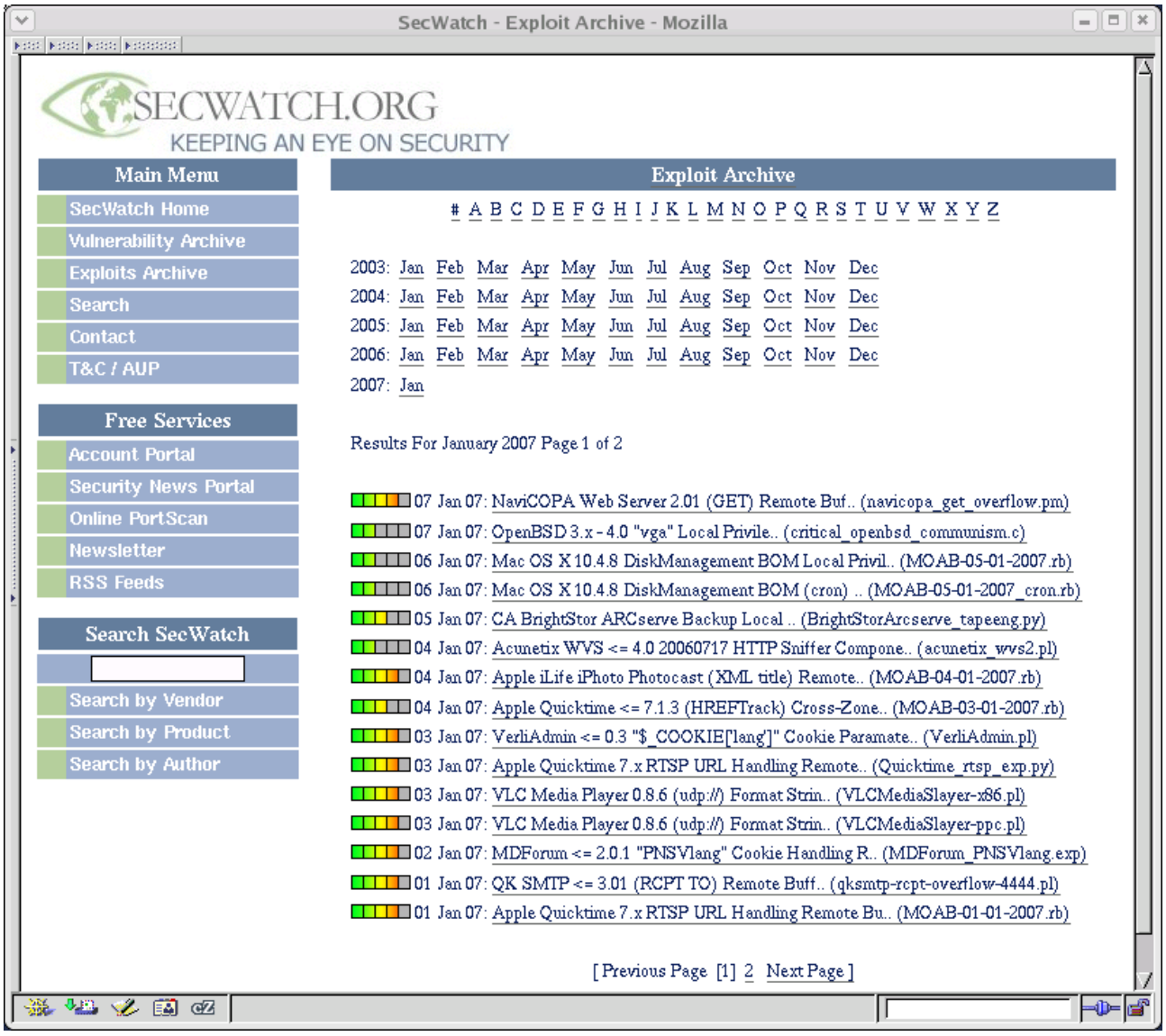


FIG. 6.7 – Les *exploits* du mois de janvier 2007

6.3 Conclusion

Les outils présentés dans ce chapitre sont à la disposition de tous, il est indispensable que les administrateurs de réseaux les connaissent. L'utilisation règulière de `Nessus` ne dispense pas de faire le tour des intrusions proposées par `Metasploit` ou `SecWatch.org` : la faille *Arbitrary Read File* présentée en exemple (cf. section 6.2.1) n'a pas été détectée. Comme précisé en section 2.4, il est nécessaire de consulter journellement les avis et alertes diffusés par le CERTA.

