

Proxy Bypassing with a SSL VPN

Version 1.0

Adrián Puente Z.

February 27, 2008



Abstract

This article define a method to bypass the corporate proxy to help Sm4rt's consultants in the Pen-tests. Using the HTTP's connect method you can create a SSL tunnel to an external server outside the corporate network and route all the traffic through it using the HTTPS' CONNECT method that a lot of enterprises proxys has by default.

This article is written only for educational purposes the author nor the company takes any responsibility in the bad use of this information. Please read the disclaimer on page [3](#) for more details.



Licensed by Creative Commons.

Contents

1 Disclaimer	3
2 Introduction	3
2.1 What is a Proxy Server	3
2.2 What is a VPN	3
2.3 What are SSL/TLS Certificates	4
3 SSLTunnel	4
3.1 Why this works?	4
3.2 Implementation	5
3.2.1 Server	5
3.2.2 Client	9
4 Antispurious User Protection	19
4.1 Chatty user	19
4.2 IP/Certificate blocking	19
4.3 Domain Policies	20
4.4 IDS/IPS	20
4.5 Practical case	20
5 Final Ideas	21
5.1 Proxy and Anonymity	21
5.2 Virtualization	22
5.3 Other Tools	22
5.4 Greetings and Shouts	22
6 References	23

1 Disclaimer

The guide is provided by the authors "as is" and any express or implied warranties, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose, are disclaimed. The authors do not warrant that the operation of `ssltunnel` and any software discussed in this guide, whether as the result of following the guide, or otherwise, will be uninterrupted or error-free. The authors do not warrant that the use of the guide will not infringe any copyright, trade secret, patent, or other proprietary or contractual rights of another party. The limitation of liability set forth in this agreement is applicable to any claim that the guide, or any portion thereof, infringes another's copyright, trade secret, patent, or other proprietary or contractual rights. In no event shall the authors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits, or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of or in connection with the use or performance of `ssltunnel` and any software discussed in this guide, whether as the result of following the guide, or otherwise, even if advised of the possibility of such damage.

2 Introduction

This project started by the necessity of having a channel to access external information during an internal pentest. Tools as listed in section 5.3 on page 22 works to connect to an external port that is allowed by the proxy like 80, 8080, 443. Having a service like SSH in an external allowed port allows you to have tunnels that redirects to an squid server on your box.

This tools are awesome but Corkscrew and HTTPtunnel doesn't works with ISA server that asks for an NTLM authentication and `ntlmmaps` just opens a port to the outside so you can't redirect your traffic by routing through this tunnel.

What we want to do is an *almost* undetectable VPN using SSL certificates in both sides to assure privacy

2.1 What is a Proxy Server

A proxy server is a server (a computer system or an application program) which services the requests of its clients by forwarding requests to other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource, available from a different server. The proxy server provides the resource by connecting to the specified server and requesting the service on behalf of the client. A proxy server may optionally alter the client's request or the server's response, and sometimes it may serve the request without contacting the specified server. In this case, it would 'cache' the first request to the remote server, so it could save the information for later, and make everything as fast as possible [1].

2.2 What is a VPN

virtual private network (VPN) is a communications network tunneled through another network, and dedicated for a specific network. One common application is secure communications through the public Internet, but a VPN need not have explicit security features, such as authentication or content encryption. VPNs, for example, can be used to separate the traffic of different user communities over an underlying network with strong security features.

A VPN may have best-effort performance, or may have a defined Service Level Agreement (SLA) between the VPN customer and the VPN service provider. Generally, a VPN has a topology more complex than point-to-point. The distinguishing characteristic of VPNs are not security or performance, but that they overlay other network(s) to provide a certain functionality that is meaningful to a user community[2].

2.3 What are SSL/TLS Certificates

The TLS protocol allows applications to communicate across a network in a way designed to prevent eavesdropping, tampering, and message forgery. TLS provides endpoint authentication and communications privacy over the Internet using cryptography. Typically, only the server is authenticated (i.e., its identity is ensured) while the client remains unauthenticated; this means that the end user (whether an individual or an application, such as a Web browser) can be sure with whom they are communicating. The next level of security?in which both ends of the "conversation" are sure with whom they are communicating?is known as mutual authentication. Mutual authentication requires public key infrastructure (PKI) deployment to clients unless TLS-PSK or TLS-SRP are used, which provide strong mutual authentication without needing to deploy a PKI ¹ [3].

In our case we are going to use mutual authentication to avoid MITM² attacks. Using mutual authentication assure us that the Sysadmin of the network doesn't know the kind of traffic we are routing and make harder to detects this proxy bypass. In the other hand we are making a long lasting SSL/TLS connection to an external IP that can be pointing to a Dynamic DNS domain.

3 SSLTunnel

SSLTunnel allows to mount a PPP session encapsulated into SSL. That allows to make a poor man's VPN between two Unix machines or two networks, without requiring to set up an IPsec technology [4].

3.1 Why this works?

The principle is to use the SSL client certificates, as in HTTPS:

1. The server listens on port 443 of the destination machine;
2. the client connects himself (if need be, through a relay like Squid, ISA-Server, the proxy does not have *ANY* mean to check if it is a navigator ; - ; HTTPS Web server session, because the beginning of the not encrypted session and the SSL negotiation are exactly identical);
3. at the establishment of the connection, the server forks;
4. the server sends its certificate, the client checks that it is well signed by an authority it trusts;

¹In cryptography, a public key infrastructure (PKI) is an arrangement that binds public keys with respective user identities by means of a certificate authority (CA). The user identity must be unique for each CA. The binding is established through the registration and issuance process, which, depending on the level of assurance the binding has, may carried out by software at a CA, or under human supervision. The PKI role that assures this binding is called the Registration Authority (RA) . For each user, the user identity, the public key, their binding, validity conditions and other attributes are made unforgeable in public key certificates issued by the CA.

