IPsec L2TP VPN server

From Gentoo Wiki

Contents

Introduction

IPsec/L2TP is another commonly used VPN protocol used in Windows. All version of Windows since Windows 2000 have support built-in, not requiring an external client (like OpenVPN does) making it very convenient. However, it is significantly harder to set up on the server side on Linux, as there's at least 3 layers involved: IPsec, L2TP, and PPP.

This guide will not cover setting up DHCP, RADIUS, Samba or PKI or setting up a Linux client. It does cover some Windows configuration for the purposes of troubleshooting the server

For the purposes of this guide:

- Domain is example.com
- Server name is vpn.example.com
- CA file is called `ca.crt`
- Server cert is `vpn.example.com.crt`
- Server key is `vpn.example.com.key`
- Client cert is `client.example.com.crt`
- Client key is `client.example.com.key`

IPsec

The first layer - and most difficult one - to set up is IPsec. Note IPsec is a peer-to-peer, so in IPsec terminology, the *client* is the called the **initiator** and the *server* is called the **responder**

Windows uses IKEv1 for the process. There are 4 implementation of IPsec in Portage: ipsec-tools (racoon), libreswan, openswan, and strongswan. If you are running *arch*, you need to add the appropriate entries to `/etc/portage/package.keywords`.

ipsec-tools (racoon)

> **Note**
> net-firewall/ipsec-tools (http://packages.gentoo.org/package/net-firewall/ipsec-tools) must be compiled with the **nat** flag, if either the server is behind NAT, or you want to support clients behind NAT

ipsec-tools is the least featured one, but for those coming from *BSD, it may be more familiar. However unlike *BSD, Linux does not use separate interface for ipsec

After installing ipsec-tools, we need to create a few files:

```
root # mkdir /etc/racoon/certs
root # mkdir /etc/racoon/script
```

PSK setup

FILE /etc/racoon/racoon.conf

```
path certificate "/etc/racoon/certs";
path pre_shared_key "/etc/racoon/psk.txt";
path script "/etc/racoon/scripts";

remote anonymous {

        exchange_mode main;
        my_identifier fqdn "vpn.example.com";

        passive on;
        generate_policy on;
        nat_traversal on;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group 14;
        }
}

sainfo anonymous {
        encryption_algorithm aes, 3des;
        authentication_algorithm hmac_sha1, hmac_md5;
        compression_algorithm deflate;
}
```

Each entry in the PSK file consists of an identifier and the key. Windows identifies itself according to its idea its FQDN. The key may be specified as either a string or a hex number (starting with 0x)

FILE /etc/racoon/psk.conf

```
client.example.com 0x87839cfdab5f74bc211de156d2902d128bec3243
```

PKI Setup

Copy `ca.crt`, `vpn.example.com.crt`, and `vpn.example.com.key` to `/etc/racoon/certs`. Make sure `vpn.example.com.key` is not world-readable or writable

FILE /etc/racoon/racoon.conf

```
path certificate "/etc/racoon/certs";
path pre_shared_key "/etc/racoon/psk.txt";
path script "/etc/racoon/scripts";

remote anonymous {

        exchange_mode main;
        my_identifier fqdn "vpn.example.com";

        certificate_type x509 "vpn.example.com.crt" "vpn.example.com.key";
        ca_type x509 "ca.crt";

        passive on;
        generate_policy on;
        nat_traversal on;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method rsasig;
                dh_group 14;
        }
}

sainfo anonymous {
        encryption_algorithm aes, 3des;
        authentication_algorithm hmac_sha1, hmac_md5;
        compression_algorithm deflate;
}
```

Troubleshooting

Creating securiy policies and NAT

The *generate_policy on*; should cause racoon to generate the appropriate Security Policies for us. However, in the presence of NAT (at least, if the server is behind NAT) it doesn't generate the policies quite the way one would hope. So, if no traffic flows over the tunnel, the policy will need to be defined manually:

Polices are created in the `/etc/ipsec-tools.conf` file:

FILE `/etc/ipsec-tools.conf`

```
#!/usr/sbin/setkey -f

flush;
spdflush;
spdadd vpn.example.com[l2tp] 0.0.0.0/0 udp -P out ipsec
        esp/transport//require;
spdadd 0.0.0.0/0 vpn.example.com[l2tp] udp -P in ipsec
        esp/transport//require;
```

Note that, while this policy says we want ALL L2TP traffic going to and from the server to be protected, if there's no Security Association, the traffic will be not protected and will travel the Internet in the usual way - it won't be dropped just because there's no Security Association.

