



HERVÉ SCHAUER CONSULTANTS
Network Security Agency since 1989
Specialized in Unix, Windows, TCP/IP and Internet

SSL VPN connection multiplexing techniques



Franck Davy <fd@hsc.fr>



Agenda

IPsec protocol: Standards vs Proprietary and Alternative solutions

SSL VPN Port Forwarding techniques

SSL VPN Tunnel Management protocols:

SSH vs SSL

Multiplexing Data Streams with SOCKS

Multiplexing with HTTP Tunneling

Tunneling filtering

Conclusion

References

Checkpoint Solution

- > Checkpoint FW-1/VPN-1 Gateway
- > VPN Client software
 - > SecuRemote, SecureClient (personal firewall integration)

Authorized protocols

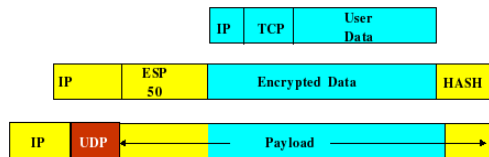
- > IKE Protocol: 500/udp & 500/tcp
- > FW-1 Topology: 264/tcp
- > **Without NAT :**
 - > (IP Protocol) ESP: 50
 - > (IP Protocol) FWZ: 94
 - > RDP service (FWZ negotiation): 259/udp
- > **With NAT :**
 - > IPSec ESP encapsulated in UDP datagrams: 2746/udp

Cisco Solution

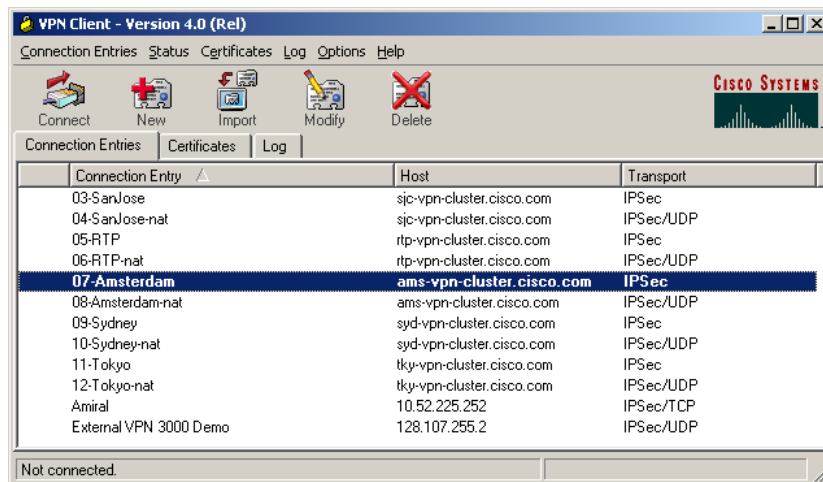
- > Concentrator 3000/5000 Gateway, PIX Firewall, IOS VPN
- > VPN Client software

Authorized protocols

- > IKE protocol: 500/udp
- > **With NAT:**
 - > (IP Protocol) ESP : 50
- > **Without NAT:**
 - > **NAT-T (IETF Draft)** – IPSec ESP encapsulated in UDP datagrams: 4500/udp
 - > Cisco specific:
 - > IPSec ESP encapsulated in UDP flow: 10000/udp
 - > IPSec ESP encapsulated in TCP connection: 10000/tcp



HSC Proprietary IPsec Solutions (3)



The screenshot shows the 'VPN Client - Version 4.0 (Rel)' window. It has a menu bar with 'Connection Entries', 'Status', 'Certificates', 'Log', 'Options', and 'Help'. Below the menu bar is a toolbar with icons for 'Connect', 'New', 'Import', 'Modify', and 'Delete'. The main area contains a table with the following data:

Connection Entry	Host	Transport
03-SanJose	sjc-vpn-cluster.cisco.com	IPSec
04-SanJose-nat	sjc-vpn-cluster.cisco.com	IPSec/UDP
05-RTP	rtp-vpn-cluster.cisco.com	IPSec
06-RTP-nat	rtp-vpn-cluster.cisco.com	IPSec/UDP
07-Amsterdam	ams-vpn-cluster.cisco.com	IPSec
08-Amsterdam-nat	ams-vpn-cluster.cisco.com	IPSec/UDP
09-Sydney	syd-vpn-cluster.cisco.com	IPSec
10-Sydney-nat	syd-vpn-cluster.cisco.com	IPSec/UDP
11-Tokyo	tky-vpn-cluster.cisco.com	IPSec
12-Tokyo-nat	tky-vpn-cluster.cisco.com	IPSec/UDP
Amiral	10.52.225.252	IPSec/TCP
External VPN 3000 Demo	128.107.255.2	IPSec/UDP

At the bottom of the window, it says 'Not connected.' The Cisco Systems logo is visible in the top right corner of the window.

