99_VPN Virtual Private Networks(Tunelling)

Table of Contents

Introduction:

VPN or Tunelling allow to use Internet Gateways and the Internet to join Insecure Private networks together in a Secure way. The Private networks can then communicate transparently and securely between each other via the internet.

This document explains different configurations for some Tunneling requirements using IPSEC and Free S/WAN Package. It is composed of some settings and scripts of my own projects and a reformated document of Jean-Francois Nadeau which explains other tunnelling possibilities.

IPSEC Tunelling:

Requirements:

- The Kernel is compiled with the IPSEC capabilities
- The user space IPSEC implementation package Free S/WAN is installed

Main Components:

- PLUTO — Daemon responsible for the Keys Negociations
- KLIPS — Daemon responsible for the Encrypted Data Transfer
- `/etc/ipsec.conf` — IPSEC configuration file.
- `/etc/ipsec.secrets` — IPSEC RSA Public and Private keys.
- `/etc/init.d/ipsec` — IPSEC Tunnels start/stop script.

`/etc/ipsec.conf` description:

Ths file is the main configuration file of IPSEC. It contains 3 distinct parts:

- A section called `config setup`. Used as general IPSEC setup configurations.
- A section called `conn %default`.
  Used as default configurations which are valid for ALL connections(tunnels).
  It is recommended that all parameters that are common to all connections should be put here.
- A section for each connection called `conn` *`connection-name`*.
  Used to configure individual connections. The *`connection-name`* is the unique name for each connection(tunnel). You can give any name here but it should contain no spaces.
- IMPORTANT: NO empty lines should appear within a section. Each line is either a comment (started by a #) or a valid entry in the form of: `keyword=value`
  Empty lines are only used between sections.
  See `man ipsec.conf` for the description of all configuration entries.

IPSEC Tunelling Notes for the following settings:

- I used SuSE 8.1 system on all sides of the tunnels (The kernel supports already the IPSEC)
- I used the IPSEC implementation Free S/WAN from SuSE 8.1.
- The IPSEC is Version 1.98
- Most of the time the **left** gateway(s) initiated the tunnel(s) buildup and the **right** gateway was passive waiting for **left** gateway(s) to start the tunnel(s).

General preparations:

- Each Gateway needs to have at least in its `/etc/ipsec.conf` the following info:
  - Own local IPSEC RSA Public Key entry
  - Remote Gateway's IPSEC RSA Public Key entry

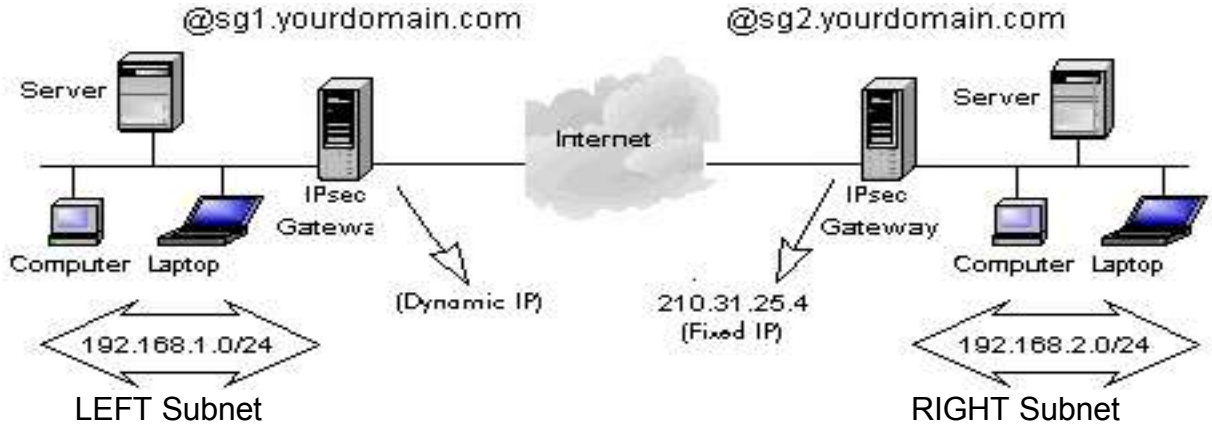
  Each of these entries is a very long single line and MUST stay in a single line. The following steps describe the steps the procedure to obtain this.
- Erase the original set of IPSEC RSA keys on **ALL** IPSEC Gateways :
  ```
  rm /etc/ipsec.secrets
  ```
- Generate new IPSEC RSA keys on **ALL** IPSEC Gateways using the command:
  ```
  ipsec newhostkey --hostname $(hostname -f) \
          --output /etc/ipsec.secrets --bits 2192
  ```
  This will produce a new file called `/etc/ipsec.secrets` which contains the Public and the Private IPSEC RSA keys. This file should have the access mode 640 and belong to root and group root. DO NOT SHARE this file.
- On the **right** IPSEC Gateway:
  - Extract its RSA Public key to the `/etc/ipsec.conf`:
    ```
    ipsec showhostkey --right >> /etc/ipsec.conf
    ipsec showhostkey --right >> /etc/ipsec.pubkey.$(hostname)
    ```
  - Edit the `/etc/ipsec.conf` and move this new RSA Public key line (from the end of the file) to the `conn %default` section:
    ```
    eg.  conn %default
                .........
                rightrsasigkey=0sAQOcZ.......
    ```
- On the **left** IPSEC Gateway:
  - Extract its RSA Public key to the `/etc/ipsec.conf`:
    ```
    ipsec showhostkey --left >> /etc/ipsec.conf
    ipsec showhostkey --left >> /etc/ipsec.pubkey.$(hostname)
    ```
  - Edit the `/etc/ipsec.conf` and move this new RSA Public key line (from the end of the file) to the `conn %default` section:
    ```
    eg.  conn %default
                .........
                leftrsasigkey=0sAQOcZ........
    ```
- Exchange the generated files (`/etc/ipsec.pubkey.$(hostname)`) between Gateways and insert their content to the `/etc/ipsec.conf.` In each IPSEC gateway the `/etc/ipsec.conf` should have the local RSA Public Key entry as well as one RSA public Key entry per remote gateway you want to connect to. These entries should be inserted in the body of each remote connection section.
  ```
  eg.  conn %default
              .........
              rightrsasigkey=0BmAPggLZ........
       conn home-office
              .........
              leftrsasigkey=0sAQOcZ........(eg. public key of home Gateway)
       conn road-office
              .........
              leftrsasigkey=0RtUGdA........(eg. public key of  Laptop)
  ```
- The rest of the document following are common topologies with their respective configurations of the `/etc/ipsec.conf`.

Simple subnet to subnet

Using one fixed and one dynamic internet address.
Description:

- One fixed IP address (on the right gateway)
- One dynamic IP address (given by the ISP on the left gateway)
- The addresses of the ISPs are unknown
- The gateway who has the dynamic address initiates the connection.
- left and right subnets can surf the web via proxy server on both gateways.

