HERVÉ SCHAUER CONSULTANTS
Cabinet de Consultants en Sécurité Informatique depuis 1989
Spécialisé sur Unix, Windows, TCP/IP et Internet

Ossir Groupe SUR
Mardi 13 Juin 2006

Détection de tunnels en périphérie du réseau

Guillaume Lehembre <Guillaume.Lehembre@hsc.fr>
Alain Thivillon <Alain.Thivillon@hsc.fr>

Agenda

- Problématique des tunnels
- Différents types de tunnels (HTTP, HTTPS, ICMP, DNS)
- Détection de tunnels basée sur les journaux applicatifs
- Détection de tunnels en temps réel
- Moltunnel

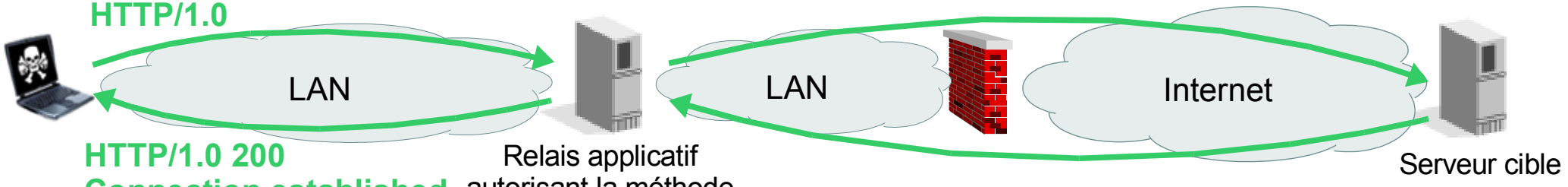
Problématique des tunnels

- But des tunnels : contourner la politique de sécurité
  - Accès à des protocoles non prévus
  - Accès à des ressources non autorisées (réseaux, applications, ...)
  - Cacher ses actions (sites Web, mails, chat, ...)
- Cadre :
  - Réseau filtré, mécanismes de relayage
  - Usage pratique des techniques de tunnels (non intrusif)

  ↳ non basé sur de « réels » canaux cachés (efficacité)
- Utilisation détournée de protocoles standards : HTTP, HTTPS, ICMP, DNS

Tunnels HTTP / HTTPS - 1

- Les plus « simples » et les plus utilisés
- Méthodes de contournement :
  - 1. Relayage arbitraire TCP dans CONNECT



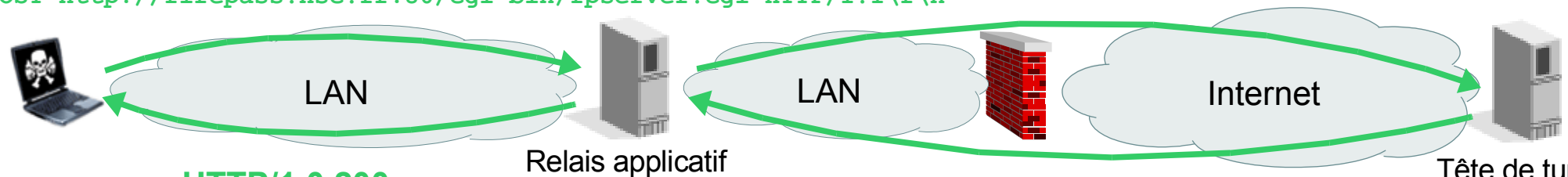

```
$ socat tcp4-listen:2222 proxy:carbone.hsc.fr:unserveurssh.com:22 proxyport=8080
$ telnet carbone.hsc.fr 8080
Trying 192.70.106.49...
Connected to carbone.hsc.fr.
Escape character is '^]'.
CONNECT unserveurssh.com:443 HTTP/1.0

HTTP/1.0 200 Connection established

SSH-2.0-OpenSSH_4.2
$ ssh -p 2222 user@localhost
debug1: Connecting to localhost [127.0.0.1] port 2222.
debug1: Connection established.
debug1: Remote protocol version 2.0, remote software version OpenSSH_4.2
```