HSC Proprietary IPsec Solutions (4)

IPsec standard protocols are IKE (500/UDP), and AH (51)/ESP (50) IP protocols – usually filtered by firewalls or not compatible with private address space (NAT issues): proprietary IPsec and alternative solutions (like VPN based on SSL/TLS protocols) have emerged.

SSL is a Netscape specification, TLS an IETF standard (RFC2246). They provide Transport Layer security, *i.e.* per application security, and not a Network Layer Security: proprietary extensions are still required to provide IPsec equivalent access to resources – extensions dealing with data streams multiplexing over a single SSL/TLS session, for instance.

SSL VPN application tunneling techniques :

Port Forwarding

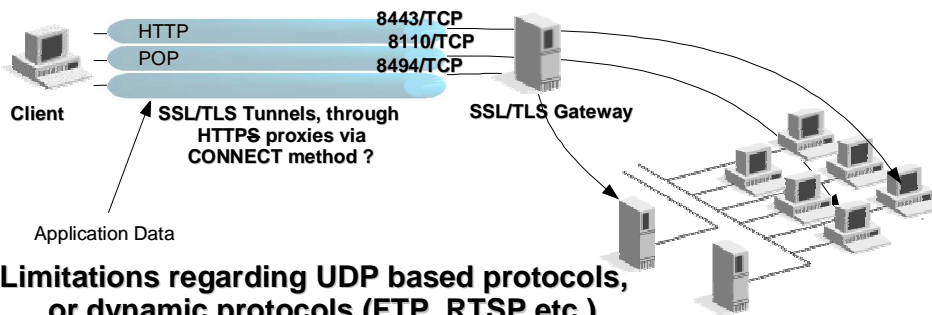
Data streams multiplexing

Port Forwarding (1)

Port Forwarding

Encrypted/authenticated SSLv3/TLSv1 tunnel between the client and a Gateway

- > Lightweight or native client
- > Each data flow is redirected (wrapped) inside a dedicated tunnel



Limitations regarding UDP based protocols, or dynamic protocols (FTP, RTSP etc.), and ports allocation on the gateway

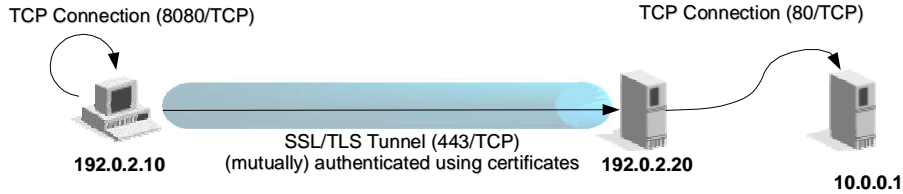
HSC Port Forwarding (2)

Example: starting stunnel in daemon + wrapper modes

```

Server side
# stunnel -d 443 -r 10.0.0.1:80 -v 2 -a /etc/ssl/trusted -p stunnel.pem
Enter PEM pass phrase:
$ netstat -antp | grep stunnel
tcp    0    0 0.0.0.0:443          0.0.0.0:*           LISTEN  2401/stunnel

Client side
$ stunnel -c -r 192.0.2.20:443 -d 8080 -a /etc/ssl/trusted
    
```

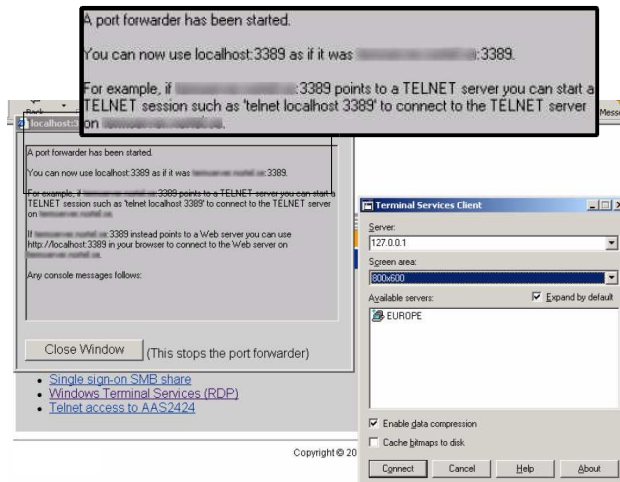


Limitation:
One tunnel (port) per application
No TCP connection multiplexing



HSC Port Forwarding (3)

Port forwarding



HSC Tunnel management protocols (1): SSH vs SSL protocols

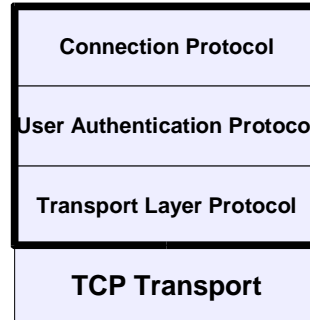
« SSH Connection Protocol » draft

> <http://www.openssh.org/txt/draft-ietf-secsh-connect-15.txt>

« SSH is a protocol for secure remote login and other secure network services over an insecure network. This document describes the SSH Connection Protocol. It provides interactive login sessions, remote execution of commands, **forwarded TCP/IP connections**, and forwarded X11 connections.