Chapitre 7

Détecter les attaques

`Nessus` permet de détecter des vulnérabilités répertoriées, des outils tels `Metasploit` montrent encore plus les risques encourus, mais comment être informé si ces outils (ou d'autres) servent lors de tentavives d'intrusions ?

Les détecteurs d'intrusions répondent à cette question, ils permettent de détecter des activités anormales sur un réseau[1] ou sur une machine[2].

Les HIDS détectent les activités suspectes en effectuant des études comportementales (analyse des journaux du système et des applications) ou en vérifiant l'intégrité des fichiers ainsi que celle du noyau. Les NIDS agissent comme des sondes qui analysent tous les paquets pour les comparer à une base de signatures d'attaques connues et répertoriées. La suite de ce chapitre présente `snort`, un des NIDS les plus utilisés.

7.1 snort

`snort`[3] est un NIDS développé sous licence GPL par la société Sourcefire[4]. `snort` s'appuie sur une bibliothèque de signatures dont la distribution et la mise à jour nécessitent une souscription annuelle[5] de 399 $.

`snort` capture et analyse les paquets qu'il voit passer et peut être utilisé comme *sniffer* (avec ou sans écriture dans un journal) ou comme NIDS.

Le fonctionnement de `snort` comme NIDS est basé un fichier de configuration

[1] Les NIDS (Network based Intrusion Detection System).
[2] Les HIDS (Host based Intrusion Detection System).
[3] http://www.snort.org
[4] http://www.sourcefire.com
[5] La base des signatures est accessible gratuitement avec un différé d'un mois, comme dans le cas de `Nessus` (cf. 6.1), les administrateurs apprécieront en fonction de la sensibilité des sites gérés si le délai est envisageable.

(`snort.conf`) ainsi qu'une base de signatures (les règles). Parmi les options de `snort` configurables dans `snort.conf`, on peut citer les suivantes :

- `HOME_NET` : une liste des réseaux à surveiller ([10.1.1.0/24,192.168.1.0] par exemple)
- `DNS_SERVERS`, `HTTP_SERVERS`, ... : une liste des serveurs à surveiller. Il n'est pas utile de traiter les attaques qui sont sans objets, i.e qui ciblent des machines n'hébergeant pas le service attaqué.
- RULE_PATH : le répertoire contenant la base des signatures (`/usr/local/etc/snort/rules` par défaut).

7.1.1 Les règles de snort

Les paquets capturés par une sonde `snort` sont analysés en fonction de règles qui décrivent des attaques ou des vulnérabilités connues. Les règles dont l'utilisation est configurable dans `snort.conf` sont regroupées par nature dans des fichiers comme dans les exemples suivants : :

- `bad-traffic.rules` : des signatures figurant du trafic qui ne devrait jamais apparaître sur n'importe quel réseau ;
- `web-php.rules` : des signatures de scripts `php` réputés vulnérables ;
- `scan.rules` : des signatures figurant les scanners (`nmap`, ...) ;
- ...

Les règles de `snort` sont écrites dans un langage qui permet de décrire un paquet à analyser. Une règle est divisée en deux parties : la première permet de sélectionner les paquets ciblés, la seconde indique le sous-ensemble du paquet à inspecter en détail ainsi que le message d'alerte à générer si l'analyse est positive. L'exemple suivant est extrait de `x11.rules` qui ici génère une alerte si un usage de l'authentification `MIT-MAGIC-COOKIE-1`[6] est détecté :

```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET 6000
2 (msg:"X11 MIT Magic Cookie detected"; flow:established;
3 content:"MIT-MAGIC-COOKIE-1";
4 reference:arachnids,396; classtype:attempted-user; sid:1225; rev:4;)
```

La ligne 1 correspond à l'entête de la règle, elle sélectionne les paquets à analyser de la manière suivante :

[6] Ce procédé permet d'autoriser des clients X Window distants à utiliser un serveur X en partageant un clef. La clef est véhiculée en clair ...