LibreSwan

LibreSwan is a fork of Openswan (itself a fork of FreeS/WAN). It is actually forked by the remaining original developers of Openswan, however after the original developers left Xelerance, a dispute between the original developers and their employer opened up on the name "Openswan".

Its desirable to have each VPN configuration on it own file, this can be done by uncommenting the last line in {Path|/etc/ipsec.conf}:

FILE /etc/ipsec.conf

```
# You may put your configuration (.conf) file in the "/etc/ipsec.d/" directory
# by uncommenting this line
#include /etc/ipsec.d/*.conf
```

NAT traversal is enabled by default in the libreswan config file, so no special configuration needs to be done

PSK setup

A shared key must be created. It may either be specified by a quoted string or by a hex number. (PUT_VPN_SERVER_IP should be the server's IP address. The domain name can be used, but its not recommended by the libreswan deleopers.)

FILE /etc/ipsec.d/vpn.example.com.secret

```
PUT_VPN_SERVER_IP  %any  : PSK 0x87839cfdab5f74bc211de156d2902d128bec3243
```

OR

FILE /etc/ipsec.d/vpn.example.com.secret

```
PUT_VPN_SERVER_IP  %any  : PSK "password_pass"
```

Then create /etc/ipsec.d/vpn.example.com.conf

FILE /etc/ipsec.d/vpn.example.com.conf

```
conn vpnserver
        type=transport
        authby=secret
        pfs=no
        rekey=no
        keyingtries=1
        left=%defaultroute
        leftprotoport=udp/l2tp
        leftid=@vpn.example.com
        right=%any
        rightprotoport=udp/%any
        auto=add
```

PKI setup

Unlike, Openswan, NSS is required. To make things easy, a PKCS#12 bundle should be created containing the servers secret key, the server's certificate and the CA certificate.

user $ openssl pkcs12 -export -certfile ca.crt -inkey vpn.example.com.key -in vpn.example.com.crt -out /etc/ipsec.d/vpn.example.com.p12

The bundle can then be imported into the NSS database:

```
root # cd /etc/ipsec.d
root # pk12util -i somewhere/vpn.example.com.p12 -d .
```

The Libreswan configuration files will refer to the nickname for the imported objects. Use Use **certutil -L -d .** and **certutil -K -d .** to see what they are.

FILE /etc/ipsec.d/vpn.example.com.secrets

```
: RSA "vpn.example.com"
```

vpn.example.com was the nickname obtained via **certutil -K -d .**

FILE /etc/ipsec.d/vpn.example.com.conf

```
conn vpnserver
        type=transport
        authby=rsasig
        pfs=no
        rekey=no
        keyingtries=1
        left=%defaultroute
        leftprotoport=udp/l2tp
        leftcert=vpn.example.com
        leftid=@vpn.example.com
        right=%any
        rightprotoport=udp/%any
        rightrsasigkey=%cert
        auto=add
```

vpn.example.com was the nickname obtained via **certutil -L -d .**

Openswan

Openswan is a fork of the original FreeS/WAN,

Note
Unlike most config files, the OpenSwan config file are extremely picky about whitespace. Tabs, spaces, column numbers and blank lines count!