All of these channels are multiplexed into a single encrypted tunnel. The SSH Connection Protocol has been designed to run on top of the SSH transport layer and user authentication protocols. »

SSH protocols stacks



SSH protocol provides tunnel management through « Connection Protocol » layer

- 11 -

© Herve Schauer Consultants 2005



HSC Tunnel management protocols (2): SSH vs SSL protocols

SSH tunnel management :

On the wire: One TCP connection (22/TCP)

→ **Applicative flows – relying on TCP transport protocol – are wrapped and multiplexed into a single SSH session (and a single TCP connection)**

```
> ssh -N -f -2 -L 10143:192.168.0.4:143 \  
-L 10389:192.168.0.4:389 \  
-L 10080:localhost:80 \  
192.168.0.1  
  
> -#  
The following connections are open:  
#3 client-session (t4 r0 i0/0 o0/0 fd 7/8)  
#4 direct-tcpip: listening port 10389 for 192.168.0.4 port 389,  
connect from 127.0.0.1 port 33462 (t4 r1 i0/0 o0/0 fd 10/10)  
#5 direct-tcpip: listening port 10143 for 192.168.0.4 port 143,  
connect from 127.0.0.1 port 33463 (t4 r2 i0/0 o0/0 fd 11/11)  
#6 direct-tcpip: listening port 10080 for localhost port 80,  
connect from 127.0.0.1 port 33465 (t4 r3 i0/0 o0/0 fd 12/12)
```

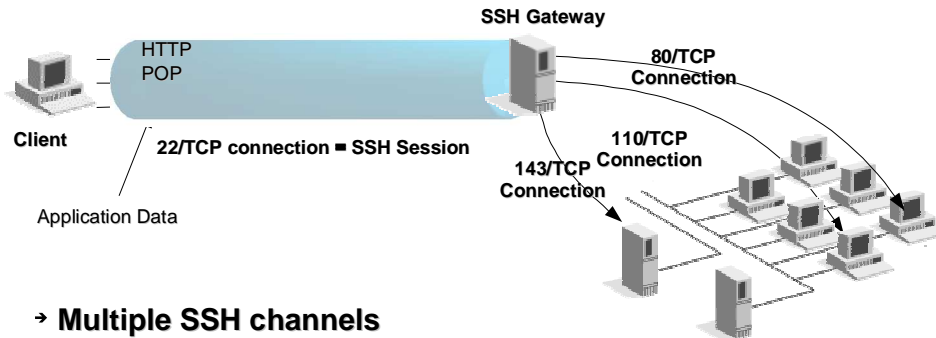
- 12 -

© Herve Schauer Consultants 2005



HSC Tunnel management protocols (3): SSH vs SSL protocols

SSH tunnel management:

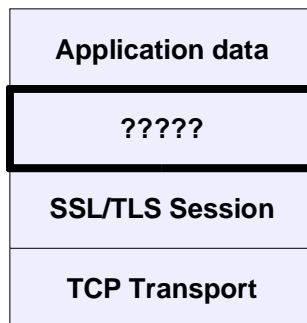


- Multiple SSH channels
- One SSH Session
- One TCP Connection (22/TCP)

SSL/TLS Protocols **do not provide** a management mechanism for multiple tunnels: they need to rely on a third-part protocol to handle channel management

HSC Tunnel management protocols (4): SSH vs SSL protocols

Multiplexing data streams protocols

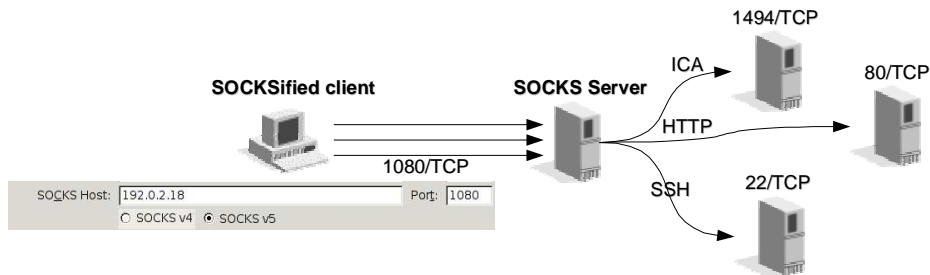


SOCKS v5 based MUX layer ?
 HTTP encapsulation layer ?
 PPP encapsulation layer ?
 Other proprietary solution ?