  - 2. Utilisation de SSL

Tunnels HTTP / HTTPS - 2

- 3. Requêtes HTTP classiques de type GET et/ou POST sur CGI ou directement dans les URL

```
POST http://tun.hsc.fr:80/index.html?crap=1143719371 HTTP/1.1\r\n
GET http://tun.hsc.fr:80/index.html?crap=1143719371 HTTP/1.1\r\n
---
POST http://loophole.hsc.fr:80/display/mode/forum/minimize.asp HTTP/1.0\r\n
---
POST http://firepass.hsc.fr:80/cgi-bin/fpserver.cgi HTTP/1.1\r\n
```

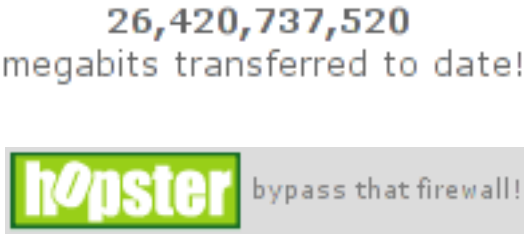


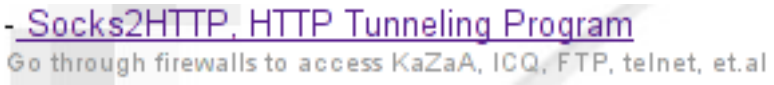
HTTP/1.0 200
Connection established
SSH-2.0-OpenSSH_4.2

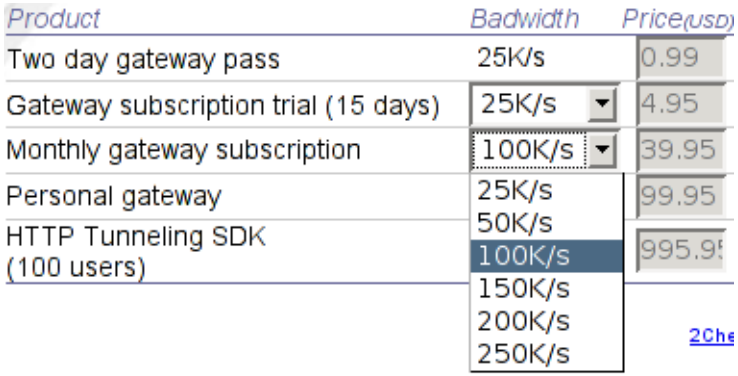
- Utilisation fréquente de mécanismes de polling
- Outils tunnels HTTP : GNU HTTPTunnel, LoopHole, Firepass
- Outils tunnels HTTPS : OpenVPN, Stunnel, SSLTunnel (HSC), SSL-Explorer, VPN SSL commerciaux, etc.

Tunnels HTTP commerciaux ...

- Affichent clairement leurs intentions : IM, P2P, IRC, etc
- Fournisseurs de tunnels mettant à disposition leur propre tête de tunnel HTTP :
  - HTTP Tunnel (http://www.http-tunnel.com)
  - Hopster (http://www.hopster.com)
  - Totalrc (http://www.totalrc.net)
- Versions gratuites limitées :
  - Bande passante (~ 2 ko/s !)
  - Temps