Note
Openswan is currently package.mask'd in favor of Libreswan

First, we to create some files:

```
root # touch /etc/ipsec.secrets
root # chmod 600 /etc/ipsec.secrets
```

PSK setup

First, we need to create a private key. They may either be specified by a quoted string or by a hex number

FILE /etc/ipsec.secrets

```
PUT_VPN_SERVER_IP  %any  : PSK 0x87839cfdab5f74bc211de156d2902d128bec3243
PUT_VPN_SERVER_IP  %any  : PSK "password_pass"
```

Then create `/etc/ipsec.conf`:

FILE `/etc/ipsec.conf`

```
conn vpnserver
        type=transport
        authby=secret
        pfs=no
        rekey=no
        keyingtries=1
        left=%defaultroute
        leftprotoport=udp/l2tp
        leftid=@vpn.example.com
        right=%any
        rightprotoport=udp/%any
        auto=add
```

PKI Setup

Without NSS

We need to copy the appropriate files to the right places

```
root # cd /etc/ipsec.d
root # cp somewhere/ca.crt cacerts
root # cp somewhere/vpn.example.com.crt certs
root # cp somewhere/vpn.example.com.key private
```

Edit `ipsec.secrets`:

FILE `/etc/ipsec.secrets`

```
: RSA vpn.example.com.key
```

And then edit `ipsec.conf`:

FILE `/etc/ipsec.conf`

```
config setup
        # whatever, but note the position of hash for the comment
        nat_traveral=yes
        # more stuff...

conn vpnserver
        type=transport
        authby=rsasig
        pfs=no
        rekey=no
        keyingtries=1
        left=%defaultroute
        leftcert=vpn.example.com.crt
        leftprotoport=udp/l2tp
        leftid=@vpn.example.com
        right=%any
        rightprotoport=udp/%any
        rightrsasigkey=%cert
        auto=add
```

With NSS

> **Note**
> dev-util/nss (http://packages.gentoo.org/package/dev-util/nss) need to be compiled with the utils flag for these programs to be available. First, create a PKCS12 file with the CA certificate, server certificate, and server key

```
user $ openssl pkcs12 -export -certfile ca.crt -inkey vpn.example.com.key -in vpn.example.com.crt -out vpn.example.p12
```

Then we need to create a NSS database first, them import the certificates and keys into it. Leave the password blank, just hit ENTER twice. After importing,

```
root # cd /etc/ipsec.d
root # certutil -N -d .
root # pk12util -i somewhere/vpn.example.com.p12 -d
```

After importing, Use **certutil -L -d .** and **certutil -K -d .** to see what the "friendly names" are.

FILE /etc/ipsec.secrets

```
: RSA "VPN.EXAMPLE.COM - EXAMPLE.COM"
```

FILE /etc/ipsec.conf

```
config setup
        # whatever, but note the position of hash for the comment
        nat_traveral=yes
        # more stuff...

conn vpnserver
        type=transport
        authby=rsasig
        pfs=no
        rekey=no
        keyingtries=1
        left=%defaultroute
        leftcert="VPN.EXAMPLE.COM - EXAMPLE.COM"
        leftprotoport=udp/l2tp
        leftid=@vpn.example.com
        right=%any
        rightprotoport=udp/%any
        rightrsasigkey=%cert
        auto=add
```

The configuration above works just fine if the the server is behind NAT.

Troubleshooting

strongSwan

strongSwan is a fork of FreeS/WAN (although much code has been replaced). As of strongSwan 5.0, NAT traversal is automatic, no configuration is needed. strongSwan does not create an ipsec.secrets file, thus, one must be created:

```
root # OLD_UMASK="$(umask -p)"; umask 002; >/etc/ipsec.secrets; $(echo ${OLD_UMASK})
```

PSK setup

A shared key must be created. It may either be specified by a quoted string or by a hex number. (PUT_VPN_SERVER_IP should be the server's IP address.

FILE /etc/ipsec.secrets

```
PUT_VPN_SERVER_IP  %any  : PSK 0x87839cfdab5f74bc211de156d2902d128bec3243
```

OR

FILE /etc/ipsec.secrets

```
PUT_VPN_SERVER_IP  %any  : PSK "password_pass"
```

FILE /etc/ipsec.conf

```
conn vpnserver
        type=transport
        authby=secret
        pfs=no
        rekey=no
        keyingtries=1
        left=%any
        leftprotoport=udp/l2tp
        leftid=@vpn.example.com
        right=%any
        rightprotoport=udp/%any
        auto=add
```

When both *left* and *right* are '%any', strongSwan assumes the **l**ocal machine is **l**eft.