- Multiplexing performed, on client-side, by Java applets or Active X controls

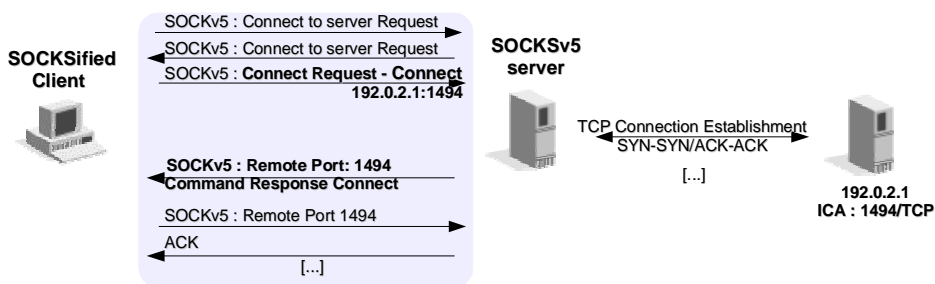
HSC Tunnel management protocols (5): « SOCKSification »

SOCKS (RFC1928):
Generic proxy, « shim layer » between application and transport layers



HSC Tunnel management protocols (6): « SOCKSification »

SOCKS Protocol example:



SOCKSv5 (RFC1928) protocol supports TCP and UDP based protocols and can (optionally) support SSL/TLS tunneling

→ Use of combined SOCKS + SSL provides UDP and TCP protocols encapsulation and multiplexing, through a SSL/TLS tunnel.

HSC Tunnel management protocols (7): « SOCKSification »

Application « SOCKSification »

- > SOCKS protocol **natively** supported in usual programs (namely browsers)
- > **Libraries** can intercept **connect()** system call and **SOCKSify** applications, without recompiling or modification, providing transparent network access through SOCKS

```
$ echo -n "HEAD / HTTP/1.0\n\n\n" | strace -econnect nc www.hsc.fr 80  
connect(3, {sa_family=AF_INET, sin_port=htons(53), sin_addr=inet_addr("192.168.0.1")}, 28) = 0  
connect(3, {sa_family=AF_INET, sin_port=htons(80), sin_addr=inet_addr("192.70.106.33")}, 16) = 0  
HTTP/1.0 200 OK [...]
```

```
$ echo -n "HEAD / HTTP/1.0\n\n\n" | strace -econnect tsocks nc www.hsc.fr 80  
connect(3, {sa_family=AF_INET, sin_port=htons(53), sin_addr=inet_addr("192.168.0.1")}, 28) = 0  
connect(3, {sa_family=AF_INET, sin_port=htons(1080), sin_addr=inet_addr("127.0.0.1")}, 16) = 0  
HTTP/1.0 200 OK [...]
```

Example: tsocks program transparently proxies applications (TCP establishment), requiring to load `/usr/lib/libtsocks.so` setting environment variable `LD_PRELOAD` :

```
$ tsocks -on ; tsocks -sh  
LD_PRELOAD="/usr/lib/libtsocks.so"
```

HSC Tunnel management protocols (8): HTTP as « transport layer »

An increasingly large number of applications are using HTTP protocol as a transport layer protocol, in order to bypass corporate TCP/IP outbound filtering through HTTP proxies, compliant with HTTP/1.X standards

Typically: Real Time Streaming Protocol (RTSP), used for media streaming

Several modes are available, based on FTP-like data and control connections

Initial version: hybrid mode, relying on UDP protocol for data streaming (RTP, 554/UDP), and TCP connection for control channel

Revised version: TCP-only mode, for both data and control channels – TCP connections are always established from the client

«HTTP tunneling» version: RTSP and RTP channels are tunneled over HTTP. HTTP transport is built from two separate HTTP GET and POST method requests initiated by the client.

HSC Tunnel management protocols (9): HTTP as « transport layer »

Taken from <http://developer.apple.com/>, about QuickTime :

«Using standard RTSP/RTP, a single TCP connection can be used to stream a QuickTime presentation to a user. Such a connection is not sufficient to reach users on private IP networks behind firewalls where HTTP proxy servers provide clients with indirect access to the Internet.

To reach such clients, QuickTime 4.1 supports the placement of RTSP and RTP data in HTTP requests and replies. As a result, viewers behind firewalls can access QuickTime presentations through HTTP proxy servers.»

P2P, IM software and trojans are using HTTP tunneling – so can do SSL VPN.