²In cryptography, a man-in-the-middle attack (MITM) is an attack in which an attacker is able to read, insert and modify at will messages between two parties without either party knowing that the link between them has been compromised. The attacker must be able to observe and intercept messages going between the two victims. The MITM attack can work against public-key cryptography and is also particularly applicable to the original Diffie-Hellman key exchange protocol, when used without authentication.

5. the client sends his certificate;
6. the server checks this certificate and seeks if it corresponds to a certificate declared in its base;
7. the crypted session starts;
8. the server sends its banner with its version number and its protocol version;
9. the client receives the banner, checks and sends his;
10. the client forks, opens a pty, launches pppd in client mode on this pty, without specifying which IP address it wants;
11. the server gets PPP parameters from the user file, changes its identity, opens a pty, forks and launches pppd on this pty with the options given by the file;
12. the PPP session is established between the two ends, the program at each end cyphers/unencrypters and reads/sends the data in the pty connected to pppd.

So the proxy allows the connection with no questions made and the traffic goes encrypted so is difficult for a Sysadmin to know what kind of traffic is being routed inside the tunnel.

3.2 Implementation

The server is the main part of this. This daemon runs as root so he can make PPP connections and routing. Maybe the server can be chrooted or drops privileges, maybe in the next release this can be added.

The server must be running in the 443 server so the SSL/TLS transaction isn't suspicious. The server uses the OpenSSL 0.9.7a libraries that is known to have serious vulnerabilities so is recommended to have the service in a virtual machine isolated from the critical pentest network.

3.2.1 Server

We first download the OpenSSL 0.9.7a libraries from the official site, compile it and install it. Compiling it takes time so we can compile them in a linux box, targzip it and distribute it to other machines. I have done that between i386 machines with Debian/Ubuntu and worked great.

Terminal

```
cd /local/src
wget http://www.openssl.org/source/openssl-0.9.7a.tar.gz
tar zxvf openssl-0.9.7a.tar.gz
cd openssl-0.9.7a
./config && make all test install
```

If everything goes great you should have a new ssl folder in `/usr/local/`. Now we install the server. We need the zlib's libraries so in Ubuntu/Debian we install them using the *aptitude* comand.

Terminal

```
aptitude install zlib1g-dev
wget http://www.hsc.fr/ressources/outils/ssl tunnel/download/ssl tunnel-1.16.tar.gz
tar zxvf ssl tunnel-1.16.tar.gz
cd ssl tunnel-1.16
./configure --with-openssl=/usr/local/ssl/ --disable-client
make all install
```

Three files should be created:

- /usr/local/libexec/pppserver
- /usr/local/etc/ssl tunnel/tunnel.conf.default
- /usr/local/sbin/pppwho

In the server directory we can find a service start script that we must copy to the system initialization directory.

Terminal

```
cp server/pppserver.sh /etc/init.d/
chmod +x /etc/init.d/pppserver.sh
update-rc.d pppserver.sh defaults
```

Now we have installed the service in a Debian/Ubuntu type box, now we have to configure it.

Certificates creation

First we have to become our own Certificate Authority. In cryptography, a certificate authority or certification authority (CA) is an entity which issues digital certificates for use by other parties. It is an example of a trusted third party. CAs are characteristic of many public key infrastructure (PKI) schemes.

A CA issues digital certificates which contain a public key and the identity of the owner. The CA also attests that the public key contained in the certificate belongs to the person, organization, server or other entity noted in the certificate. A CA's obligation in such schemes is to verify an applicant's credentials, so that users and relying parties can trust the information in the CA's certificates.

If the user trusts the CA and can verify the CA's signature, then they can also verify that a certain public key does indeed belong to whoever is identified in the certificate. If the CA can be subverted, then the security of the entire system is lost.

Suppose an attacker, Mallory (to use the Alice and Bob convention), manages to get a CA to issue a false certificate tying Alice to the wrong public key; the corresponding private key is known to Mallory. If Bob subsequently obtains and uses Alice's public key in this (bogus) certificate, the security of his communications to her could be compromised by Mallory - since Bob's messages could be decrypted by Mallory, or he could be tricked into accepting forged signatures from Alice [5].

First we should create a new CA certificate with the scripts OpenSSL has to manipulate certificates:

Terminal

```
/usr/local/ssl/misc/CA.pl -newca
root@bucefalo:~/sslcerts# ls -R
./demoCA:
cacert.pem  certs  crl  index.txt  newcerts  private  serial

./demoCA/certs:

./demoCA/crl:

./demoCA/newcerts:

./demoCA/private:
cakey.pem
```

Now we have the CA certificate, using it we create some server and client certificates signed by the CA [6].

Terminal

```
# For server certificate
/usr/local/ssl/misc/CA.pl -newreq-nodes
# For client certificates
/usr/local/ssl/misc/CA.pl -newreq
# For signing both kind of certificates.
# You have to sign each certificate with the CA.
/usr/local/ssl/misc/CA.pl -sign
```

Configuration

Now we have the private key certificate (*newreq.pem*) and the public key certificate or identity certificate (*newcert.pem*). We have to configure our server with the user's certificate information. Using the next commands we can obtain the exact information from the client certificate.

Terminal

```
/usr/local/ssl/bin/openssl x509 -noout -subject < newcert.pem
/usr/local/ssl/bin/openssl x509 -noout -issuer < newcert.pem
/usr/local/ssl/bin/openssl x509 -noout -fingerprint < newcert.pem
```

Here I put a user example configuration. This information should be in the file *user* in the */usr/local/etc/ssltunnel* directory. We use the fingerprint option to ensure the identity of the client. If we don't use the exact information in the server's user file the connection can't be established.