PKI Setup

THe certificates and keys must be copied to the appropriate directories

```
root # cp ca.crt /etc/ipsec.d/cacerts
root # cp vpn.example.com.crt /etc/ipsec.d/certs
root # cp vpn.example.com.key /etc/ipsec.d/private
root # chown -R ipsec: /etc/ipsec.d
```

FILE /etc/ipsec.secrets

```
: RSA vpn.example.com.key
```

FILE /etc/ipsec.conf

```
conn vpnserver
        type=transport
        authby=rsasig
        pfs=no
        rekey=no
        keyingtries=1
        left=%defaultroute
        leftprotoport=udp/l2tp
        leftcert=server.crt
        leftid=@vpn.example.com
        right=%any
        rightprotoport=udp/%any
        rightrsasigkey=%cert
        auto=add
```

As before, when both *left* and *right* are '%any', strongSwan assumes the **l**ocal machine is **l**eft.

Troubleshooting

IPsec passthrough / broken NAT

strongSwan does not seem to work with IPsec passthough. It return "cannot respond to IPsec SA request because no connection is known" or (which heavy editing of the config file) INVALID_HASH_INFORMATION error. (This may not be true anymore with strongSwan 5.0)

IPsec Troubleshooting

Server behind NAT

Opening ports

2 port need to be open: UDP port 500 (for ISAKMP) and UDP port 4500 (for NAT Traversal). Forward those to your VPN server

Also IP Protocols (not ports!) 50 (ESP) and 51 (AH) need to be allowed as well, if the router has a per-IP protocol setting (most don't).

IPsec passthough / broken NAT

Many routers have an "IPsec passthrough" options, which can mean 1 of 2 things:

1. Mangle IPsec packets in broken way not compatible with IPsec NAT Traversal
2. Allow all IPsec packets through the router unmolested

If it means (1), DISABLE IPsec passthrough; if it means (2) ENABLE IPsec passthrough
Unfortunately, there are routers that will discard all IPsec traffic, even if the ports are forwarded, and only support method (1). For those with such a router, there are 3 options:

1. Upgrade the firmware, if a newer version is available that behaves properly
2. Open a bug/defect report with the make of the router, if its not EOL/Legacy
3. Get a different router. Linksys and D-link routers are reported behave properly

This tutorial was written with such a router (a Zyxel P-330W) - and (3) is the only option available.

Windows Vista/Server 2008 clients

These OSs do not automatically support IPSec/L2TP servers behind NAT. See http://support.microsoft.com/kb/926179/en-us (http://support.microsoft.com/kb/926179/en-us) for the registry edit to make them support it.

Limitation of Pre-shared keys (PSK)

There is no provision within the IPsec protocol to negotiate PSKs. The only information available to choose which key to use is based on the source and destination IP addresses. Since, in the usual scenario, the responder won't know the initiator's IP in advance, EVERYONE must use the same pre-shared key. Therefore, certificates (PKI) are highly recommended over pre-shared keys (PSK), even for only 1 connection. However generating certificates and creating a PKI is a rather complex process and out of scope of this document.

L2TP

The second layer, l2tp is much easier to setup. Like IPsec, l2tp is a peer-to-peer protocol. The client side called the L2TP Access Concentrator or LAC and the server side is called the L2TP Network Server or LNS

> **Warning**
> L2TP is totally insecure, and must NOT be accessible outside the IPsec connection

If you are using iptables to block all l2tp connection outside the ipsec layer:

```
root # iptables -t filter -A INPUT -p udp -m policy --dir in --pol ipsec -m udp --dport l2tp -j ACCEPT
root # iptables -t filter -A INPUT -p udp -m udp --dport l2tp -j REJECT --reject-with icmp-port-unreachable
root # iptables -t filter -A OUTPUT -p udp -m policy --dir out --pol ipsec -m udp --sport l2tp -j ACCEPT
root # iptables -t filter -A OUTPUT -p udp -m udp --sport l2tp -j REJECT --reject-with icmp-port-unreachable
```

If you are using ufw as your firewall you will need to get ufw to accept incoming & outgoing connections using the ESP protocol to allow ipsec authentication, and to block all l2tp connection outside the ipsec layer edit the following file:

FILE /etc/ufw/before.rules

```
# Allow L2TP only over IPSEC
-A ufw-before-input -m policy --dir in --pol ipsec -p udp --dport l2tp -j ACCEPT
-A ufw-before-input -p udp -m udp --dport l2tp -j REJECT --reject-with icmp-port-unreachable
-A ufw-before-output -m policy --dir out --pol ipsec -p udp --sport l2tp -j ACCEPT
-A ufw-before-output -p udp -m udp --sport l2tp -j REJECT --reject-with icmp-port-unreachable

# Allow IPSEC authentication using ESP protocol
-A ufw-before-input -p esp -j ACCEPT
-A ufw-before-output -p esp -j ACCEPT
```

Using xl2tpd

Unlike other L2TP servers, xl2tpd can maintain an IP address pool without a DHCP or RADIUS server. This is a layering violation, but for small setup its extremely convenient

FILE /etc/xl2tpd/xl2tpd.conf

```
[global]
port = 1701
access control = no

[lns default]
ip range = 172.21.118.2-172.21.118.254
local ip = 172.21.118.1
require authentication = yes
name = LinuxVPN
pppoptfile = /etc/ppp/options.xl2tpd
```

If you want to use a RADIUS or DHCP server, leave off the "ip range" and "local ip" parts. If the connection is unstable, try adding "length bit = yes" to the "lns default" section. If you don't want to use PPP authentication, change "require authentication = yes" to "refuse authentication = yes"

We'll need to create the options file as well

FILE /etc/ppp/options.xl2tpd

```
noccp
auth
crtscts
mtu 1410
mru 1410
nodefaultroute
lock
proxyarp
silent
```

Using rp-l2tp

Configuration of rp-l2tp is simple:

FILE /etc/rp-l2tp/rp-l2tpd.conf

```
# Global section (by default, we start in global mode)
global

# Load handlers
load-handler "sync-pppd.so"
load-handler "cmd.so"

# Bind address
listen-port 1701

section peer
peer 0.0.0.0
mask 0
lns-handler sync-pppd

lns-pppd-opts "noccp auth crtscts mtu 1410 mru 1410 nodefaultroute lock proxyar
p silent"
```

You'll need to specify the server IP as well in lns-pppd-opts (in the form a.b.c.d:). You'll also need to setup pppd to use an IP address pool, either via DHCP or RADIUS.

pppd

Authentication

No authentication

The easiest way to setup is no authentication at all. In that case, make sure "noauth" is specified. Useful for testing purposes, otherwise not recommended - especially if you use PSK. For certain PKI setups, such a configuration may be sensible - for example, all the client machines are under your control, or all the users are trusted and the keys are on machines no one else, besides the users' have access to, or all connections are unattended (using this method to connected multiple sites)

Authentication via chap.secrets

For small users (typically, those wanting to connect their home network from elsewhere), since add the information to chap.secrets

FILE `/etc/ppp/chap-secrets`

```
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
avatar          *       unontanium              *
```

Note
If you are authenticating with domain. the client name will need to be mangled appropriately, in this case, *EXAMPLE\\avatar*

Warning
`/etc/ppp/chap-secrets` contains unencrypted passwords, there ensure only root can read or write it

Authentication via Samba

If you're machine part of (or hosting) an MS Domain or ADS forest, and you are using winbind, Samba can do the authentication. Add plugin *winbind.so* to your ppp options. Setting up Samba and pppd to do this is beyond the scope of this document.

Authentication via RADIUS

If you are running a RADIUS server on the same machine, pppd can use RADIUS. Ensure the **radius** USE flag is set on net-dialup/ppp (http://packages.gentoo.org/package/net-dialup/ppp). Then add *plugin radius.so* to your PPP options. Setting up RADIUS and pppd to do this is beyond the scope of this document.