Topology:



Left gateway's ipsec.conf

```
config setup
    interfaces=%defaultroute
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    uniqueids=yes

conn %default
    keyingtries=0
    disablearrivalcheck=no
    authby=rsasig
    right=210.31.25.4
    rightsubnet=192.168.2.0/24
    rightnexthop=
    rightrsasigkey=0x-right-public-key
    rightid=@sg2.yourdomain.com
    auto=start

conn sg1-sg2
    left=%defaultroute
    leftsubnet=192.168.1.0/24
    leftnexthop=
    leftrsasigkey=0x-left-public-key
    leftid=@sg1.yourdomain.com
```

Right gateway's ipsec.conf

```
config setup
    interfaces=%defaultroute
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    uniqueids=yes

conn %default
    keyingtries=0
    disablearrivalcheck=no
    authby=rsasig
    right=210.31.25.4
    rightsubnet=192.168.2.0/24
    rightnexthop=%defaultroute
    rightrsasigkey=0x-right-public-key
    rightid=@sg2.yourdomain.com
    auto=add

conn sg1-sg2
    left=0.0.0.0
    leftsubnet=192.168.1.0/24
    leftnexthop=
    leftrsasigkey=0x-left-public-key
    leftid=@sg1.yourdomain.com
```

- Below I explain all parameters in those 2 above configuration files.
  As you may have noticed, both the files are almost the same.
  I underlined the parameters that are different.

ipsec.conf Parameters:

config setup Section

interfaces=%defaultroute

This parameter can be either the interface name (eg. "ipsec0=ppp0") or the magic word %defaultroute. If the interface name is given then the declaration MUST be inside quotes("....") unless the magic word %defaultroute is given.
The interface Must be the first interface normally the packets to the internet.
In the special case of ADSL using PPPoE, the first interface is ppp0 and not eth1.
(ipsec0--->**ppp0**--->eth1--->DSL Modem---->Internet)
IMPORTANT: If we want to use the magic word %defaultroute somewhere else in the configuration file, we must declare the the interfaces=%defaultroute.
Anyway this magic word is accurate for most of the situations.

klipsdebug=none and plutodebug=none

These parameters control how much debugging information is sent from pluto or klips to the syslog as **kernel.info**. The valid values are:

none	Just the minimum connections establishment information.
tunnel	Tunnelling information
tunnel-xmit	Tunnelling transmit only information
pfkey	Userspace communication information
xform	Transform selection and manipulation information
eroute	*eroute* table manipulation information
spi	SA table manipulation information
radij	*radij* tree manipulation information
esp	Encryptions transforms information
ah	Authentication transforms code
rcv	Receive information
ipcomp	IP compression transforms information
verbose	Give even more information
all	Give all possible information (includes all of the above values)

plutoload=%search

This parameter tells which connection(s) should be loaded in its internal database when ipsec is started. Default is none. If the special value %search is used, all connections with auto=add, auto=route, or auto=start are loaded.
In most cases plutoload=%search is correct.

plutostart=%search

This parameter tells which connection(s) should be started or routed when ipsec is started.
Default is none. If the special value %search is used, all connections with auto=route or auto=start are routed(route table is built-up), and all connections with auto=start are routed and started.

uniqueids=yes

If this parameter is set to `yes`, then it insures that if a connection is lost with the remote gateway and then is re-initiated, then the resources of the old connection will be freed. If set to `no` (default) then the same remote gateway can have many connections to the same local gateway for supporting different subnets behind the gateways. In this case it is good practice to regularly(maybe once a day) shutdown the ipsec and bring it immediately back up.

conn %default Section

Note: I placed the parameters of the right gateway here because all of my ipsec connections were do with only one remote gateway.

keyingtries=0

This parameter sets the number of attempts pluto should do to try to establish a successful key exchange when a connection is initiated. (default is 3) A value of 0 sets it to infinite...it never stops trying until it succeeds. In our case, that's what we want.

disablearrivalcheck=no

If this parameter is set to no then it insures that a packet emerging from a tunnel has plausible addresses in its header. Default is yes.

authby=rsasig

This parameter sets the keying mode and the sort of keys used for authentication. Valid values are:

`secret`	for shared secrets (the default)
`rsasig`	for RSA digital signatures

right=210.31.25.4

IP address of the right gateway. In our case since the left gateway initiates the connection, this address must be known (via DNS) or a known fixed IP. The connections initiator gateway may have a dynamic or a fixed IP, but the gateway initiating the connection must know whith whoom it should start a connection. In case this IP is also Dynamic then it is maybe possible to give a domain name gere which would be resolved by a dynamic name server. Either left or right may be % `defaultroute`, but not both.

rightsubnet=192.168.2.0/24

Subnet address behind the right gateway. This is the subnet that will be transparently visible to the left subnet hosts behind the left gateway .

rightnexthop=

This parameter sets the IP number of the next router on the way to the internet from the right gateway. Leave empty if not known. If right parameter is set to `%defaultroute` or if this present value is `%defaultroute` then it is automatically filled in with the default gateway's IP address. As you might notice, the right gateway's `ipsec.conf` has a different setting:
It has **rightnexthop=%defaultroute.**
In our case, although the parameter **right=210.31.25.4** provides the right gateway's fixed IP address, the connection to the internet is done via DSL. The IPS provides a dynamic IP for the next hop which can only be known only when the connection to internet is established.

rightrsasigkey=*0x-right-public-key*

This parameter only makes sence if the **authby** is set to **rsasig.** It lists the RSA public key for the atuthentification of the right gateway. This line can be enormously long but MUST stay all on one line. This key line can be produced easily by issuing the following command on the right gateway: `ipsec showhostkey --right >> /etc/ipsec.conf`

rightid=@sg2.yourdomain.com

This parameter identifies the connection to the other end when it is initiated. (default is right).
Valid values are: IP Number or an FQDN preceeded by the character @. The FQDN does not need to be valid DNS name. This name will not be queried for name resolution. Any name that looks like an FQDN can be given here, as long as it matches the same connection ID on the other side.

auto=start

Tells what should happen when ipsec starts. Valid values are:

`add`	Wait for the remote gateway to initiate the connection.
`route`	Writes the route table for the connection but does not initiate it.
`start`	Writes the route table for the connection and initiates it.

In our case it is set to `start` since left should initiate this connection. If left should initiate all the connections then this parameter can be moved to the section `conn %default`.
Otherwise each connection should set this parameter.

Individual Connections *Sections*

conn sg1-sg2

Identifies the section of parameters for each individual connection. It must be the same name on both sides of the connection. NO spaces are allowed within the name.

left=%defaultroute

Same principle as for left=.... parameter. It provides the IP Address or hostname of the left ipsec gateway.
Valid values are:

- IP address.	eg. `145.46.23.168`
- FQDN ending optionally with a `'.'`	eg. myvpnhost.com
- `0.0.0.0`	In case the IP address in unknowable.
- `%defaultroute`	For dynamic IP address given by the ISP.