Config Data

```
# File: /usr/local/etc/ssltunnel/users
user /C=CN/ST=Mexico/L=Mexico City/O=Sm4rt/OU=S.A.R.D./CN=Adrian Puente Z.
fingerprint 6D:65:91:00:A5:C6:F1:CB:10:12:6B:36:4D:E1:01:E3
```

```
issuer /C=CN/ST=Mexico City/O=Sm4rt/OU=S.A.R.D./CN=Sm4rt CA
command /usr/sbin/pppd
pty 1
args 172.16.44.254:172.16.44.1 nodefaultroute nodetach noauth
args lcp-echo-failure 10 lcp-echo-interval 10
```

For each client we have to add the same lines defining a new IP. The format for the IP option is [LocalPP-PIP]:[ClientPPPIP] the others commands are tuned for the pppd. First we copy the server's certificates we have done in the ssltunnel configuration directory:

Terminal

```
cp newcert.pem /usr/local/etc/ssltunnel/server.crt
cp newreq.pem /usr/local/etc/ssltunnel/server.key
cp demoCA/cacert.pem /usr/local/etc/ssltunnel/trusted.pem
```

Now we create the configuration file.

Config Data

```
# File: /usr/local/etc/ssltunnel/tunnel.conf
# Key from the server
keyfile /usr/local/etc/ssltunnel/server.key
# Public certificate for the server
certfile /usr/local/etc/ssltunnel/server.crt
# CA trusted certificate
cacertfile /usr/local/etc/ssltunnel/trusted.pem
# User configuration file
userfile /usr/local/etc/ssltunnel/users
# Server's log file
wtmp /var/log/ssltunnel.wtmp
# Pidfile
pidfile /var/run/pppserver.pid
# timeout for the acceptance of the client's certificate
timeout 20
# Max numbers of users
maxusers 10
# port
port 443
# IP address to listen. You can add many comma separated IPs
# This parameter is optional.
listenaddr 10.33.33.200
# Lock directory for lock files.
lockdir /var/lock/ssltunnel
```

Now we start the ssltunnel service and you should see the port open.

Terminal

```
/etc/init.d/pppserver.sh start

root@bucefalo:~/sslcerts# ss -lpn
Recv-Q Send-Q          Local Address:Port          Peer Address:Port
0         0          10.33.33.132:443             *:*
users: (("pppserver",19246,4))

root@bucefalo:~/sslcerts# ps ax | grep pppserver
19246 ?        Ss      0:00 pppserver: accepting connections
```

Now we have to configure out kernel and pppd to route the client's packet to the external or pentester's network. For this we are going to add the next lines to the /etc/ppp/ip-up script.

Config Data

```
OUTNETWORK='10.33.33.0/24'
echo 1 > /proc/sys/net/ipv4/conf/${PPP_IFACE}/forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s ${OUTNETWORK} -d 0.0.0.0/0 -j MASQUERADE
iptables -A INPUT -p TCP -m state --state RELATED -j ACCEPT
```

Now each time a client connects the ip-up script will reconfigure the kernel and iptables to route the traffic from that client to the external network. OK, now the server is up and running, now we have to configure the client in another box. As CA you have to make the clients' certificates and update the users database in the server's configuration directory.

3.2.2 Client

The client can be installed in Linux/UNIX boxes or Windows. I have tested the client under Ubuntu, Debian and FreeBSD, under FreeBSD, you must install libiconv. On a Windows box you have to create an special interface but the installation is really simple.

Installation on Windows Systems

The installation is simple but you need to do some tricks first. This procedure has 5 parts [7]:

A.- Installing OpenSSL

This part is a ,little tricky because OpenSSL doesn't have precompiled Win32 binaries so you have two ways on doing this:

1. Compile the sources using Cygwin with the mingw compiler for the cross platform. I have compiled it and uploaded it to my microsite: <http://sm4rt.com/ch0ks/>, download and uncompress it on the drive and add the bin directory to the System PATH.

2. You can download the binaries from Stunnel's³ site, download the zip file and uncompress the content into the %System32% directory or any directory in the PATH.

B.- Installing the Driver

Unzip ssltunnel-windows.zip file in any directory on your hard disk, like c:\ssltunnel. You should be able to go easily to this directory from the Windows command line, so I recommend using an easy path.

1. This first step is to install the PPPoP (PPP Over Pipe) HSC NDIS Driver on your system. This driver will appear as an ISDN Network card in the Windows device manager. Go to the Control Panel Folder, Choose < Add Hardware >, click < Next >. Choose < Yes, I have already ... >, click < Next >.
2. In the next screen, scroll down to < Add a new hardware device >, then click < Next >.
3. Choose then < Install the hardware that I >, click < Next >.
4. Choose < Network adapters >, Click Next, then choose < Have Disk >, then < Next >.
5. Choose < Browse >, and select the folder where you unzipped all files, choose < Finish >.

Voilà, the ssltunnel driver is now installed. This diver is going to be used as the interface for the SSLTunnel client. It will emulate a RAS⁴ using the SSL tunnel that the client has made.

C.- Configure ssltunnel.ssc file.

Copy the ssltunnel-sample.ssc file from the distribution to ssltunnel.ssc, then open it with any text editor. This is my configuration file as an example.

Config Data

```
# Sample ssltunnel Windows configuration
#
# ===== Base Params =====
# Tunnel Server IP Address
remotehost vpn.some.place.com
# Tunnel Server Port
port 443
#
# ===== Proxy Params =====
# Use Proxy if 1
useproxy 0
# Proxy Address
proxy 10.1.2.3
# Proxy Port
proxyport 8080
# Proxy User
```

³Stunnel is a program that allows you to encrypt arbitrary TCP connections inside SSL (Secure Sockets Layer) available on both Unix and Windows. Stunnel can allow you to secure non-SSL aware daemons and protocols (like POP, IMAP, LDAP, etc) by having Stunnel provide the encryption, requiring no changes to the daemon's code. <http://www.stunnel.org/>

⁴Remote Access Service. For example a dialup to a ISP or a VPN to an enterprise.

```

proxyuser proxyuser
# Proxy Pass
proxypass proxypass
# User-Agent header
useragent Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
#
#
# ===== SSL Params =====
# Key File
keyfile client.key
# Key Certificate
certfile client.crt
# CA certificate
cacertfile trusted.pem
#
# ===== Networks Params =====
# Network Timeout
timeout 10
# "Phone Number" , should match phone number in Windows
# connection parameters
peer 1
# Proto to use (tls1,tcp,udp)
proto tls1
# Dump more info to stdout if 1
verbose 1
# Restart after close if 1
restart 1
# Set a route to tunnel extremity via current default gw
# before connecting. Help to solve a "chicken and egg" problem
# when ppp given address is on the same subnet as tunnel server.
setroute 0
#
# ===== RAS Parameters =====
# Start automatically ras (not always works)
startras 0
# Start this ras
ras SSLTunnel
    
```

D.- Create a new internet connection.