HSC Tunnel management protocols (10): HTTP as « transport layer »

HTTP GET and POST methods can carry an indefinite amount of data in their reply (from server) and message body (from client) respectively.

```
POST /SmpDsBhgRl HTTP/1.0
Accept: application/x-rtsp-tunnelled, */*
Content-type: application/x-pncommand
Content-length: 32767
```

HTTP POST Request

```
3fe06463-55b0-448d-be04-120bd3cb7a1f
1BUSU9OuyBydHNwOi8vcm[...]
```

HTTP POST Request body : Base64 encoded
RTSP Control channel

```
GET /SmpDsBhgRl3fe06463-55b0-448d-be04-120bd3cb7a1f HTTP/1.0
Accept: application/x-rtsp-tunnelled, */*
ClientID: WinNT_5.0_6.0.10.505_play32_RN9GPD_en-US_686_axembed
X-Actual-URL: rtsp://realserver.domain.tld/videos/20d05122002.rm
```

HTTP GET Request

```
HTTP/1.0 200 OK
Server: RMServer 1.0
Content-type: audio/x-pn-realaudio
Content-length: 2147483000
```

```
RTSP/1.0 200 OK
CSeq: 1
Date: Mon, 06 Dec 1999 08:13:01 GMT
Server: RealServer Version 6.1.3.946 (sunos-5.6-sparc)
Public: OPTIONS, DESCRIBE, ANNOUNCE, SETUP, GET_PARAMETER, TEARDOWN
StatsMask: 3
[...]
```

HTTP Reply body : Base64 encoded
RTSP/RTP Control/Data channel

HSC Tunnel management protocols (11): HTTP as « transport layer »

SSL VPN: Example of Telnet protocol

```

8 15 0.2118 (0.0017) S>C application_data
48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.
7 16 1.0181 (0.2172) S>C application_data
03 00 00 02 00 6c 00 0d 0a 58 58 58 58 58 58 58 .....1...XXXXXXXX
2e xx xx xx xx xx xx xx xx 2f 4f 70 65 6e 56 69 .XXXXXXXXXX/OpenVi
65 77 20 5b 48 50 2d 55 58 20 42 2e 31 30 2e 32 ew [HP-UX B.10.2
30 5d 0d 0a xx xx xx xx xx xx xx xx 20 20 20 0]..XXXXXXXXXX
0a 0d 0a ...
8 16 0.3637 (0.1518) C>S application_data
POST /id/param?channel=12345 HTTP/1.1
8 17 0.3638 (0.0000) C>S application_data
01 02 07 00 01 00 04 00 00 02 00 .....
8 18 0.3654 (0.0016) S>C application_data
48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.
7 17 1.0467 (0.0286) S>C application_data
03 00 00 03 00 07 00 6c 6f 67 69 6e 3a 20 .....login:

```

Decrypted with SSLDump (<http://www.rtfm.com/ssldump>) with RSA private key and RSA static key exchange protocol – i.e. no PFS.



HSC Tunnel management protocols (12): HTTP as « transport layer »

```

7 55 7.5653 (0.0100) S>C application_data
03 00 00 29 00 09 00 50 61 73 73 77 6f 72 64 3a .....Password:
8 247 8.2791 (1.3847) C>S application_data
POST /id/param?channel=12345 HTTP/1.1
8 248 8.2792 (0.0001) C>S application_data
01 02 08 00 01 00 06 00 00 01 00 48 .....H
8 249 8.2805 (0.0013) S>C application_data
48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.
8 250 8.3970 (0.1165) C>S application_data
POST /id/param?channel=12345 HTTP/1.1
8 251 8.3972 (0.0001) C>S application_data
01 02 08 00 01 00 06 00 00 01 00 53 .....S
8 252 8.3991 (0.0019) S>C application_data
48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.
8 253 8.4816 (0.0825) C>S application_data
POST /id/param?channel=12345 HTTP/1.1
8 254 8.4817 (0.0001) C>S application_data
01 02 08 00 01 00 06 00 00 01 00 43 .....C
8 255 8.4834 (0.0016) S>C application_data
48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.
[... ]
7 57 11.8551 (0.0084) S>C application_data
03 00 00 2b 00 18 00 4c 6f 67 69 6e 20 69 6e 63 ...+...Login inc
6f 72 72 65 63 74 0d 0a 6c 6f 67 69 6e 3a 20 correct..login:
4 42 21 6266 (18.4522) C>S alert warning close notify

```

HTTP POST Request for each input letter (no echo)

HTTPS Tunneling – HTTP Tunneling over SSL/TLS Tunnel
 RFC2616 and 2246 Compliant, but security policy compliant ?