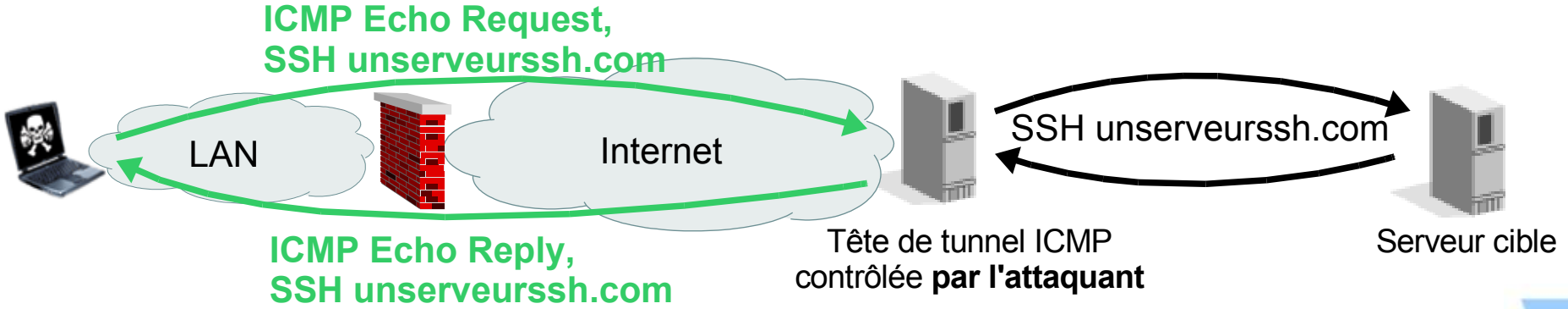


Tunnels ICMP

- Données arbitraires inclues dans la charge utile des requêtes *ICMP Echo Request* (type 8) et *ICMP Echo Reply* (type 0)
- Gestion de la connexion à l'aide de messages d'états

magic	ip	port	state	ack	length	seq	rsv	data ...

- Contenu habituel de la charge utile : informations temporelles
- Outils : Loki, PingTunnel, Skeeve, etc.
- Détection : quantité anormale de requêtes ICMP, analyse de la charge utile des requêtes ...

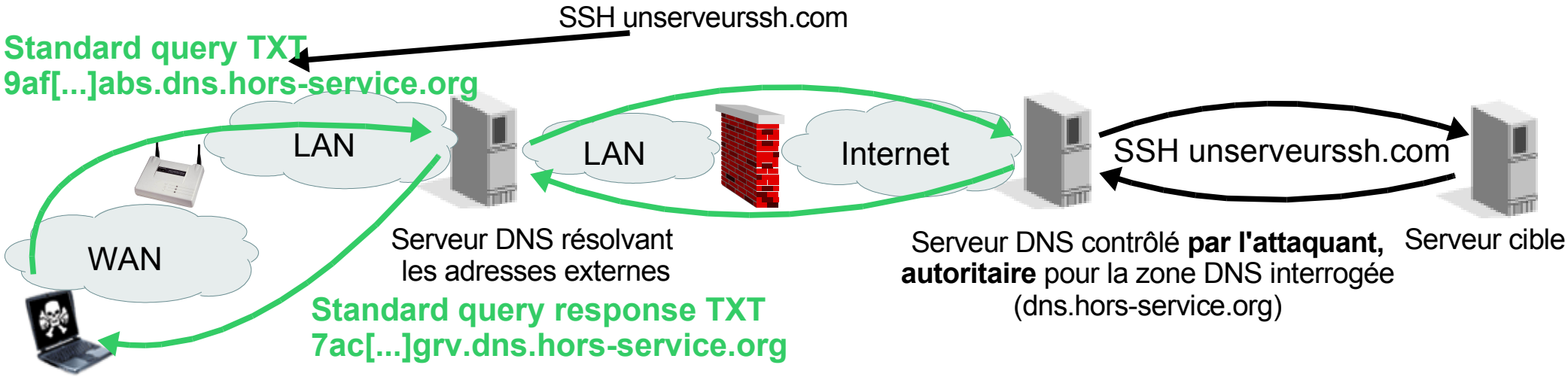


Tunnels DNS - 1

- DNS internes permettant trop souvent de résoudre des adresses externes
  - Considéré à tord comme un protocole « inoffensif »
  - Données arbitraires pouvant être inclues dans le nom DNS recherché : les enregistrements A, TXT, KEY sont transmis de bout en bout dans la chaîne de serveurs DNS
- Usages courants : Enregistrements TXT utilisés dans SPF, enregistrements KEY dans DNSSEC
- Limitation :
  - Certains serveurs DNS décompressent les paquets à la volée
  - Question : Longueur totale < 253 caractères, 63 caractères par « . », base32 conseillé
  - Réponses : paquet DNS < 512 bits au niveau IP pour éviter la fragmentation, base64

Tunnels DNS - 2

- Parades :
  - DNS interne, seul le relais applicatif doit pouvoir interroger le DNS externe
  - Filtrage des requêtes DNS et de leur nombre
  - Utilisation de plusieurs DNS avec ajustement TTL (problématique du cache DNS Windows dans le cas des HotSpot publics ...)
- Outils : NSTX, Oxyman; DNS2TCP (HSC), etc.