You should now create a new connection, using the New Connection Wizard provided by Windows 2000 and later.

1. In the Control Panel, click on Network connections and choose « Create a new connection » in the left

- panel, or « New Connection Wizard ». click « Next » and choose "Connect to the Internet".
2. Choose "Set up my connection manually", click « Next ».
 3. Choose « Connect using a dial-up modem », click « Next ».
 4. Choose the first "ISDN Channel PPPoP WAN Adapter", click « Next ».
 5. Give a name to the connection (choose the name you want, it doesn't really matter), click « Next ».
 6. Enter the number you have set in the « peer » parameter in ssltunnel.rc file, Click « Next ».
 7. Choose a user and password (if your tunnel server does not authenticate user using pap or chap, you can enter anything) and uncheck both checkboxes at the bottom, click « Next » and then « Finish ».
 8. Now you should get the "Connect SSL Tunnel" RAS Dialog Box. First you have to change some default parameters in the « Properties » dialog to make SSLTunnel work.
 - (a) Choose « Properties » , then « Networking » tab. In the first drop-down list, verify that the current item is « PPP: Windows ... », then Choose « Settings ». Uncheck all settings (« Enable LCP extensions », « Enable Software compression » and « Negotiate ... »). Click « OK ».
 - (b) In the middle list, select « Internet Protocol (TCP/IP) » then click « Properties ». Choose « Advanced ».
 - Uncheck « Use default gateway on remote network ». If you plan to pass all traffic to the tunnel, you can leave this checked.
 - Check « Use IP Header » compression.
 - (c) If you plan to use an internal DNS server during tunnel connection (for example, to reach an Intranet server), fill the « DNS » tab box.

Connection is now OK, you can choose « Cancel », or go to the next step.

E.- Start connection.

All set now we try to make our first connection to the server. This are the steps.

1. Open a CMD⁵ dialog box, go to directory where SSLTunnel files are located, then type « ssltun.exe ssltunnel.ssc»
2. In the Start Menu, choose Network Connections and then choose the connection just created, then choose « Dial ». Something like this should scroll down in the ssltun.exe window.
3. When you are successfully connected, you can choose « Status » on the connection, and verify you settings typing « ipconfig /all » in a CMD dialog box to display advanced settings such as DNS.
4. When you want to disconnect, the best way is to use the « Disconnect » choice in taskbar, or « Disconnect » button in the status dialog box. You can also hit Ctrl+C in ssltunnel command window, but Windows will probably ask you if you want to connect again.

⁵[Start] -i [Run] -i cmd [enter].

5. If you have left the default setting for the « restart » parameter in ssltunnel.rc file, when you disconnect, ssltun.exe will restart from scratch and will wait for a new connection. To kill the client, please hit Ctrl+C.
6. You should add some routing rules so you can reach the pentester's network. For this you use the next command:

Terminal

```
# route add <Remote IP Network> mask <Network Mask> <IP VPN Box>  
route add 192.168.22.0/24 mask 255.255.255.0 10.33.33.250
```

You can share the connection with other machines using the "Share Connection" property of the RAS device, although the connection isn't as solid as in the Linux/UNIX's client.

Optional: Install as a Service

SSLTunnel can be used as a Windows Service, to let users start connection without launching a console executable nor a GUI. To install service this are the steps:

1. Go to your unzipped directory
2. Copy tunnel.dll and pptunnel.exe to the %SYSTEM32% directory. Type « pptunnel.exe -i » in the Run box to install service.
3. Open regedit, and go to HKLM\System\CurrentControlSet\Services\pptunnel. Create a « Tunnels » subkey. In the Tunnels subkey, create a subkey for each of the tunnels you plan to use (in this case « HSC » and « Rominet »).
4. In each subkey, create a value of type « REG_SZ » with name « configfile ». Fill it with the name of the SSLTunnel configuration file required for this tunnel. Please remember that Certificate paths in the config file should be absolute in that case!
5. In the Services Manager (services.msc), locate « PPPoP tunnel » service, go to the tab « Account » select « Allow service to interact ». This will help you to debug connection during tests: a debug console will be created on the desktop.

You can now launch the service (with control panel or via command line using « net start pptunnel »). Verify that tunnels you have setup are created (you will see something like « Tunnel Name: rominet » for each of them, and SSL initialization). Initialization files are reloaded automatically, but tunnel names are static: if you change registry, you should restart the service.

When you are happy with your setup, you can uncheck « Allow service to interact ... » and configure it to start automatically at boot.

Installation on Linux/UNIX systems

The installation on a Linux/UNIX system is very straight forward. You have to do the same steps as we do on the server installation but with others parameters.

Terminal

```
cd /local/src
wget http://www.openssl.org/source/openssl-0.9.7a.tar.gz
tar zxvf openssl-0.9.7a.tar.gz
cd openssl-0.9.7a
./config && make all test install
cd ..
wget http://www.hsc.fr/ressources/outils/ssltunnel/download/ssltunnel-1.16.tar.gz
tar zxvf ssltunnel-1.16.tar.gz
cd ssltunnel-1.16
./configure --with-openssl=/usr/local/ssl/ --disable-server
make all install
```

It will install the binary `/usr/local/bin/pppclient` and it should be on your `PATH`. Now we have to configure the client. I recommend to create a `./ssltunnel` directory where you can have multiple configuration types that you can use for different networks or `pppd` routing. Here is a sample configuration for the client, please check the comments I put on it to simple the configuration task.