The 0.0.0.0 is our case for the `ipsec.conf` of the right gateway, because the dynamic address of the remote left gateway is not knowable in advance. On the gateway which gets the dynamic Internet address, we need to enter the magic word `%defaultroute`.

leftsubnet=192.168.1.0/24

Subnet address behind the left gateway. This is the subnet that will be transparently visible to the right subnet hosts behind the right gateway.

leftnexthop=

Same principle as the `rightnexthop` parameter. Here we leave it empty since it is overwritten by the proper information provoked by the above **left=%defaultroute.** We could also have written **leftnexthop=%defaultroute** in the `ipsec.conf` of the left gateway, but not so in the `ipsec.conf` of the right gateway. Therfore for simplicity's sake both sides are left empty.

leftrsasigkey=*0x-left-public-key*

Same principle as with **rightrsasigkey** except that it contains the RSA Public Key of the left gateway instead. This key line can be produced easily by issuing the following command on the left gateway:

```
ipsec showhostkey --left >> /etc/ipsec.conf
```

leftid=@sg1.yourdomain.com

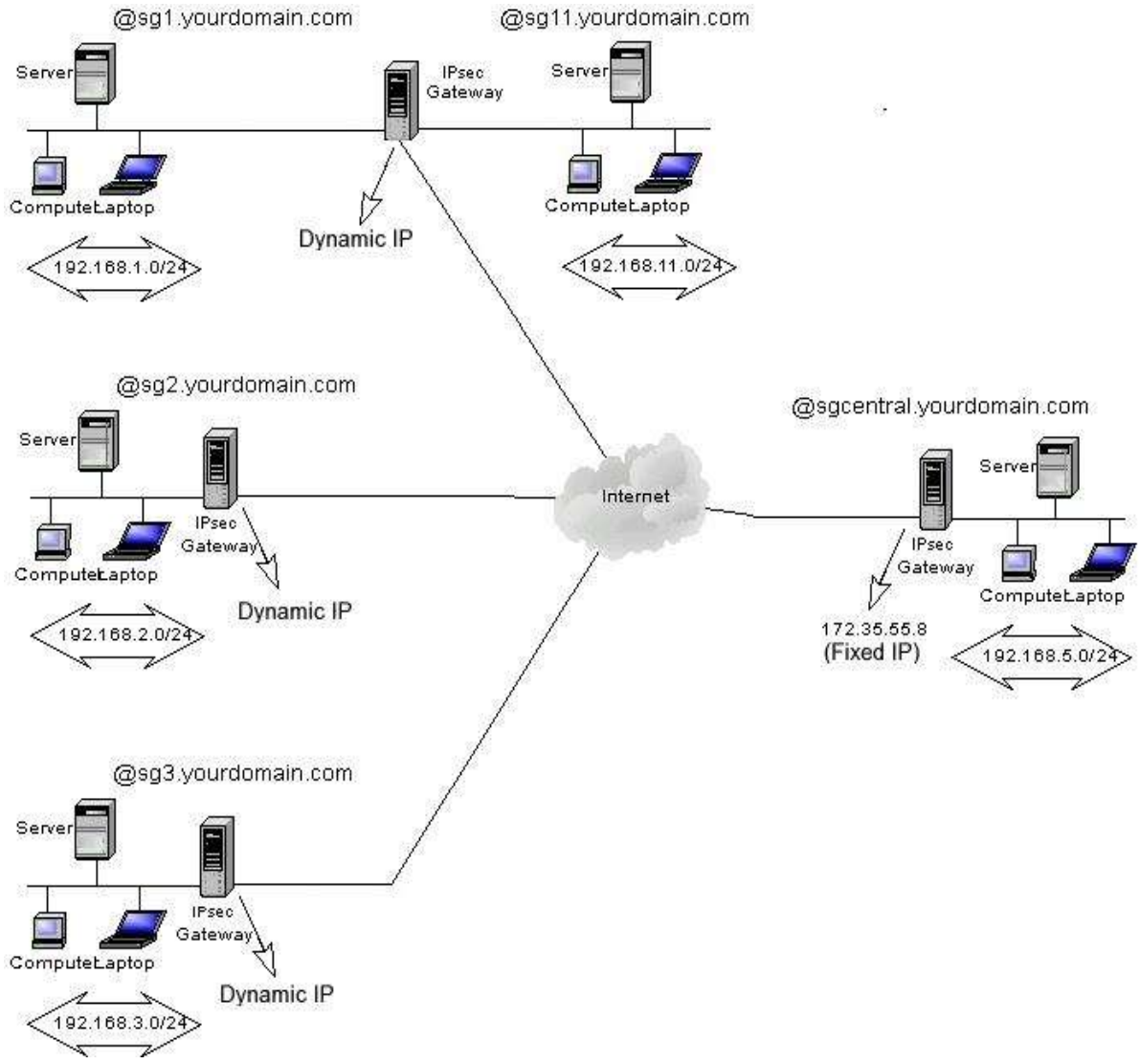
Same principle as **rightid** except that it contains the connection ID of the left gateway.

Multiple satellite subnets to a central subnet

In this example each different satellite subnet is connected to the Internet and surfs via gateways and connects to the central Subnet via IPSEC. The conditions are as follows:

- Gateway of Central Subnet(right) gets a Fixed IP from the ISP (Fixed Ip DSL)
- Gateways of every satellite subnet(left) receives a Dynamic IP from their ISP (DSL)
- The addresses of all the ISPs are unknown
- The satellite gateways who has dynamic IPs are initiating the IPSEC connections.
- ALL subnets (left and right) can surf the web via masquerading set on their gateways.
- All gateways (left and right) are protected from the internet via a firewall.
- Satellite subnets can communicate with the central subnet but not between each other.
- Satellite gateways supports 1 or more subnets behind them.
  This means that one connection section(in `ipsec.conf`) per subnet must exist.

Topology:



Note: There are only 3 satellite gateways shown here, but this configuration can be expanded to as many satellite gateways and subnets behind them as needed.

Here are the `ipsec.conf` configurations of gateways

Left (sg1) **gateway's `ipsec.conf`**

```
config setup
    interfaces=%defaultroute
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    uniqueids=no

conn %default
   keyingtries=0
   disablearrivalcheck=no
   authby=rsasig
   right=172.35.55.8
   rightsubnet=192.168.5.0/24
   rightnexthop=
   rightrsasigkey=right-public-key
   rightid=@sgcentral.yourdomain.com
   auto=start

conn sg1-sgcentral
   left=%defaultroute
   leftsubnet=192.168.1.0/24
   leftnexthop=
   leftrsasigkey=left-sg1-public-key
   leftid=@sg1.yourdomain.com

conn sg11-sgcentral
   left=%defaultroute
   leftsubnet=192.168.11.0/24
   leftnexthop=
   leftrsasigkey=left-sg1-public-key
   leftid=@sg11.yourdomain.com
```

Right(sgcentral) **gateway's `ipsec.conf`**

```
config setup
   interfaces=%defaultroute
   klipsdebug=none
   plutodebug=none
   plutoload=%search
   plutostart=%search
   uniqueids=no
conn %default
   keyingtries=0
   disablearrivalcheck=no
   authby=rsasig
   right=172.35.55.8
   rightsubnet=192.168.5.0/24
   rightnexthop=%defaultroute
   rightrsasigkey=right-public-key
   rightid=@sgcentral.yourdomain.com
   auto=add
conn sg1-sgcentral
   left=0.0.0.0
   leftsubnet=192.168.1.0/24
   leftnexthop=
   leftrsasigkey=left-sg1-public-key
   leftid=@sg1.yourdomain.com
conn sg11-sgcentral
   left=0.0.0.0
   leftsubnet=192.168.11.0/24
   leftnexthop=
   leftrsasigkey=left-sg1-public-key
   leftid=@sg11.yourdomain.com
conn sg2-sgcentral
   left=0.0.0.0
   leftsubnet=192.168.2.0/24
   leftnexthop=
   leftrsasigkey=left-sg2-public-key
   leftid=@sg1.yourdomain.com
conn sg3-sgcentral
   left=0.0.0.0
   leftsubnet=192.168.3.0/24
   leftnexthop=
   leftrsasigkey=left-sg3-public-key
   leftid=@sg1.yourdomain.com
```