HSC HTTP Tunneling Filtering (1): HTTP, an asymmetric protocol

Asymmetric HTTP exchanges

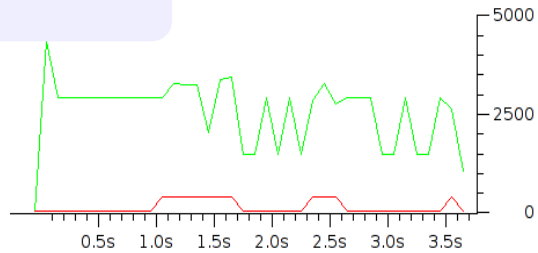
HTTP Request includes a request-line, (optional) header fields and (optional) body – when POST HTTP method is used to submit data to a CGI

HTTP Response contains data: **Content-Length** header indicates the size of the entity-body

```
$ echo -e "GET / HTTP/1.0\r\n\r\n" | nc www.hsc.fr 80
HTTP/1.1 200 OK
[...]
Content-Length: 57445
Connection: close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML lang="fr">
[...]
```

Y-AXIS : TCP segment length (SUM)
 HTTP Server flow (www.hsc.fr)
 HTTP Client flow



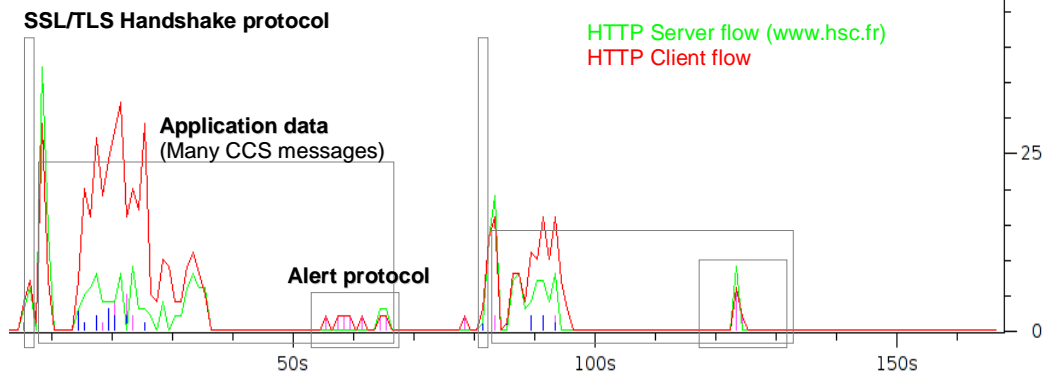
HSC HTTP Tunneling Filtering (2): HTTPS traffic analysis (Packets count)

HTTPS == HTTP over SSL/TLS

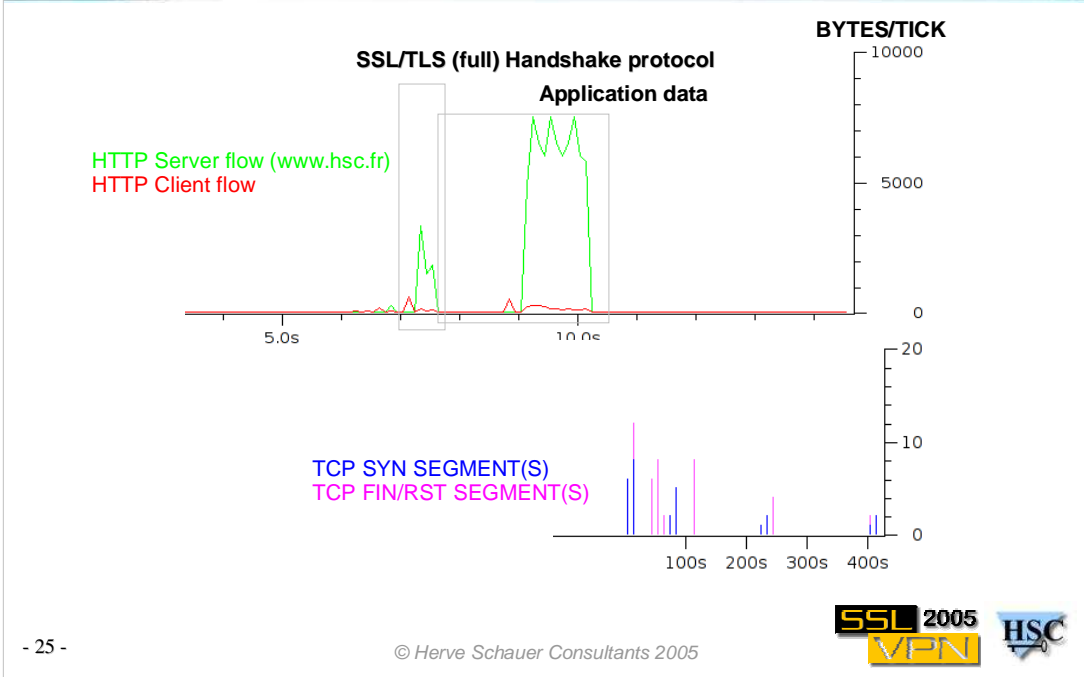
Preliminary HandShake protocol: negotiation of SSL/TLS cryptographic parameters