Config Data

```
# File: ~/.ssltunnel/ssltunnel-sm4rt
# You can use this option to debug the app.
verbose                                1
# The remote host you are connecting to
remotehost                             vpn.some.place.com
# The port you are connecting in the remote host
port                                    443
# The PATH for the ppp daemon.
localppp                               /usr/sbin/pppd
# This is useful in BSD systems
bsdppp                                  0
# This param goes to the pppd daemon and is
# is usefull to set different routing configurations
ipparam                                 SSLVPN
# Basically means that a particular machine will
# respond to ARP requests for hosts other than itself.
# useful when the ppp network has the same address
# as the customer's
localproxyarp                           0
# Local echo interval with 0 we desactivate it.
localechoint                            10
# Maximun echo that we can lose.
localechofail                           10
# Start pppd as debug.
localdebug                               1
```

```
# Timeout before disconnect.
timeout                20
# Activates/deactivates the authentication in the local proxy
useproxy              0
# IP of the proxy. Only set if useproxy is 1
proxy                 10.1.2.3
# The port of the proxy
proxyport             8080
# The username of the proxy
proxyuser             proxyuser
# The password of the proxy
proxypass            proxypass
# The CA's certificate
cacertfile            /home/apuente/.ssltunnel/trusted.pem
# The Certificate's key
keyfile               /home/apuente/.ssltunnel/client.key
# The user certificate
certfile              /home/apuente/.ssltunnel/client.crt
# Binary option that forks the pppclient command in the background
daemon               0
# Binary option that retries the connection if it fails.
autoreconnect        1
# The client logfile.
logfile              /var/tmp/pppclient.log
```

You need to install the client's and the CA certificate you've done in the `/.ssltunnel` directory and has to match the PATH you configured in the configuration file. Tip: the file `newcert.pem` is the `client.crt` and the `newreq.pem` is the `client.key`. After all this is done you can test the connection to the server.

If something fails you always can use the server's logs. In most of the cases the problems are in the users database and is because the subject, issuer and/or fingerprint doesn't match.

Routing configuration

Here comes the tricky part where we make the routing configuration. I'll talk about three main scenarios:

Routing Client

This configuration is useful to avoid being detected but stills you can search in the internet or have a pager connected. Your traffic is routed into the enterprise traffic and can access servers in the routed You can have a Squid proxy installed inside your network and use it on some applications as the browser and the pager avoiding generating too much traffic and become suspicious.

Do you remember the `ipparam` option on the `ssltunnel` client's configuration file? Here is where we are

going to use it.

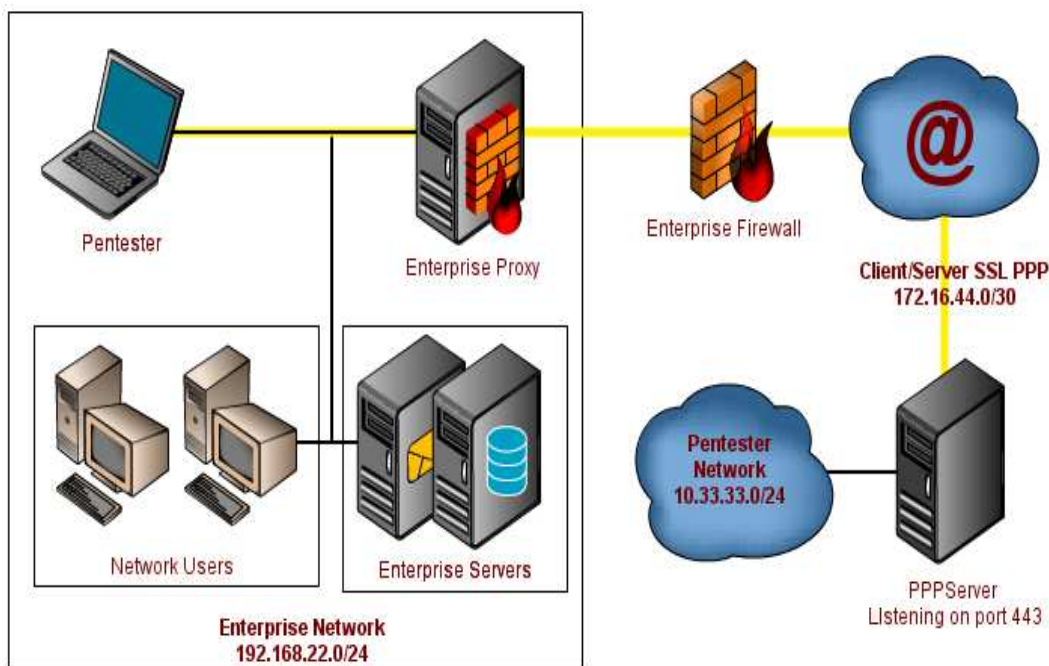


Figure 1: SSLTunnel uses SSL on port 443 to bypass the enterprise proxy and firewall.

Now, see the figure 1 and you can see there are 3 networks to work with:

Customers: 192.168.22.0/24

The SSL VPN: 172.16.44.0/30. This is a small network (2 host) and is configured in the server's configuration file.

Pentesters': 10.33.33.0/24. This is the network we should have set in the server's /etc/ppp/ip-up OUT-NETWORK variable.

So, in the client's /etc/ppp/ip-up file you have to put the next code:

```


Config Data



```

else if [${PPP_IPPARAM} = 'justrouting']; then
 route add -net 10.33.33.0/24 ${PPP_IFACE}
 # You can add another net if it is a Virtual Server with NAT.
 # route add -net 192.168.222.0/24 gw 10.33.33.254
fi

```


```

With this configuration you can reach the SSLTunnel server that can have a Squid proxy or some tools, just use your imagination.

Leech Client

This configuration is useful if you want to bypass the corporate proxy network. We use it in Pentest to get new tools or search for information about a specific system. All the traffic is routed through the tunnel but you can still route traffic inside the enterprise network.