Left (sg2) **gateway's `ipsec.conf`**

Same `config setup` & `conn %default` as `sg1`
except the connection is adjusted to sg2

```
conn sg2-sgcentral
   left=%defaultroute
   leftsubnet=192.168.2.0/24
   leftnexthop=
   leftrsasigkey=left-sg2-public-key
   leftid=@sg2.yourdomain.com
```

Left (sg3) **gateway's `ipsec.conf`**

Same `config setup` & `conn %default` as `sg1`
except the connection is adjusted to sg3

```
conn sg3-sgcentral
   left=%defaultroute
   leftsubnet=192.168.3.0/24
   leftnexthop=
   leftrsasigkey=left-sg3-public-key
   leftid=@sg2.yourdomain.com
```

Description of some `ipsec.conf` parameters:

The difference between the parameters of a simple subnet to subnet (already shown) and this one is the fact that the central gateway which needs to be accessible from all the remote subnets, therefore configure a single connection section per remote subnet. The central gateway needs to include the RSA public key of each individual remote gateway in their respective sections. If one gateway is supporting multiple subnets (like sg1/sg11) then the RSA Public key is the same in each remote subnet section in the `ipsec.conf` of the central gateway and in the respective remote gateway.

There is a special parameter (**`uniqueids=no`**)that needs to be set if one remote satellite gateway supports more than one subnet. This will prevent the central gateway to allow only one connection per satellite gateway. The danger with this parameter is that if conections are lost and rebuilt from the satellite gateways then the old connections' resources will not be freed. To unpredictable results the ipsec on the central gateway should regularly be shut down for a short moment and brought back up.
This situation brings an undesirable side-effect. When the central gateway shuts its `ipsec` down for a moment, all the connections to satellite gateways are lost. The solution is to create `cron` jobs, in the satellite gateways, that verifies regularly the integrity of the connection to the central gateway and restarts the `ipsec` if the connection is lost. A example of the `cron` job and scripts is given below.

root crontab:

```
MAILTO=""
-* * * * * /root/check_ipsec_conn
```

Content of `/root/check_ipsec_conn` script:

```
#!/bin/bash
# file: /root/check_ipsec_conn
# author: Pierre Burri
# date:   16-Apr-2003
#-------------------------------------------------------------
# send a ping to check if ipsec connection is still there
int_if=eth0
server_ip=192.168.1.1
my_own_ip=$(/sbin/ifconfig $int_if |/usr/bin/grep "inet [aA]d" |/
usr/bin/cut -d: -f2 |/usr/bin/cut -d"
/bin/ping -c 1 -w 5 -I $my_own_ip $server_ip &> /dev/null
if [ $? -ne 0 ]
then
    /sbin/ifconfig ppp0 &> /dev/null
    if [ $? -eq 0 ]
    then
        # we lost the connection
        /etc/init.d/ipsec restart
    fi
fi
```

IPSEC and Free SWAN Tutorial

(by Jean-Francois Nadeau)

Introduction :

This tutorial describes the configuration needed to setup Ipsec tunnels in various situations such as RoadWarriors and NATed/Firewall gateways.
This document assumes you have already installed Freeswan and knows a bit of the Ipsec/Freeswan terminology. You should read the Freeswan documentation before use of the present document.
My Ipsec gateways are :

- Intel Pentium II 300 Mhz (much more than needed)
- 128 MB of RAM (32 MB is enough)
- Redhat 6.0 (with minor updates such as the nettools, netkit and ipchains packages.)
- Kernel 2.2.14 to 2.2.19.
- Freeswan 1.5 to 1.9
- I use 3DES-MD5 and ESP in all my configurations.
- All configurations are done in tunnel mode.
- I use most default values for rekeying, exept for RoadWarriors.
- The RoadWarrior always initiates the negociation and authentification of the tunnels.

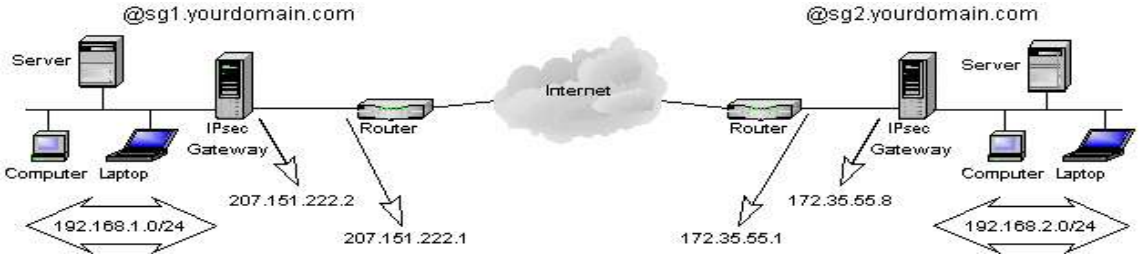
The practical configurations covered are :

- Simple subnet-to-subnet configuration (RSA).
- Subnet-to-subnet configuration with a NATed gateway (RSA).
- RoadWarrior configuration : Freeswan-to-Freeswan (RSA).
- RoadWarrior configuration : NAI's PGPnet-to-Freeswan (PSK).
- RoadWarrior configuration : IRE's Safenet SoftPK-to-Freeswan (PSK).
- Using a central Ipsec gateway as a "tunnel hub".
- Subnet-to-Subnet : Win2000-to-Freeswan (PSK).

Simple Subnet-to-Subnet configuration

(Fixed IP on both Gateways)

This is the simplest case and easiest to setup. We have 2 LANs that we want to link together through the Internet. Each LAN is connected to the internet via router or firewall with static IP adresses.
For example :



Both Ipsec gateways will have the same ipsec.conf configuration file.

ipsec.conf	ipsec.secrets*

config setup
interfaces="ipsec0=eth0"
klipsdebug=none
plutodebug=none
plutoload=%search
plutostart=%search
 	
conn %default
keyingtries=0
 	
conn site1-site2
left=207.151.222.2
leftsubnet=192.168.1.0/24
leftnexthop=207.151.222.1
right=172.35.55.8
rightsubnet=192.168.2.0/24
rightnexthop=172.35.55.1
auto=start
authby=rsasig
leftid=@sg1.yourdomain.com
rightid=@sg2.yourdomain.com
leftrsasigkey=0x-left-public-key
rightrsasigkey=0x-right-public-key
	: rsa {
	# 256 bits, Thu Apr 13 00:29:47 2000
	# for signatures only, UNSAFE FOR ENCRYPTION
	#pubkey=0x01035c3c464da4fb8c9a61fb8c798d91a5
	Modulus: 0x5c3c464da4fb8c9a61fb8c798d91a5d5946
	PublicExponent: 0x03
	# everything after this point is secret
	PrivateExponent:0x3d7d525dbc41525da65e61193848
	Prime1: 0x9c9e5f3bd5d345020052560b8a2a0bd4dd9
	Prime2:
	0x96c34af8e1d95fc3454551fc29f15a4c69b79
	Exponent1: 0x686994d28e8ac0036e407b1715d3893b
	Exponent2: 0x648231fb413b952e5152c6a0e6dd9bcfb
	Coefficient:
	0x05ac43c3b6a5d192f392c521b98334d6
	}

*Not truly valid, you should create a RSA signature of at least 1024 bits on each gateway. And be carefull with the necessary whites spaces.
On each gateway :

```
/usr/local/lib/ipsec/ipsec rsasigkey 1024 >> mykey
```

- Then paste the key in `/etc/ipsec.secrets`. Finally, paste the value from the field #pubkey into the corresponding rsasig key parameter in the `ipsec.conf` file.
- Restart the **ipsec** service on both gateways and observe the logs for any errors.
- This configuration assumes that both Ipsec gateways use the interface eth0 to reach the internet. Most default values were used here for simplicity.
- Both gateways must have IP forwarding turned on to allow packets from the leftsubnet to reach the rightsubnet and vice-versa.
- Ipchains must be installed and used to permit traffic from leftsubnet to rightsubnet.
- Remember that Ipsec is as secure as your gateways are. I recommend to only accept Ipsec traffic on the interface visible to Internet.