Application data (HTTP) transmission

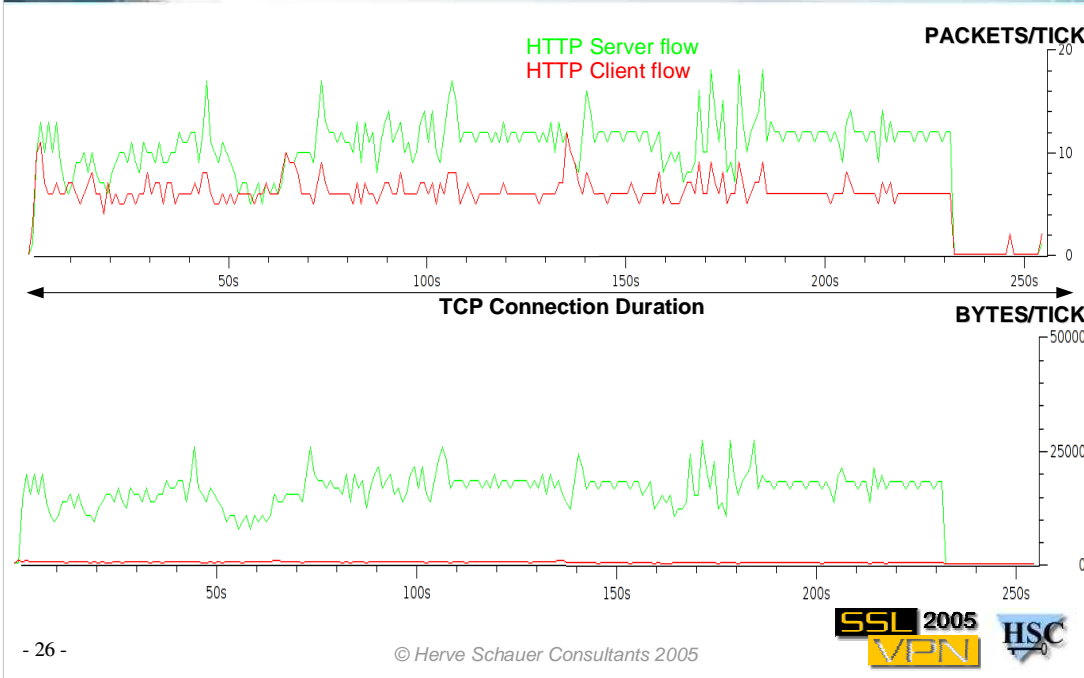
PACKETS/TICK



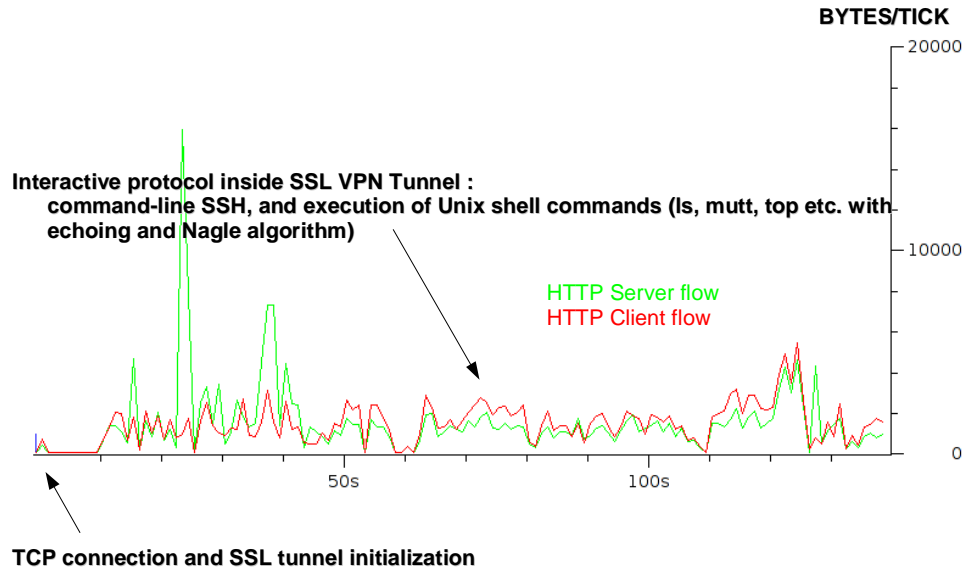
HSC HTTP Tunneling Filtering (3): HTTPS traffic analysis (Bytes count)