Watch again the 1 and now think... How can you use the SSLTunnel server as a proxy and automate the configuration so it works each time you make the SSL VPN? The trick consists in using the customer's gateway just as a host gateway to the SSLTunnel server IP and use the SSLTunnel IP as the default gateway. Tricky isn't it? Here is the code that makes the magic.

Config Data

```
if [ ${PPP_IPPARAM} = 'road' ]; then
    IP_DNS=`grep nameserver /etc/resolv.conf | head -1`
    IP_SERVER=`host vpn.some.place.com ${IP_DNS} | \
awk '( /vpn.some.place.com/ ){print $4}'`
    IP_GW=`route -vn | awk '( /^0.0.0.0/ ){print $2}'`
    route add -net 10.33.33.0/24 ${PPP_IFACE}
    route add -net 192.168.222.0/24 gw 10.33.33.254
    route add -host ${IP_SERVER} gw ${IP_GW}
    echo nameserver 192.168.222.254 >> /etc/resolv.conf
    route del default gw ${IP_GW}
    route add default gw 10.33.33.250
fi
```

Network 2 Network

This configuration can be helpful when you have part of the staff outside and you need reinforcements. This is the most complicated configuration but you can add the customer's network to yours and work from the office using the client's box as a router. The problem is that it generates a lot of traffic and you need a computer plugged in the customer's network.

Imagine to connect the customer's network with yours and access the enterprise servers as naturally as you use Gmail POP3 to read your mail. The solution has two approaches:

1. Both networks are yours and want to make a connection so you can add special ACLs⁶ in your router so when he gets traffic to the other network uses the SSLTunnel server as gateway and the same configuration in the other side, as a VPN on both sides should do.
2. You have created special ACLs in your side of the network but you can't manipulate customer's router so you should use NAT⁷ in the client box so you can reach the customer's servers.

⁶In networking, ACL refers to a list of rules detailing service ports or (network) daemon names that are available on a host or other layer 3 device, each with a list of hosts and/or networks permitted to use the service. Both individual servers as well as routers can have network ACLs. Access control lists can generally be configured to control both inbound and outbound traffic, and in this context they are similar to firewalls.

⁷In computer networking, Network Address Translation (NAT, also known as Network Masquerading, Native Address Translation or IP Masquerading) is a technique of transceiving network traffic through a router that involves re-writing the source and/or destination IP addresses and usually also the TCP/UDP port numbers of IP packets as they pass through. Checksums (both IP and TCP/UDP) must also be rewritten to take account of the changes. Most systems using NAT do so in order to enable multiple hosts on a private network to access the Internet using a single public IP address (see gateway). Many network administrators find NAT a convenient technique and use it widely. Nonetheless, NAT can introduce complications in communication between hosts and may have a performance impact.

This is the configuration you have to do for the network in figure 1 [8] on the server [9]. NOTE: You are going to use this configuration based in the client's certificate, remember that you give an IP per client.

Server /etc/ppp/ip-up

```
# Server's /etc/ppp/ip-up
# The IP of the Network 2 Network client is 192.168.22.1 and
# you associated a Certificate with this IP in the users database
if [ ${PPP_REMOTE} = '192.168.22.1' ]; then
    DEV=eth0
    REMOTE_NET=192.168.22.0/24
    echo 1 > /proc/sys/net/ipv4/conf/${PPP_IFACE}/forwarding
    echo 1 > /proc/sys/net/ipv4/ip_forward
    route add -net ${REMOTE_NET} ${PPP_IFACE}
    /sbin/iptables -t nat -A POSTROUTING -o ${DEV} -j MASQUERADE
    /sbin/iptables -A FORWARD -i ${PPP_IFACE} -o ${DEV} -m state \
--state RELATED,ESTABLISHED -j ACCEPT
    /sbin/iptables -A FORWARD -i ${DEV} -o ${PPP_IFACE} -j ACCEPT
fi
```

Now in the client we have to do NAT so the packets doesn't get lost in the customer's network. The problem with route vs NAT is that when we route the packets the receiver has to know the origin of the packet and if he doesn't has a route to the origin's network the packet gets lost, when we NAT we make the client to send the packet as if he is the origin having a routable and known IP on the network, so when the receiver answers the packet the client reroutes the packet to the real origin in the network he knows.

Remember the ipparam parameter in the ssltunnel client configuration file? Here is an example of it working. So using that parameter you can have multiples configurations for different kinds of networks.

Client /etc/ppp/ip-up

```
# Client's /etc/ppp/ip-up
# The ipparam of the Network 2 Network client is net2net.
# Remember to change the interface depending on your needs.
if [ ${PPP_IPPARAM} = 'net2net' ]; then
    DEV=eth0
    REMOTE_NET=10.33.33.0/24
    echo 1 > /proc/sys/net/ipv4/conf/${PPP_IFACE}/forwarding
    echo 1 > /proc/sys/net/ipv4/ip_forward
    route add -net ${REMOTE_NET} ${PPP_IFACE}
    # route add -net 192.168.234.0/24 gw 10.11.11.250
    /sbin/iptables -t nat -A POSTROUTING -o ${DEV} -j MASQUERADE
    /sbin/iptables -A FORWARD -i ${DEV} -o ${PPP_IFACE} -m state \
--state RELATED,ESTABLISHED -j ACCEPT
    /sbin/iptables -A FORWARD -i ${PPP_IFACE} -o ${DEV} -j ACCEPT
fi
```

Remember that you can add more networks depending on your topology. Now the problem goes with the external nodes that in this case has two profiles: The pentest team is ready to attack and the customer's machines waiting to be owned. On the pentest team's machines you have to set some routing rules (you can avoid this making the adjustments on the default router). Here are three configurations, remember you are using the IP of the VPN box on your network as gateway for the customer's network.