Firewall settings on the left gateway :

IPCHAINS

```
# Default policies
/sbin/ipchains -P input ACCEPT
/sbin/ipchains -P forward DENY

# Only allow ipsec traffic, ESP and AH from and to the Internet
/sbin/ipchains -A input -p UDP -d 207.151.222.2/32 500 -j ACCEPT
/sbin/ipchains -A input -p 50 -d 207.151.222.2/32 -j ACCEPT
/sbin/ipchains -A input -p 51 -d 207.151.222.2/32 -j ACCEPT

# Allows internal subnet access
/sbin/ipchains -A input -s 192.168.1.0/24 -j ACCEPT

# Allows traffic from and to internal LANs
/sbin/ipchains -A forward -b -s 192.168.1.0/24 -d 192.168.2.0/24 -j ACCEPT

# Default input policy back to deny
/sbin/ipchains -P input DENY
```

IPTABLES

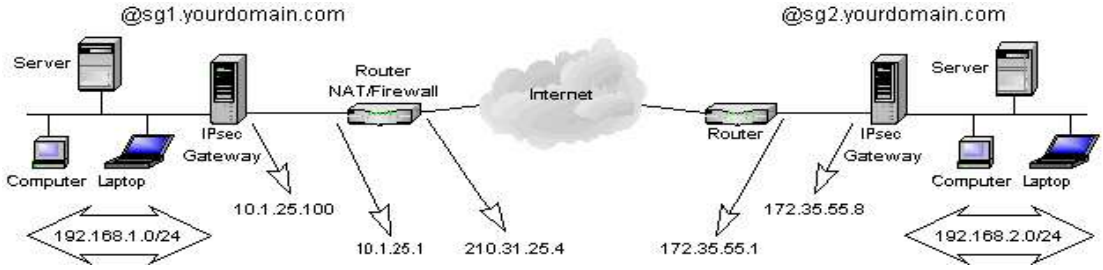
```
# Allow ipsec traffic, ESP and AH from and to the Internet
# INPUT Table
/sbin/iptables -A INPUT -p udp -m udp –sport 500 --dport 500 -j ACCEPT
/sbin/iptables -A INPUT -p 50 -j ACCEPT
/sbin/iptables -A INPUT -p 51 -j ACCEPT
# OUTPUT Table
/sbin/iptables -A OUTPUT -p udp -m udp –sport 500 --dport 500 -j ACCEPT
/sbin/iptables -A OUTPUT -p 50 -j ACCEPT
/sbin/iptables -A OUTPUT -p 51 -j ACCEPT
```

Disable any unused services (inetd.conf) and protect the remaining services called from inetd (hosts.allow and hosts.deny). Do not run daemons that should not resides on a security gate. I.e DNS, Sendmail and such services with a big security history. I often run SSH on those gateways, its the only backdoor if your tunnel stops working. If using SSH here horrifies you, use a good internal PPP access with callback support... just in case.

That's it for the simple subnet-to-subnet case. The tricky ones coming...

Subnet-to-Subnet configuration with a NATed gateway.

This is the first tricky configuration as one of the gateways is behind a router/firewall doing Network Address Translation (NAT). I use the term NAT here for a 1 to 1 translation. I.e one external IP address is converted to 1 internal IP address and vice-versa. This is not masquerading or PAT. This configuration can only work using ESP, because AH does not support modifications in the IP header. As of Freeswan 1.3, this configuration also only works with RSA authentication.



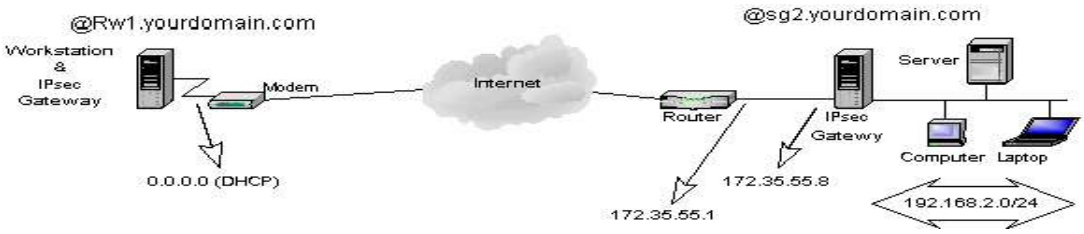
For example :
The difference is that the left gateway is no longer being exposed directly to the Internet.
The trick here is to have 2 different configuration files :

Left gateway's ipsec.conf	Right gateway's ipsec.conf
config setup interfaces=%defaultroute klipsdebug=none plutodebug=none plutoload=%search plutostart=%search **conn %default** keyingtries=0 **conn site1-site2** left=%defaulroute leftsubnet=192.168.1.0/24 leftnexthop= right=172.35.55.8 rightsubnet=192.168.2.0/24 rightnexthop=172.35.55.1 auto=start authby=rsasig leftid=@sg1.yourdomain.com rightid=@sg2.yourdomain.com leftrsasigkey=0x--left-public-key rightrsasigkey=0x--right-public-key	**config setup** interfaces=%defaultroute klipsdebug=none plutodebug=none plutoload=%search plutostart=%search **conn %default** keyingtries=0 **conn site1-site2** left=210.31.25.4 leftsubnet=192.168.1.0/24 leftnexthop=210.31.25.1 right=%defaultroute rightsubnet=192.168.2.0/24 rightnexthop= auto=start authby=rsasig leftid=@sg1.yourdomain.com rightid=@sg2.yourdomain.com leftrsasigkey=0x--left-public-key rightrsasigkey=0x--right-public-key

The right gateway only have to see left as its external NATed address.
This work because authentication is based on @sgx.yourdomain.com, not on a real IP address
(the @ says to KLIPS to not resolve that name to an IP. No need to put registered hosts names here).
Proceed the same way as the simple subnet-to-subnet configuration for your ipsec.secrets files.
The NAT/Firewall device was in fact a CISCO 1600 router in my setup. Don't forget that the router's access-lists (IOS) or firewall rules must let pass UDP 500 and ESP (50) for that setup to work.
Altough I did not have the chance to test it, this could work even if both gateways are NATed, if you adjust the configuration.

RoadWarrior Configuration : Freeswan-To-Freeswan.

This setup is usefull for moving users with laptop to connect to a central network using Ipsec. As most of the time they will be using a modem over an async connection to the ISP, my example describes that case.



The difficulties of this configuration origins from the fact that the initiator of the tunnel IP configuration (DHCP) is volatile and unknown from the right gateway. The configuration will have to reflect that situation and permit multiple users to connect at the same time.