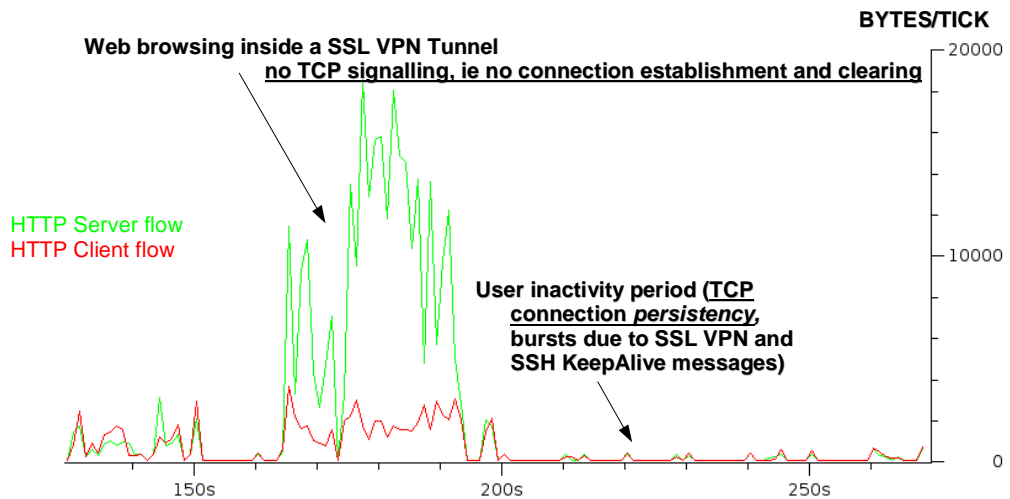
HSC HTTP Tunneling Filtering (4): Download over HTTPS



HSC HTTP Tunneling Filtering (5):
SSL VPN traffic analysis (bytes count)



HSC HTTP Tunneling Filtering (6):
SSL VPN traffic analysis (bytes count)



HSC SSL VPN traffic analysis (7) TCP Connections establishment and clearing

HTTPS Browsing (brief) analysis

In 50s : 56 HTTP Requests

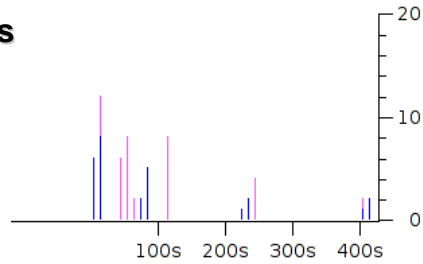
1 SSL/TLS session

14 TCP Connections

with resume handshakes

→ No session in HTTP

→ HTTP 1.1 Keep-Alive mechanism



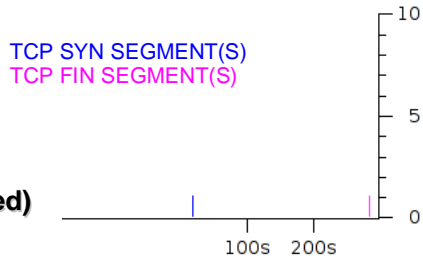
SSL VPN analysis (PPP over SSL model)

In 300s:

1 SSL/TLS session

1 TCP Connection (manually closed)

Unpredictable network behavior



HSC HTTP Tunneling Filtering (8): Tools for SSL/TLS tunnel detection

TCPStatFlow

<http://www.geocities.com/fryxar/>

Analyze TCP connections uptime, amount of inbound and outbound data

Raise an alarm when a threshold is reached :

Cumulative threshold (regarding bytes, number of packets)

Duration threshold (time)

```
$ ./tcpstatflow[2704]: looking for handler for datalink type 1 for interface eth2
./tcpstatflow[2704]: listening on eth2
[...]
./tcpstatflow[2704]: Detecting a new flow: 80.65.225.196:3240->192.168.254.1:443
./tcpstatflow[2704]: 80.65.225.196:3240->192.168.254.1:443: new flow
./tcpstatflow[2704]: Detecting a new flow: 192.168.254.1:33122->66.249.85.104:80
./tcpstatflow[2704]: 192.168.254.1:33122->66.249.85.104:80: new flow
./tcpstatflow[2704]: Detecting a new flow: 192.168.254.1:33123->66.249.85.104:80
./tcpstatflow[2704]: 192.168.254.1:33123->66.249.85.104:80: new flow
./tcpstatflow[2704]: Detecting a new flow: 192.168.254.1:33124->66.249.85.104:80
./tcpstatflow[2704]: 192.168.254.1:33124->66.249.85.104:80: new flow
[...]
Potencial tunnel = 80.65.225.196:3240->192.168.254.1:443:   packets rx=575 tx=548,
                                                           bytes rx=69000 tx=50092,
                                                           seconds=96
```



Conclusion

There is no standard, only proprietary solutions.

HTTP(S) Tunneling – through CONNECT HTTP method, or pure GET/POST Requests wrapping – is *currently* the best way to provide connectivity through HTTPS proxies and bypass firewalls...

... however, tunneling techniques are also implemented by P2P or IM software. In the future (?), security policies should be enforced by border equipments not only with static or dynamic IP filtering, but also with statistical techniques.



References

HSC tips and presentations

<http://hsc.fr>

Ethereal – A Network Protocol Analyzer

<http://ethereal.com>

SSLDump

<http://www.rtfm.com/ssldump/>

TCPStatFlow

<http://geocities.com/fryxar/>

Gray-World.NET – *Unusual firewall bypassing techniques, network and computer security.*

<http://gray-world.net>