- `alert` : l'action à déclancher si l'analyse est positive. Ici `alert` indique de générer une alerte dans le fichier `alert`. Par défaut, le paquet sera aussi journalisé (démarrer snort avec l'option -N pour supprimer la journalisation des paquets engendrant des alertes) ;
- `tcp` : sélection du protocole ;
- `$EXTERNAL_NET` : adresse IP source (cf. `snort.conf`) ;
- `any` : port source ;
- `$HOME_NET` : adresse IP destination (cf. `snort.conf`) ;
- `6000` : port destination ;

Les lignes 2,3 et 4 sont des options de la règle :

- `msg` : le message qui sera indiqué dans l'alerte ;
- `content` : le texte qui est recherché dans le contenu du paquet ;
- `flow` : le sens du traffic TCP ;
- `reference` : des références de la vulnérabilité sur des sites (bugtraa, cve, nessus, arachnids, ...) dédiés aux attaques ;

Une documentation sur l'écriture des règles de `snort` est disponible dans le Snort Users Manual[7]

7.1.2 Comment placer la sonde snort

cf. section 5.2.2.

7.1.3 Utiliser snort

Le démarrage de snort comme NIDS peut s'effectuer avec la ligne de commande suivante qui indique un démarrage comme *daemon* (NIDS), avec un fichier de configuration situé dans `/usr/local/etc/snort` :

```
snort -d -c /usr/local/etc/snort/snort.conf
```

Pour utiliser snort dans de bonnes conditions, il convient de décider quelles sont les règles qui doivent être utilisées. La configuration des règles s'effectue dans le fichier `snort.conf`, il faut commenter les règles inutiles. Par exemple, un site qui n'exploite pas de serveurs IIS a intérêt à ne pas générer d'alertes s'il reçoit des attaques sur ce type de service.

[7] http://www.snort.org/docs/snort_manual/2.6.1/snort_manual.pdf

Il convient également de configurer une méthode de propagation des alertes adaptée aux administrateurs des réseaux à surveiller. Le fichier des alertes de snort(`/var/log/snort/alert`) n'est pas de lecture aisée (cf. figure 7.1), un *mail* relatif à chaque alerte peut très rapidement s'avérer inexploitable : une interface de consultation exploitant le fichier des alertes est nécessaire.

```
1  [**] [1:2189:4] BAD-TRAFFIC IP Proto 103 PIM [**]
2  [Classification: Detection of a non-standard protocol or event] [Priority: 2]
3  01/18-15:20:08.834049 156.60.18.254 -> 224.0.0.13
4  PIM TTL:1 TOS:0xC0 ID:55939 IpLen:20 DgmLen:38
5  [Xref => http://cgi.nessus.org/plugins/dump.php3?id=11791]
6  [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0567]
7  [Xref => http://www.securityfocus.com/bid/8211]
8
9  [**] [1:1226:4] X11 xopen [**]
10 [Classification: Unknown Traffic] [Priority: 3]
11 01/19-13:17:24.608929 156.60.18.19:45985 -> 156.60.18.22:6000
12 TCP TTL:63 TOS:0x0 ID:35693 IpLen:20 DgmLen:64 DF
13 ***AP*** Seq: 0x3730914D  Ack: 0x5BCA83A1  Win: 0x5B4  TcpLen: 32
14 TCP Options (3) => NOP NOP TS: 1855134245 93899245
15 [Xref => http://www.whitehats.com/info/IDS395]
```

FIG. 7.1 – Le fichier des alertes de snort

7.1.3.1 Gérer les alertes de snort avec BASE

BASE[8] (*Basic Analysis and Security Engine*) est une interface de consultation des alertes de `snort`. BASE effectue un tri des alertes et propose de sélectionner les alertes du jour, les alertes uniques, ... Les figures 7.2 et 7.3 montrent respectivement la page d'accueil d'un service BASE et un écran de visualisation des alertes uniques.