Left gateway's ipsec.conf

```
config setup
 interfaces=%defaultroute
 klipsdebug=none
 plutodebug=none
 plutoload=%search
 plutostart=%search
conn %default
 keyingtries=1
conn Road-Central
 left=%defaultroute
 leftsubnet=
 leftnexthop=
 right=172.35.55.8
 rightsubnet=192.168.2.0/24
 rightnexthop=172.35.55.1
 auto=start
 authby=rsasig
 leftid=@rw1.yourdomain.com
 rightid=@sg2.yourdomain.com
 leftrsasigkey=0x--left-public-key
 rightrsasigkey=0x--right-public-key
```

Right gateway's ipsec.conf

```
config setup
 interfaces="ipsec0=eth0"
 klipsdebug=none
 plutodebug=none
 plutoload=%search
 plutostart=%search
conn %default
 keyingtries=1
conn Road-Central
 left=0.0.0.0
 leftsubnet=
 leftnexthop=
 right=172.35.55.8
 rightsubnet=192.168.2.0/24
 rightnexthop=172.35.55.1
 auto=add
 authby=rsasig
 leftid=@rw1.yourdomain.com
 rightid=@sg2.yourdomain.com
 leftrsasigkey=0x--left-public-key
 rightrsasigkey=0x--right-public-key
```

This setup works well with RSA authentication, and can work with PSK if you update ipsec.secrets automaticly on the RoadWarrior (had a script to do that). To bring the ipsec tunnel up as I connect to the internet, I usually remove Ipsec from start-up :

```
/sbin/chkconfig --del ipsec
```

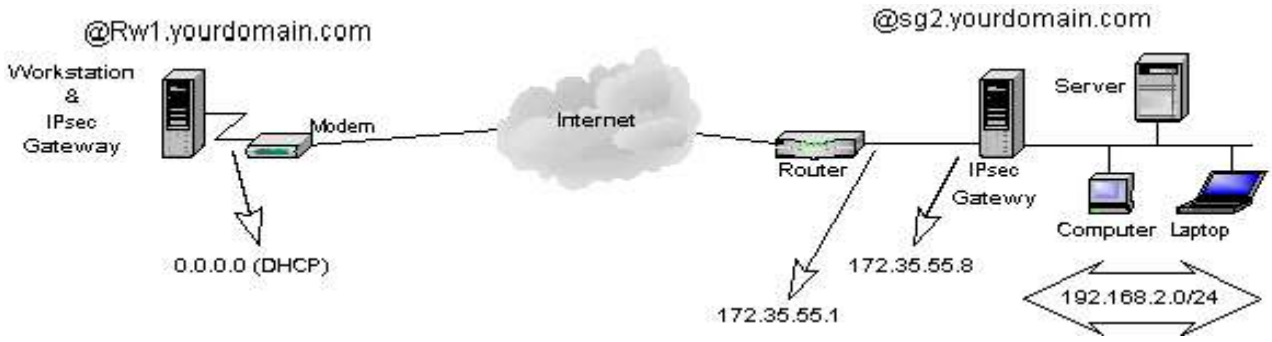
And bring the ipsec service up and down with my PPP interface through ip-up.local and ip.down.local in the /etc/ppp directory. Don't forget to make those executable.
You can NOT have both RSA and PSK RoadWarriors as of Freeswan 1.3.

Some tips about routing & RoadWarriors :

- IP forwarding must be activated (/etc/sysconfig/network) to permit routing to your LAN. Or you can also use the forwardcontrol parameter in ipsec.conf.
- If your gateway is also a masquerading gateway to the Internet, you should use the rightfirewall parameter in ipsec.conf and adapt the _updown script to ipchains (or anything used to control your firewall chains).
- The nodes on the protected subnet must use the ipsec gateway as their default gateway. Remember that the packets also needs to come back from your LAN to the RoadWarriors !

RoadWarrior configuration : NAI's PGPnet-to-Freeswan

Using Windows clients to access Freeswan is for me the key to integration of IPSec and the desktop. NAI's PGPnet is great for that task . It is pretty stable and transparent for the user. Remember that only the commercial copy of PGPnet can do tunnels as I will show in this example :



Here's the steps needed to setup PGPnet on the Win32 client (letfgateway) for that configuration (refer to NAI's documentation for installation) :

Steps	How to do it
Set up the adapter connected to the internet.	-Start - Program - PGP - Set Adapter -Select the network adapter connected to the internet
Launch the PGPnet configuration tool and set defaults options	- Start - Program - PGP – PGPnet - View – Options - General Panel : - Expert Mode - Allow communications with unconfigured hosts - Require valid authentication key - Cache passphrases between logins - *IKE Duration : **6h** - *IPsec : **6h** - Advanced panel : - Selected options : - Ciphers : **Tripple DES** - Hashes : **MD5** - Diffie-Hellman : **1024 and 1536** - Compression : **LZS and Deflate** - Make the IKE proposal : **Shared-Key - MD5 - 3DES -1024 bits on top of the list** - Make the IPSec proposal: **NONE - MD5-TrippleDES -NONE on top of the list** - Select Perfect Forward Secrecy = **1024 bits** - Press **OK**

Steps	How to do it
Create the connection's definition.	- In the Hosts panel, **ADD** - Name : Enter a name for the right gateway - Ipaddress:IP address visible to internet (172.35.55.8) - Select Secure Gateway - Set shared Paraphrase : enter you preshared key - Identity type : select IP address - Identity : enter **0.0.0.0** - Remote Authentication : select **Any** valid key - Press **Ok** - Select the newly created entry for the right gateway and click **ADD**, **YES** - Name : Enter a name for the central subnet - IP address : Enter its network IP address (192.168.2.0) - Select Insecure Subnet - Subnet Mask : enter its subnetmask (255.255.255.0) - Press **OK**, **YES**, **YES**
Test it	Ping 192.168.2.1

*I choosed to rekey faster that Freeswan to solve a common rekeying problem with Win32 Ipsec clients.

Ipsec.conf and ipsec.secrets on right gateway :

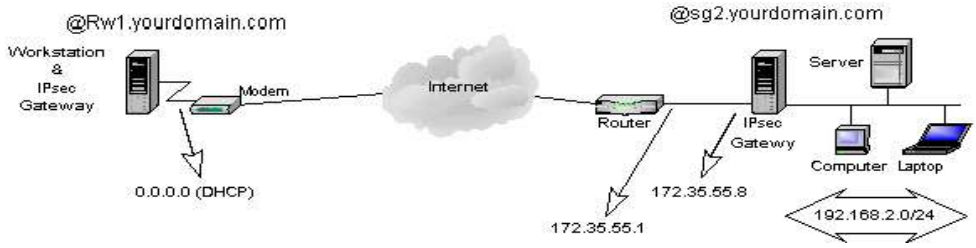
ipsec.conf	Ipsec.secrets
config setup interfaces="ipsec0=eth0" klipsdebug=none plutodebug=none plutoload=%search plutostart=%search **conn %default** keyingtries=1 **conn rw_pgp-site2** left=0.0.0.0 leftsubnet= leftnexthop= right=172.35.55.8 rightsubnet=192.168.2.0/24 rightnexthop=172.35.55.1 authby=secret auto=add	0.0.0.0 172.35.55.8 "mypresharedkey"

All your RoadWarriors will have to share the same PreShared Key.

See my tips about routing and RoadWarriors.