On Windows:

Terminal

```
# route add <Remote IP Network> mask <Network Mask> <IP VPN Box>  
route add 192.168.22.0/24 mask 255.255.255.0 10.33.33.250
```

On Linux:

Terminal

```
# route add -net <REMOTE NETWORK> gw <IP gateway box>  
route add -net 192.168.22.0/24 mask 255.255.255.0 10.33.33.250
```

That's it, you should be reaching the machines in the customer's network easily.

4 Antispurious User Protection

Imagine you are a Sysadmin and want to protect the network from an evil user. How can you notice someone is playing with SSL to bypass the proxy? What behaviour has the network with this kind of traffic?

Most of the comercial proxy servers with a default configuration won't be able to stop this kind of traffic. For example the ISA server from Microsoft can authenticate the network user with NTLMv2⁸ making really difficult for an attacker to eavesdrop the user and password of the domain but by itself he can't detect if the traffic is a SSL VPN traffic.

This are some tips you can use to to protect your network:

4.1 Chatty user

Since the user has to route large ammount of traffic through the SSLTunnel you can use the Ntop⁹ to find excesive traffic from one machine to an external IP connected to the port 443. As SSLTunnel uses TCP so you should suspect for long lasting sessions also.

4.2 IP/Certificate blocking

It's not the best idea but you can block all connections to services whos CA's certificates you doesn't know or block those networks you know are dinamyc from a dialup or PPPoE service. Yeah I know, it's not the best idea but it can work.

⁸NTLM (NT LAN Manager) (not to be confused with LAN Manager) is a Microsoft authentication protocol used with the SMB protocol. MS-CHAP is similar and is used for authentication with Microsoft remote access protocols. During protocol negotiation, the internal name is nt lm 0.12. The version number 0.12 has not been explained. It is the successor of LANMAN (Microsoft LAN Manager), an older Microsoft authentication protocol, and attempted to be backwards compatible with LANMAN. <http://en.wikipedia.org/wiki/NTLM>

⁹is a network probe that shows network usage in a way similar to what top does for processes. In interactive mode, it displays the network status on the user's terminal. In Web mode, it acts as a Web server, creating an HTML dump of the network status. It sports a NetFlow/sFlow emitter/collector, an HTTP-based client interface for creating ntop-centric monitoring applications, and RRD for persistently storing traffic statistics. <http://www.ntop.org/>

4.3 Domain Policies

You can create a GPO¹⁰ that blocks any application that opens a connection to the 443 that is not part of the white list application for this service (Browser, Specialized Application). I believe some centralized antivirus has this feature.

4.4 IDS/IPS

You can create some rules to detect and destroy those connectios that seems exrange. You can recolect enough information in the firewall and IDS logs to act legaly against the user.

4.5 Practical case

This is an example on how I would search for an spurious user:

1. I would check in the Ntop logs for long lasting tcp sessions on 443.
2. Suspecting on someone I whould make some snort rules alerting me on specific events from this user. Even you can make alerts that emails you when the incident happens.
 - (a) On 443 tcp connections.
 - (b) On OpenSSL certificates.
3. Having some alerts, trace the external IP to know where it is, who owns it, etc.
4. Tracing for a couple of days the user and the external IPs he connects to you can hopefully have enough data to have a case and do a forensic on the spurious user's machine.

Here are some examples on the snort rules you can use.

Config Data

```
# Connections on the 443 port. Not very useful but is a start.
alert tcp any any -> any 443 (msg:"SSL Connection"; sid:313371;)

# An example rule on NoIP DNS query. You can add many others but there are
# more than a thousand Dinamyc DNS services**.
alert udp any any -> any 53 (msg:"NoIP DNS Query"; content:"no-ip"; \
nocase; sid:313372;)

# More useful example rule. You know the SSLTunnel uses OpenSSL certs
alert tcp any any -> any 443 (msg:"OpenSSLCert"; \
content:"OpenSSL Generated Certificate"; nocase; sid:313373;)
```

**11

This are some general rules but you can tweak them you make a really powerfull SSLTunnel IDS appliance [10].

¹⁰Group Policy is a feature of Microsoft Windows NT family of operating systems that provides centralized management and configuration of computers and remote users in an Active Directory environment. It is part of Microsoft's IntelliMirror technologies which aim to reduce the overall cost of supporting users of Windows. These technologies relate to management of disconnected machines or roaming users and include Roaming user profiles, Folder redirection and Offline files.

¹¹<http://www.technopagan.org/dynamic/>

5 Final Ideas

Now you should have a SSLTunnel network up and running. What else you can do with it?

5.1 Proxy and Anonymity

In my SSLTunnel server I installed a Squid Proxy Server¹² to reroute my Firefox¹³, Skype¹⁴ and Pidgin¹⁵ when I am using the "Routing Client" configuration, explained in section 3.2.2 on page 15.

Config Data

```

http_port 8080
icp_port 3130
hierarchy_stoplist cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY
acl apache rep_header Server ^Apache
broken_vary_encoding allow apache
cache_mem 8 MB
access_log /var/log/squid/access.log squid
hosts_file /etc/hosts
refresh_pattern ^ftp:          1440      20%      10080
refresh_pattern ^gopher:       1440      0%       1440
refresh_pattern .               0         20%     4320
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl purge method PURGE
acl CONNECT method CONNECT
# The next 3 lines makes Squid an "all open proxy"
http_access allow all
http_reply_access allow all
icp_access allow all
cache_effective_group proxy
coredump_dir /var/spool/squid
    
```

¹²Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests, to caching web, DNS and other computer network lookups for a group of people sharing network resources, to aiding security by filtering traffic. Although primarily used for HTTP and FTP, Squid includes limited support for several other protocols including TLS, SSL, Internet Gopher and HTTPS. The development version of Squid (3.1) includes IPv6 and ICAP support. <http://www.squid-cache.org/>

¹³Mozilla Firefox is a web browser, gopher client and FTP client project descended from the Mozilla Application Suite, managed by the Mozilla Corporation. <http://www.mozilla.com/en-US/firefox/>

¹⁴Skype (IPA: [skaip], rhymes with type) is a software program created by the Swedish and Danish entrepreneurs Niklas Zennström and Janus Friis. Skype allows users to make telephone calls over the internet to other Skype users free of charge, or to landlines and cell phones for a fee. <http://www.skype.com/>

¹⁵Pidgin (formerly named Gaim) is a multi-platform instant messaging client that supports many commonly used instant messaging protocols. <http://www.pidgin.im/>

5.2 Virtualization

You can implement this solution on a Damn Small Linux¹⁶ with the correct scripts to configure the SSLTunnel and have it running with QEMU¹⁷ that works the same in Linux as in Windows. DSL comes with an script to run it using QEMU and occupies 50 Mb.