[8] http://base.secureideas.net/

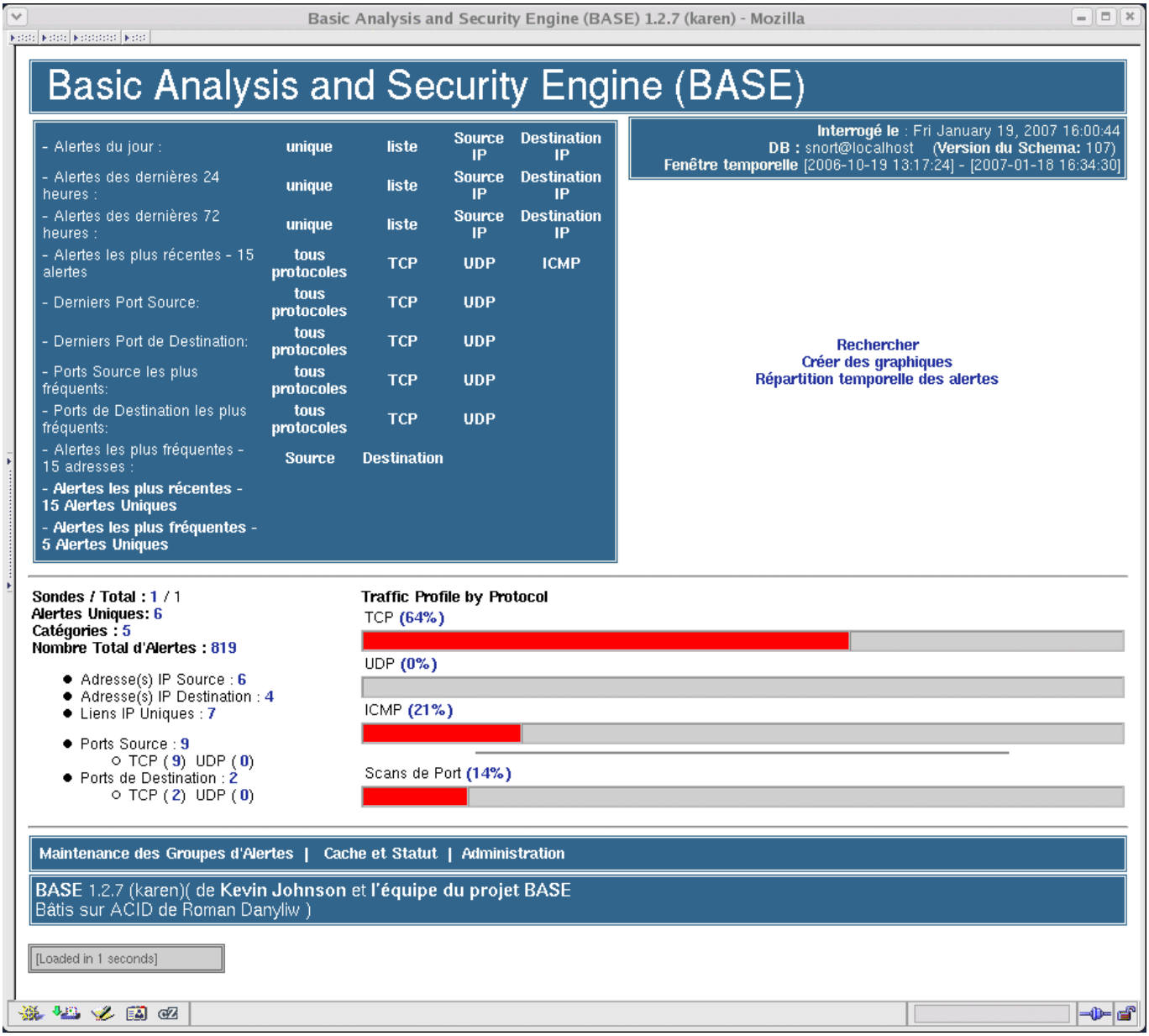


FIG. 7.2 – La page d'accueil d'un service BASE

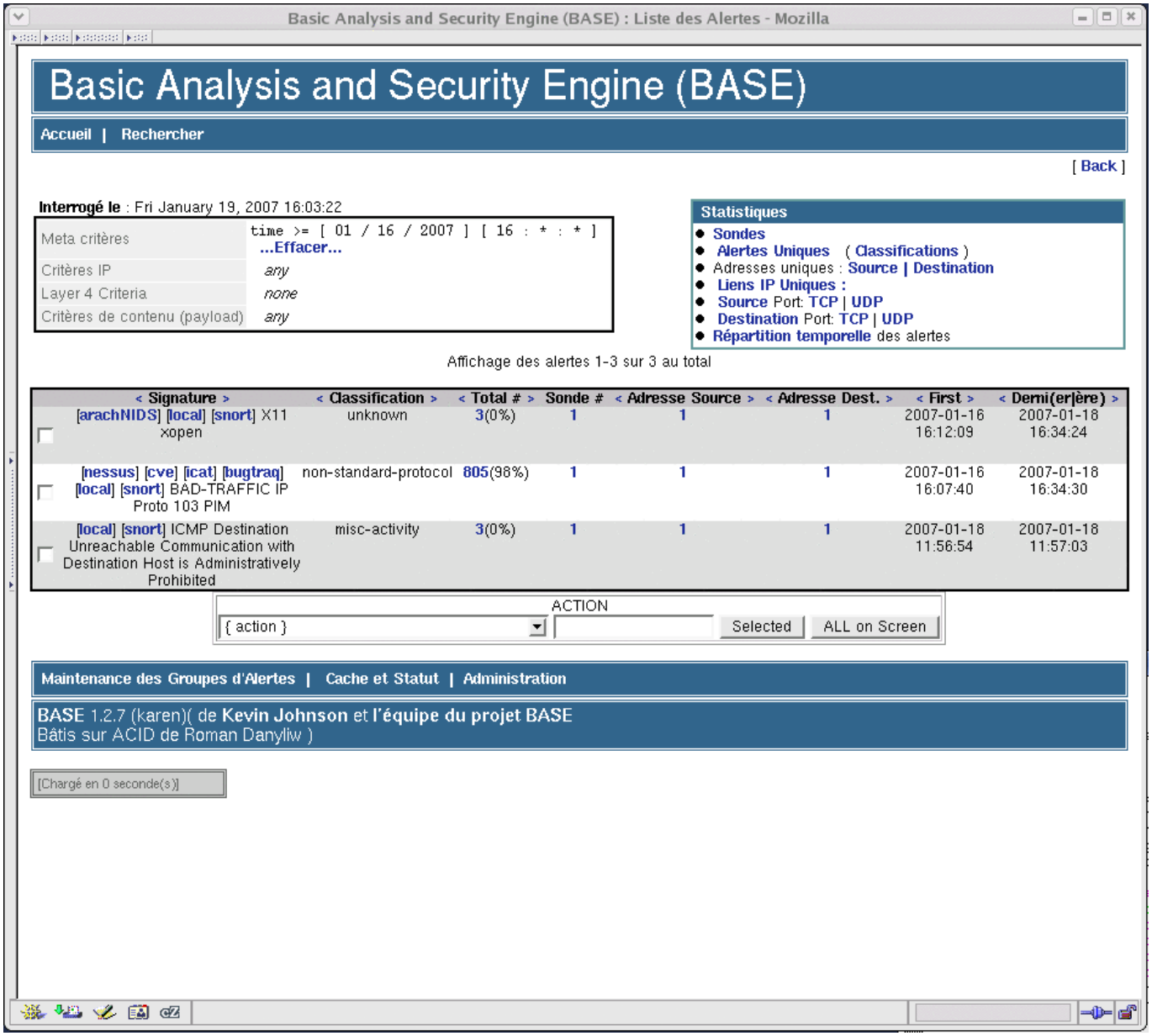


FIG. 7.3 – Les alertes uniques

7.1.3.2 snort et les faux positifs

Le problème principal posé par l'exploitation d'une sonde `snort` est l'analyse des alertes positionnées. Quelles sont les alertes justifiées ? Comment éliminer les alertes intempestives (les faux positifs) ?

L'utilisation d'une interface de consultation des alertes telle que BASE procure

un premier élément de confort : les alertes sont triées et consultables indépendamment de leurs occurences. Il reste tout de même à vérifier l'importance des faits signalés, cette opération n'est pas automatisable, elle doit être effectuée avec précision.

L'exemple suivant, issu du fichier contenant des règles ciblant les attaques sur des scripts CGI, illustre la part d'investigation laissée aux administrateurs : une alerte est remontée, elle indique qu'un des serveurs Web surveillés a reçu une requête d'accès à `sendmessage.cgi`. BASE précise la référence de la vulnérabilité (`cve-2001-1100`) : *sendmessage.cgi in W3Mail 1.0.2, and possibly other CGI programs, allows remote attackers to execute arbitrary commands via shell metacharacters in any field of the 'Compose Message' page.*. Si `W3Mail 1.0.2` est en service alors il s'agit peut être d'une attaque, dans le cas contraire l'alerte est un faux positif. Plus généralement, on notera qu'un appel à un script nommé `sendmessage.cgi` engendrera une alerte qu'il fasse partie ou pas de `W3Mail 1.0.2`.