RoadWarrior Configuration : IRE's SafeNet/SoftPK-to- Freeswan

Topology: (Using the same example as for PGPnet)



IRE's SafeNet/SoftPK (3DES) is a much lighter software to do Ipsec tunnels but does not integrates PGP in emails and local encryption. If your only goal is to do some Ipsec tunnels on a Win32 desktop, SafeNet is the good choice as it is cheaper than NAI's PGPnet. Its configuration is not complex and works in most cases (exept with ADSL as it does not support a PPPoe interface). Lets do the Safenet's setup on the Win32 desktop (left gateway) for the configuration above :

Steps	How to do it
Launch the Safenet Security Editor and create a new Security Policy.	Start - Program - Safenet Soft-PK - Security Policy Editor File - New Connection Enter a name for the connection
Set the connection's parameters	Select the newly created connection : Connection Security : **Secure** Remote Party Identity and Adressing:***Select IP Subnet*** Enter the rightsubnet (192.168.2.0) Enter its netmask (255.255.255.0) Protocol : **ALL** Select Connect using Secure Gateway Tunnnel ID Type : ***Select IP Address*** Enter right gateway IP address (172.35.55.8) Expand properties of the connection (left pane): Select the Identity branch. Select Certificate = **none** ID Type : *Select IP Address* Port : ***Select ALL*** Local Network Interface : Select the interface used to reach internet. Click on Pre-Shared Key and enter your pre-shared key. Select the Security Policy branch Phase 1 negociation mode : **Main Mode** Select Enable Perfect Forward Secrecy PFS Key Group : **Diffie-Hellman Group 2** Select Enable Replay Protection. Expand properties of Security Policy (left pane) Expand the properties of Authentication Select the **Proposal 1** branch Authentication Method : **Pre-shared Key** Encrypt Alg : **Tripple DES** Hash Alg : **MD5** SA Life : **Seconds - 18000** Key Group : **Diffie-Hellman Group 1 (Safenet 1.x)** **Diffie-Hellman Group 2 (Safenet 2.x)** Expand the properties of Key Exchange Select the Proposal 1 branch Select Encapsulation Protocol (ESP) Encrypt Alg : **Tripple DES** Hash Alg : **MD5** SA Life : **Seconds - 18000** File - Save Changes
Test it.	Ping 192.168.2.1

All your RoadWarriors will have to share the same PreShared Key.
The right gateway's configuration (Freeswan) will be the same as the previous example.
As I said earlier, the troubleshooting is a lot easier checking the logs on the responder (right gateway). Most of the problems origins from configuration errors and typos like different netmasks entered on each side.See my tips about routing and RoadWarriors.

Using a central Ipsec gateway as a "tunnel hub"

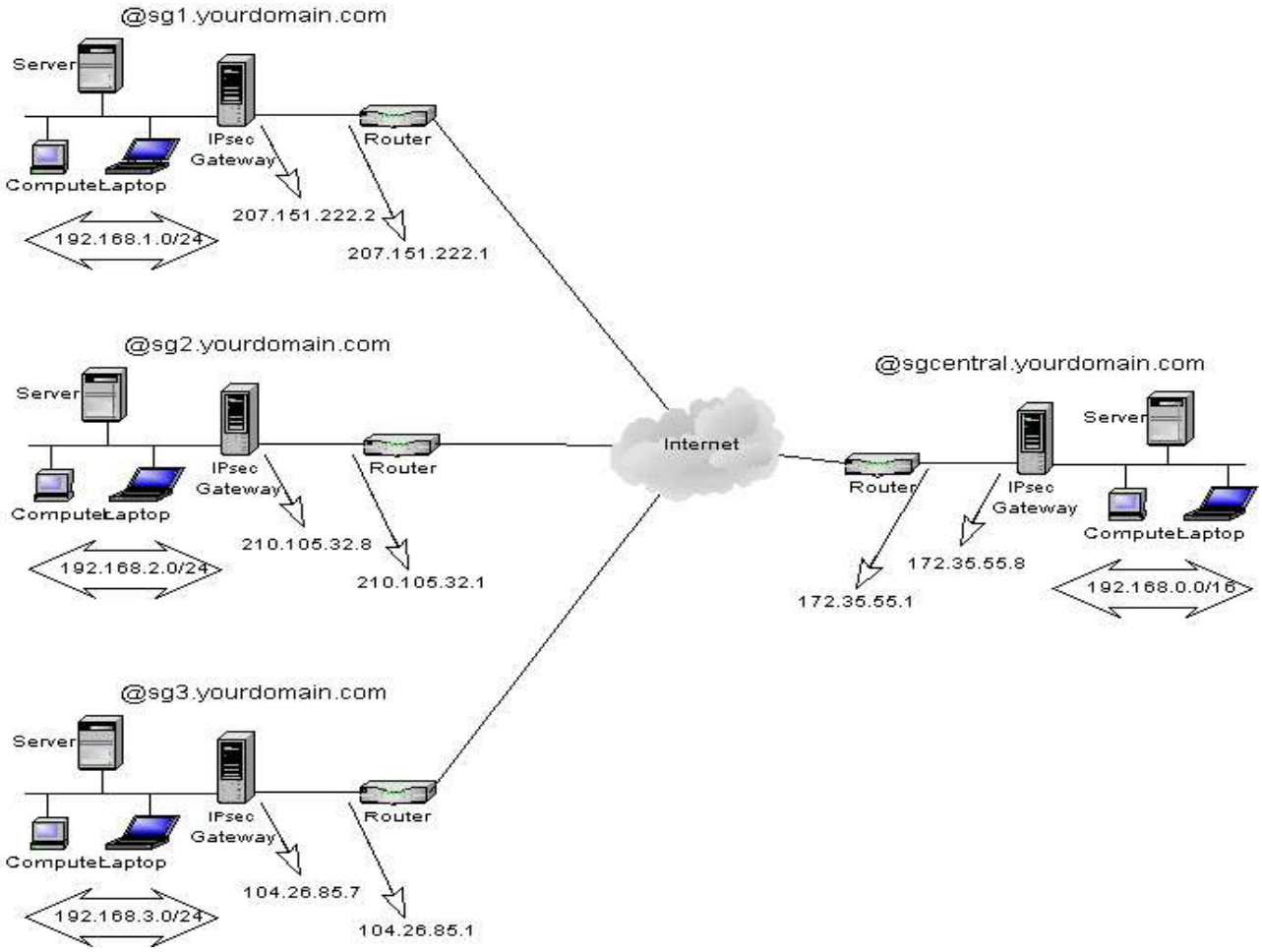
If you got multiples subnets connected to a central one, lets say a few remote locations connected to your headquarters, you might want to connect them all using Ipsec to permit communications between those remotes locations. You could create a mesh, i.e a connection from one locations to each others on all your gateways. If you got only 2 or 3 remote locations, this will work. But as you add more and more locations, it is gonna be a pain to administer as you will have to update all your ipsec gateways each time you add another tunnel. If you got 10 or more locations, this could become impossible to maintain (10*10 = 100 tunnels !).

Using one central location as a "tunnel hub" simplifies connecting all those subnets together, as only one tunnel is added for a new location. There are 3 drawbacks to that solution :

- If the central gateway dies, all your tunnels stops working.
- You will need more bandwith at central gateway's site, as all ipsec tunnels will be routed through it.
- You will need more processing power on that central gateway, as packets going between 2 remote locations will be encrypted twice.

That configuration might look like this :

Topology:



The remote locations are on the **left** side and the headquarters on the **right**.