If you compromise a computer inside the network you can load the VMPlayer with a SSLTunnel client, just change the MAC Address to make more difficult being detected as a virtual machine.

5.3 Other Tools

Before finding this awesome tool I tested others, here are a small list:

HTTP Tunnel: `httptunnel` creates a bidirectional virtual data connection tunnelled in HTTP requests.

The HTTP requests can be sent via an HTTP proxy if so desired. <http://www.nocrew.org/software/httptunnel.html>

Corkscrew: Corkscrew is a tool for tunneling SSH through HTTP proxies. <http://www.agroman.net/corkscrew/>.

NTLM Authorization Proxy Server: NTLM Authorization Proxy Server (APS) is a proxy software that allows you to authenticate via an MS Proxy Server using the proprietary NTLM. <http://ntlmaps.sourceforge.net>.

Tunneling SSH over HTTP(S): This document explains how to set up an Apache server and SSH client to allow tunneling SSH over HTTP(S). This can be useful on restricted networks that either firewall everything except HTTP traffic (tcp/80,tcp/443) or require users to use a local (HTTP) proxy. <http://dag.wieers.com/howto/ssh-http-tunneling/>.

Proxytunnel: ProxyTunnel is a program that connects stdin and stdout to a server somewhere on the network, through a standard HTTPS proxy. We mostly use it to tunnel SSH sessions through HTTP(S) proxies, allowing us to do many things that wouldn't be possible without ProxyTunnel. <http://proxytunnel.sourceforge.net/>.

5.4 Greets and Shouts

- Thanks to Alain Thivillon¹⁸ for his awesome tool!
- Thanks to David J. Bianco for his cool paper on Snort Rules [10].
- Thanks to my friend Dan Clemens from Packetninjas L.L.C¹⁹ for sharing some jutsus on Snort rules, IDS experience and 1337ness.

¹⁶Damn Small Linux or DSL is a free Linux distribution for the X86 family of personal computers. It was designed to run graphical applications on older PC hardware – for example, machines with 386/486/early-Pentium processors and very little memory. DSL is a LiveCD with a size of 50 MB. What originally started as an experiment to see how much software could fit in 50 MB eventually became a full-fledged Linux distribution. It can be installed on storage media with small capacities, like bootable business cards, USB flash drives, various memory cards, and Zip drives. <http://www.damnsmalinux.org/>

¹⁷QEMU is a processor emulator that relies on dynamic binary translation to achieve a reasonable speed while being easy to port on new host CPU architectures. In conjunction with CPU emulation, it also provides a set of device models, allowing it to run a variety of unmodified guest operating systems, thus it can be viewed as a hosted virtual machine monitor. It also provides an accelerated mode for supporting a mixture of binary translation (for kernel code) and native execution (for user code), in the same fashion as VMware Workstation and Microsoft Virtual PC. Qemu can also be used purely for CPU emulation for user level processes, in this mode of operation, it is most similar to valgrind. <http://fabrice.bellard.free.fr/qemu/>

¹⁸<http://www.hsc.fr/ressources/outils/ssltunnel/index.html.en>

¹⁹<http://www.packetninjas.net/>

- Thanks to Sm4rt Security Services that allows me to take some work time on making new techniques and this kind of papers.
- Thanks to all the enterprises which infrastructure I used to test my configurations.
- Thanks to The Servibanda Team that has always encouraged me to acquire more 1337ness.

References

- [1] WIKIPEDIA. *Proxy server*. Online Enciclopedia. [http://en.wikipedia.org/wiki/Proxy_server. Visited: February 27, 2008].
- [2] WIKIPEDIA. *Virtual private network*. Online Enciclopedia. [http://en.wikipedia.org/wiki/Virtual_private_network. Visited: February 27, 2008].
- [3] WIKIPEDIA. *Transport Layer Security*. Online Enciclopedia. [http://en.wikipedia.org/wiki/Transport_Layer_Security. Visited: February 27, 2008].
- [4] HSC CONSULTANTS. *SSLTunnel*. HSC consultants Tools. [<http://www.hsc.fr/ressources/outils/ssltunnel/index.html>. Visited: February 27, 2008].
- [5] WIKIPEDIA. *Certificate authority*. Online Enciclopedia. [http://en.wikipedia.org/wiki/Certificate_authority. Visited: February 27, 2008].
- [6] DR.-ING. LUTZ JI₂NICKE. *Postfix/TLS - Lutz's very short course on being your own CA*. Postfix Howto. [http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/doc/myownca.html. Visited: February 27, 2008].
- [7] HSC CONSULTANTS. *SSLTunnel Windows Client, Installation Guide, Version 1.16*. HSC consultants Tools PDF Documentation.
- [8] CORWIN LIGHT-WILLIAMS, JOSHUA DRAKE. *Linux PPP HOWTO*. TLDP PPP Howto. <http://tldp.org/HOWTO/PPP-HOWTO/>. Visited: February 27, 2008.
- [9] FRANK WILES. *Quick-Tip: Linux NAT in Four Steps using iptables*. Revolution Systems Publications. [<http://www.revsys.com/writings/quicktips/nat.html>. Visited: February 27, 2008].
- [10] DAVID J. BIANCO. *EZ Snort Rules Find the Truffles, Leave the Dirt*. PDF Documentation.