Détection de tunnels - Journaux

- Détection « à postériori »
- Examen des journaux :
  - User-Agent inconnu
  - Volume de données échangées
  - Nombre de « hits »
  - Ecart type entre les requêtes (journaux Bind)
  - etc.
- Limites :
  - Pas d'analyse protocolaire
  - Durée des connexions TCP, début de connexion TCP
  - Information parcelaires (exemple SQUID: pas volume uploadé)

↳ Solution : Détection de tunnels en temps réel

Détection de tunnels - 1

- Examen du trafic HTTP
  - Méthode HTTP inconnue
  - En-têtes HTTP incorrects ou incomplets (Absence du champ Host, en-têtes très longs, Content-Length incorrect, ...)
  - User-Agent inconnu
  - Réponse du serveur incorrecte (version HTTP)
- Examen HTTPS
  - Méthode CONNECT utilisée avec une destination numérique (les certificats SSL sont basés sur des FQDN) ou sur un port inhabituel.
  - Absence de champ Host sur la méthode CONNECT
  - Trafic non-SSL sur le port 443
  - Certificats suspects

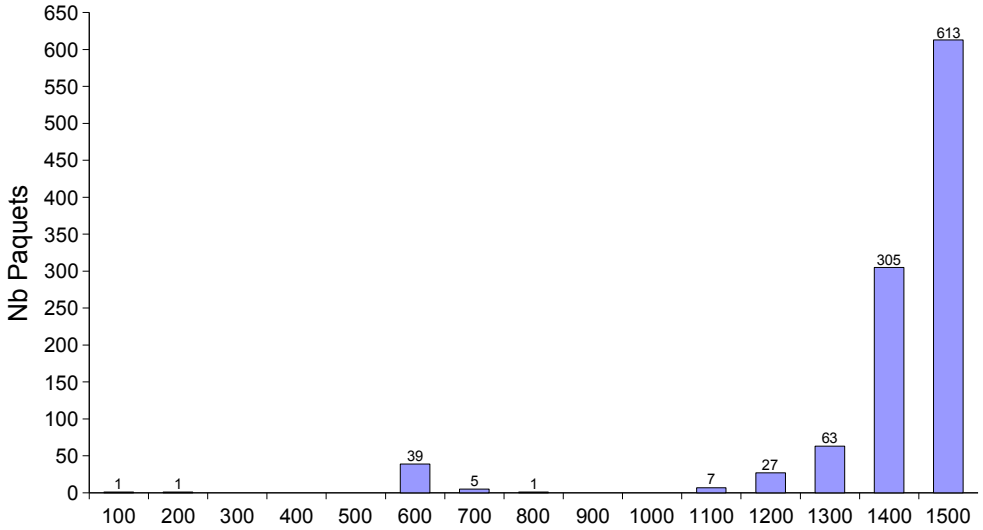
Détection de tunnels - 2

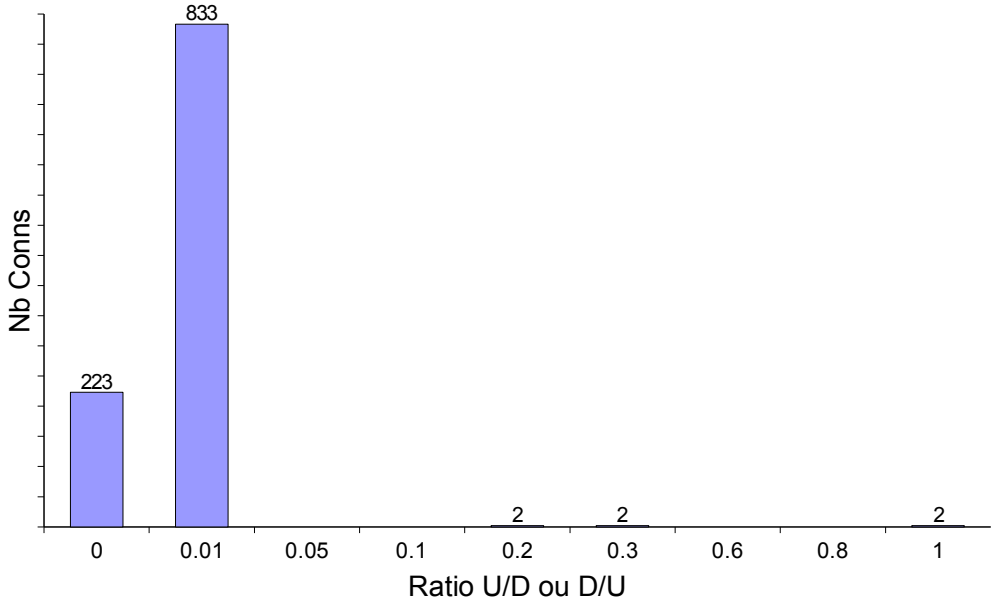
- Détection « statistiques » basiques