Authentication via EAP-TLS

If individual users have certificates (not the same as the machine certificate above), you can setup pppd to authenticate via EAP-TLS. You'll want to use this is users authenticate via smartcards or RSA secureID , Ensure the **eap-tls** is set on net-dialup/ppp (http://packages.gentoo.org/package/net-dialup/ppp). RADIUS needs to be setup (see above). You my need to include require-eap in your PPP options file as well. Setting up pppd to do this is beyond the scope of this document.

Troubleshooting

Client Troubleshooting

Windows: Correctly installing the certificate (for PKI users)

The certificate should be packaged in a PKCS12 package. You can do this using openssl or gnutls

```
user $ openssl pkcs12 -export -certfile ca.crt -inkey client.example.com.key -in client.example.com.crt -out client.example.p12
user $ certtool --load-ca-certificate ca.crt --load-certificate client.example.com.crt --load-privkey client.example.com.key --to-p12 --outfile client.example.com.p12
```

Once you have the p12 file, you can import into Windows, however the method is not obvious. DO NOT double-click the key and follow the instructions, that won't work. That imports the key into your personal certificate store, but in Windows, its the local computer that need to so the authentication, so the certificate need to be added to the local computer's store. To do that, we need to use the Microsoft Management Console (mmc). Note you must have Administrator privileges for this to work

Start -> Run -> mmc File -> Add/Remove Snap-in -> Certificates -> Add Computer Account -> Local Computer -> Finish -> OK.

Expand the Certificates. Choose any folder (doesn't matter which), right-click, choose "All Tasks", then "Import". NOW follow the wizard, but on the last step, make sure you choose "Automatically select the certificate store based on the type of certificate"

Windows: RAS networking errors

Error 766: A certificate could not be found

This means you didn't import the certificate correctly. You have to import it though MMC, and not by double-clicking the file.

Error 810: VPN connection not complete

If you're using ipsec-tools (racoon) you may see in you syslog as well: *ERROR: ignore information because ISAKMP-SAhas not been established yet.*

This means you didn't import the certificate correctly, or the p12 bundle is missing the CA certificate. You must do this though MMC, and you MUST choose "Automatically select the certificate store based on the type of certificate" at the end of the import process.

XP SP2 and above: Error 809: Server not responding (Server behind NAT)

Windows XP SP2 and Vista not, by default, connect to server behind a NAT (http://support.microsoft.com/kb/885348%7Cwill). A registry hack is required. Separate fixes are required for Windows XP and Windows Vista.

Vista: Error 835 Could not authenticate

This one occurs only when using PKI. It means the subjectAltName does not match the server you are connecting to. This often occurs when using dynamic DNS - the certificate has the internal name rather than the external name. Either add the external name to the certificate, or disable "Verify the Name and Usage attributes of the server's certificate" in Security -> Networking -> IPsec settings of the connection

Error 741: The local computer does not support required encryption type

Windows will try to negotiate MPPE a (weak) encryption. If you're not using PPP authentication, do not have a pppd with MPPE support, or MPPE supported not compiled into the kernel (or a module), you'll get this error. If you're using authentication, I recommend fixing the pppd or kernel, for minimal configuration problems, even though there's no point to double encryption. If you are not using PPP authentication, or don't want the double encryption, you can disable it by unchecking "Require data encryption" on the Security tab. Note the connection is still protected by IPsec encryption either way, this just disable the requirement for MPPE.

Mac OS X

Mac OS X clients appear to be picky on the proposals they will negotiate with. In particular:

- dh_group must be "modp1024".
- my_identifier must be an address, not an FQDN (address is the default, so just leave that line out from racoon.conf).

Mac OS X won't connect if subjectAltName does not match the server you are connecting to. Unlike Vista you cannot disable this check.

Also, Mac OS X won't connect if the server certificate contains any "Extended Key Usage" (EKU) fields (except for the deprecated ikeIntermediate one). In particular, if you are using certificates from the OpenVPN easy-rsa utility, it adds the "TLS WWW Server" or "TLS WWW Client" EKU, so such certificates will not work. However, you can use such certificates on the Mac OS X client, as it doesn't not care what on the client certificate - only the server.

See also

References

Jacco de Leeuw guide for Freeswan (http://www.jacco2.dds.nl/networking/freeswan-l2tp.html%7CIPsec%7CL2TP)