À l'issue d'une alerte, lorsque les investigations sont terminées, l'administrateur peut s'il le juge nécessaire, commenter les règles qui génèrent des alertes non justifiées.

7.1.3.3 Conclusion

à suivre ...

Chapitre 8

Détection post intrusion

Lorsqu'une intrusion réussi, le pirate obtient des droits privélégiés sur la machine cible (super utilisateur). La suite des opérations dépend alors du but recherché, parmi les actions les plus souvent opérées, on peut citer les suivantes :

- désarmer les anti-virus et autres outils de sécurité qui peuvent être installés ;
- installer des certificats dans la liste des certificats de confiance des navigateurs ;
- utiliser la machine corrompue comme émetteur de spams ou comme scanneur `nmap` ;
- ...

Les *rootkits* sont des outils prêts à l'emploi qui permettent d'automatiser les opérations post intrusion. Le but est toujours le même : rendre furtives et persistantes les opérations malveillantes opérées par les pirates. La suite de ce chapitre présente par l'exemple les techniques utilisées par les *rootkits* pour Linux ainsi que quelques détecteurs de présence de *rootkits*.

8.1 Les *rootkits*

Les *rootkits* font partie de l'Internet souterrain, ils ne s'affichent évidemment pas toujours. Une liste non exhaustive et non à jour peut toutefois être consultée par http://packetstormsecurity.org/UNIX/penetration/rootkits/index.html. Le site http://www.eviltime.com/ diffuse également les dernières versions de quelques rootkits, on peut notamment y trouver `adore-ng` et `suckit`.

Plusieurs techniques sont utilisées par les *rootkits* pour dissimuler des processus ou des répertoires, parmi celles-ci citons les suivantes :

– le remplacement des commandes usuelles utilisées par les administrateurs (`ls`, `ps`, `netstat`, `top`, ...) par des versions masquant la réalité ;

– la modification de certaines bibliothèques partagées utilisées dynamiquement par les commandes de base ;

– la modification ou l'ajout de modules au noyau du système. Ce procédé est le plus redoutable car il donne accès à l'espace mémoire géré par le noyau (*kerneland*).

8.1.1 adore-ng

`adore-ng` est un *rootkit* pour Linux qui permet d'ajouter des fonctionnalités cachées au noyau. `adore-ng` peut être utilisé avec des noyaux 2.4 ou 2.6 et procure les fonctionnalités suivantes :

– masquage de fichiers et répertoires : les commandes telles `ls` ne montrent pas les fichiers et répertoires désignés par le *hacker* ;

– masquage de processus : les commandes telles `top`, `ps`, ... ne montrent pas les processus désignés par le *hacker* ;

– mise en place de portes dérobées ;

– nettoyage des traces.

La figure 8.1 montre `adore-ng` en action via la commande *ava* qui est ici utilisée pour cacher un processus (la figure 8.2 montre la détection du processus caché par `zeppoo`).