  - Durée de la connexion : les connexions HTTP/HTTPS durent rarement plus de 10 minutes, à corréler avec la bande passante disponible et le trafic.
  - Les protocoles comme HTTP/HTTPS/POP/SMTP sont très asymétriques:
    - Émission ou réception de gros volumes de données, mais rarement les deux
    - ⇨ Ratio UPLOAD/DOWNLOAD ou DOWNLOAD/UPLOAD < 0.3
  - Taille moyenne des paquets : une vraie connexion va « bourrer » les segments TCP : des paquets de taille faible sont signe d'une connexion probablement interactive

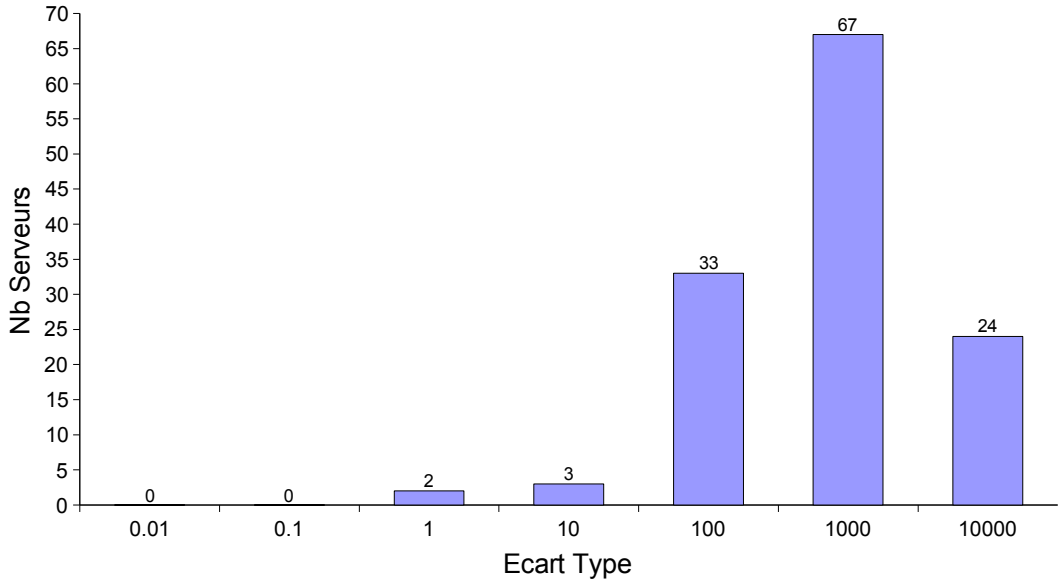
Détection de tunnels - 3

- Forme temporelle des connexions suivant chaque protocole
  - HTTP/HTTPS: Requête/Réponse ... Requête/Réponse
  - SMTP : Init/Echanges/Données Unidirectionnelles
  - POP : <RETR >DATA <RETR >DATA ...
- Examen global des connexions vers chaque serveur
  - Nombre, Durée, Volume dans chaque sens : création d'une liste de top-tens accédés par les utilisateurs, et recherche des nouveautés
  - Écart-type entre chaque requête afin de détecter les requêtes très répétitives
  - ⇨Détection de MSN dans HTTP, ou de tunnels comme LoopHole

Exemples







Moltunnel - 1

- Outil sous Unix destiné à détecter « des » tunnels
  - Pas de prétention à l'exhaustivité, mais tentative de généricité (par rapport à une approche purement IDS par *patterns*)
  - Contournements probablement faciles
- Détection « sur le fil » par écoute réseau
  - Nécessité de réassembler les sessions TCP : utilisation de libnids
  - A placer par exemple entre les utilisateurs et le relais HTTP
- Examen des connexions HTTP, des connexions SSL, des connexions Proxy et détection des anomalies en temps réel
- Écriture d'un fichier plat permettant des analyses statistiques plus fines hors ligne et pour stockage

Moltunnel - 2 : Exemples

Détection d'un SSH sur le port 443

```
> 12:13:58 192.70.106.104:53616->80.65.234.114:443 : Invalid SSL
trafic (Server Banner)
> 12:13:58 192.70.106.104:53616->80.65.234.114:443 : Invalid SSL
client Hello
```

Détection de User-Agents non conformes

```
> 12:18:20 192.70.106.104:53641->192.70.106.49:8080 : Bad User-
Agent (Maverick-HttpClient/1.0) for: CONNECT www.rominet.net:8443
```

Détection de trafic interactif

```
> 12:20:26 192.70.106.104:53543->82.123.113.221:22 : > Long
Connection 1917018 ms
> 12:20:26 192.70.106.104:53543->82.123.113.221:22 : > Small
Packets 140.0 / 57.5
```

Détection de SKYPE

```
> 12:20:26 192.16:45:24 192.70.106.104:56478->192.70.106.49:8080
: CONNECT all numeric : 212.72.49.141:443
```

Moltunnel - 3 : Exemples

Détection de VPN SSL (SSLExplorer)

```
> 13:04:18 192.70.106.104:53641->192.70.106.49:8080 : > Wrong
Ratio 0.7753 (21189 16428)
> 13:04:18 192.70.106.104:53641->192.70.106.49:8080 : > Long
Connection 2758326 ms
```

Détection de MSN, LoopHole et Firepass par examen de l'écart-type:

```
192.70.106.101-207.46.1.4                                19407   978
19.84    2.87
192.70.106.104-192.70.106.166:8080                         675   854
 0.79    0.32
192.70.106.103-smash.dyndns.org:80                        1260  1119
 1.13    0.85
```

Moltunnel - Problèmes

- LIBNIDS très peu fiable ☹
  - Détection des fins de session Windows (FIN-ACK simultané)
  - Problème des fenêtres > 64Ko
  - Attention : perte d'un paquet ⇨ Connexion plus analysable
- Ecrire un parseur HTTP fiable et exhaustif est une tâche difficile
  - HTTP/1.1 + Transferts chunked, mod_gzip, ...
- ⇨ Repartir de Snort ?
- Certains tunnels restent difficiles à détecter
  - Exemple : GnuHTTPTunnel : un POST en écriture, un GET en lecture, fermeture après quelques minutes ou 64Ko ...
- Quelques faux positifs, nombre incroyable de User-Agents visibles dans un trafic standard (XP, Acrobat, ...)

Conclusions

- Idées en l'air
  - Réseaux de neurones
  - Transformées de Fourier
  - A coupler avec un IDS analysant les protocoles
- Problèmes : volumes de données à traiter, mémoire, analyse pendant la capture, ...
- Moltunnel Disponible au second semestre sur http://www.hsc.fr/
- Solutions commerciales
  - Les vendeurs de firewall sont également vendeurs de VPN SSL ...
  - Coupure de SSL : bonne solution ? (confidentialité, certificats clients, ...)
- Le « Web 2.0 » (haha) change les métriques