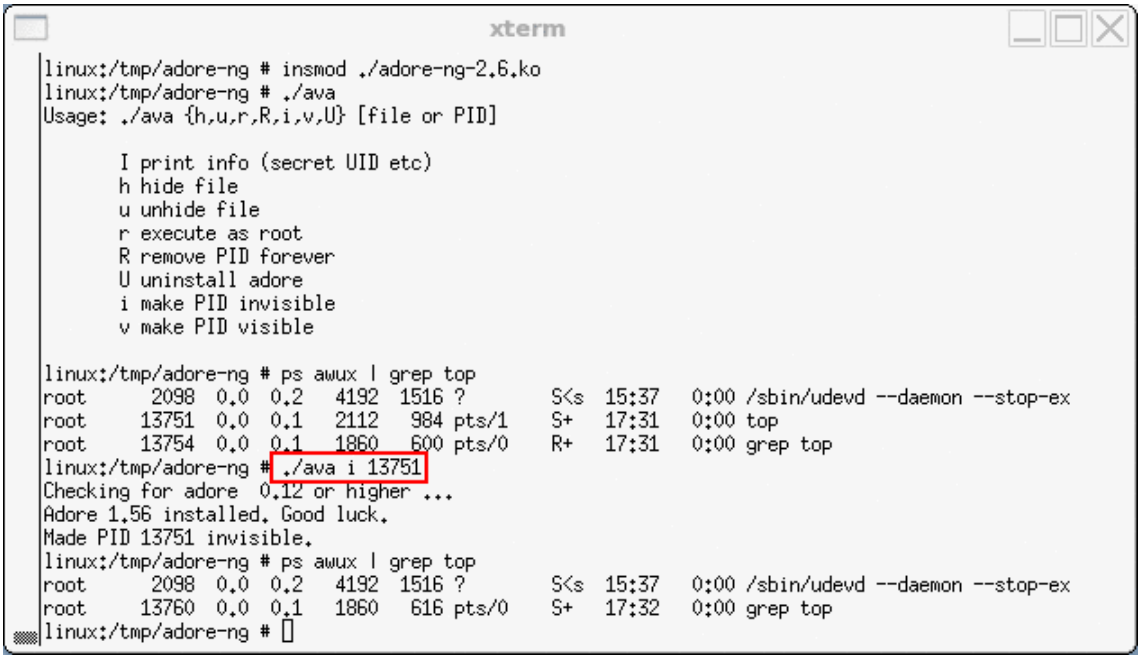


FIG. 8.1 – `adore-ng` cache un processus

Une fois installé (cf. `insmod` de la figure 8.1), `adore-ng` permet également de cacher des fichiers ou répertoires (`ava h`) ou de lancer des commandes en passant *root* (`ava r`).

8.2 Les anti *rootkits*

8.2.1 Zeppoo

`zeppoo`[1] est un outil qui permet de détecter des *rootkits* évoluant en mode utilisateur (*userland*) ou en mode noyau (*kerneland*). Pour ce faire il s'appuie principalement sur une base de signatures du système sain (mode utilisateur) et sur un parcours de `/dev/kmem` (la mémoire utilisée par le noyau) pour comparer les processus présents avec ceux observés dans le pseudo répertoire `/proc` (un processus caché apparaîtra dans le parcours de `/dev/kmem` mais ne sera pas présent dans `/proc`)..

[1] http://zeppoo.net

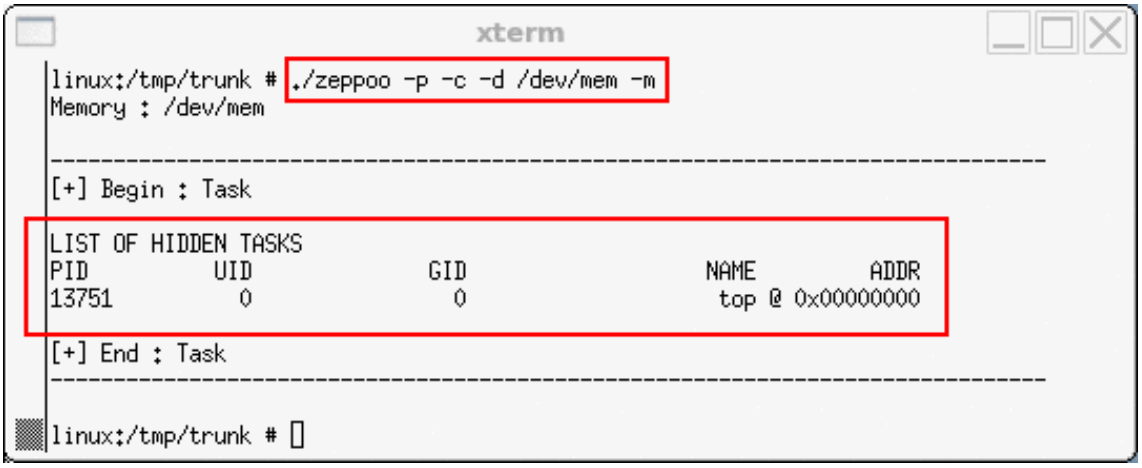


FIG. 8.2 – `zeppoo` détecte un processus caché par `adore-ng`.

Chapitre 9

Conclusion