We want the subnets 192.168.1.0/24, 192.168.2.0/24 and 192.168.3.0/24 to communicate with each others and with the central network 192.168.0.0/24.

The trick to make this work is on the graphic... all remote locations are on a C class network just as the central location. But each SA are for a longer netmask at the central site. Each remote location will create a tunnel from their subnet to the central subnet `192.168.0.0/16`. But the central subnet is a `192.168.0.0/24` network.When this is done, the central Ipsec gateway will have an eroute for each remote network. When someone on a remote location wants to talk to another remote location, lets say `192.168.1.8` to `192.168.2.16`, it thinks the destination is on the central network, but the central Ipsec gateway have a specific eroute for the `192.168.2.0/24` network, so it routes the packet through the good tunnel for that remote location.

Networks :

```
Central subnet :    192.168.0.0 255.255.255.0
Remote location 1 : 192.168.1.0 255.255.255.0
Remote location 2 : 192.168.2.0 255.255.255.0
Remote location 3 : 192.168.3.0 255.255.255.0
```

Ipsec tunnels :

```
192.168.1.0/24 --> 192.168.0.0/16 for remote location 1
192.168.2.0/24 --> 192.168.0.0/16 for remote location 2
192.168.3.0/24 --> 192.168.0.0/16 for remote location 3
```

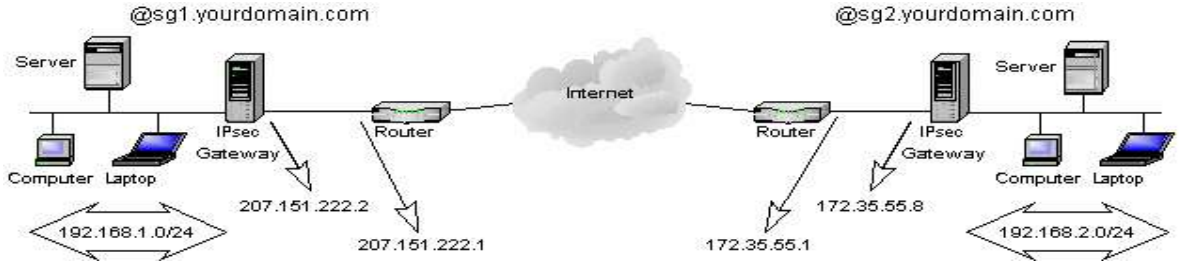
The configuration is the same as the Simple subnet-to-subnet case, just remember the network mask trick and everything should work.

You can control wich remote locations can talk to another by adjusting your forwarding rules with ipchains on the central ipsec gateway.

Subnet-to-Subnet configuration : Win2000-to-Freeswan (PSK).

This is the most complex configuration, because of microsoft's strange way to represent a connection and its parameters. This section is far from being complete, as I did not find any simple way to explain this in details yet without making you get crazy through all the menus and options, I will at first try to explain Microsoft's logic of a tunnel mode connection.

Topology:



We can use the simple subnet-to-subnet case as a model :

A connection in Microsoft's terminology is a Security Policy. An Ipsec Security Policy is configurable through an MMC console with the IP Security management Snap-in added.

When you create a new IP Security Policy (a connection) you must define rules for the traffic going throught it. You must define at least 2 rules for a tunnel mode connection to work :

- One rule for the traffic going from the leftsubnet to the rightsubnet.
- One rule for the traffic going from the rightsubnet to the leftsubnet.

A rule defines a :

- IP Filter. Describing the Ipsec subnets specifications (leftsubnet and rightsubnet).
- Filter Action. Describing the action to take when the filter applies (Require Security).
- Authentication Method. Desribing the authentication mode (PresharedKey).
- Endpoints . Describing the Ipsec gateway to reach (leftgateway or rightgateay).
- Ipsec/IKE proposals, rekeying settings and PFS are defined inside the Filter Action.

This logic can be very confusing, as you can go from properties menus inside properties menus.
Ouch !

The generics steps I recommend to make this setup work are :

1. Create a new MMC console and add the IP Security management Snap-in.
2. Create a new security policy.
3. Create a rule for the traffic from left-to-right.
4. Edit that rule and create a IP Filter that specifies the source address as the leftsubnet and the destination address as the rightsubnet.
5. Specify to Require Security on that IP Filter.
6. Edit that Filter Action, enable Perfect Forward Secrecy, and make the Ipsec proposal 3DES-MD5 on top of the list.
7. Specify the enpoint to be the right gateway's IP address.
8. Enter your preshared key.
9. Create a rule for the traffic from right-to-left.

10. Edit that rule and create a IP Filter that specifies the source address as the rightsubnet and the destination address as the leftsubnet.
11. Specify to Require Security on that IP Filter.

12. Edit that Filter Action, enable Perfect Forward Secrecy, and make the Ipsec proposal 3DES-MD5 on top of the list.
13. Specify the enpoint to be the letf gateway's IP address.
14. Enter your preshared key.
15. Save your changes
16. Select you Security Policy, and Assign it through the Action Menu.

Test it :

1. Ping a host on the other subnet, it should say at first "Negociating IP security".
2. Ping it again to see if it works.
3. Check the EventViewer for any errors.
4. Check the logs on the Freeswan gateway for any errors.
5. Run the ipsecmon program to see the connection status.

The longest SCREENSHOTS you've never seen to setup the previous example. 900 KB...you've been warned ;).

HINTS & LIMITATIONS :

- Configure the Freeswan gateway just like you would be creating a Freeswan-to-Freeswan tunnel using PSK for authentication.

- Be sure to install the Microsoft Encryption Pack for Win2000 before trying anything. It adds 3DES support to the OS, so don't even think something will work without it !

- If you modify a Security Policy after it has been assigned, you must restart the Ipsec service (or wait the default 240 minutes of the Ipsec service to read its configuration again)..

- A RoadWarrior configuration is difficult to do here. Because Win2k does not let you enter 0.0.0.0 as your endpoint. Altough a client to subnet configuration works, the Win2k client needs a static IP address.

I will need feedbacks to make this section clear, so don't hesitate to send me some comments !

IPsec with Kernel 2.6.x

The new native IPsec implementation for 2.6.x kernels greatly improves the security of Linux systems.
IPsec is an addition to IP protocol that allows authentication and encryption of IP datagrams. It is defined in detail in IETF RFCs 2401, RFC 2402, RFC 2406 and RFC 2407 (see Resources). IPsec can be used to secure a rather wide range of scenarios; one of its best-known usages is creating virtual private networks (VPNs). A VPN is a secure, private tunnel between two sub-networks using encrypted communication over the Internet.

FreeS/WAN has been the main IPsec implementation for Linux for a long time. Unfortunately, FreeS/WAN has never been integrated into the Linux kernel itself. Instead, the new native kernel IPsec implementation is based on the KAME project, a part of the UNIX/BSD family.

The USAGI project used the BSD code from the KAME project as a base for integrating IPsec into the Linux kernel. KAME's user-space tools, specifically setkey and Racoon, have been ported to Linux by the IPsec-tools Project (see Resources).

In this article, we implement a simple scenario of setting up a secure connection between two Linux systems, reblochon and gouda. We explain different IPsec user-land tools and how to use them to set up a secure connection between two systems. At the end, we present our benchmarks and discuss them.

What You Need to Install

To use IPsec, you need a kernel that supports IPsec protocols and user-land tools that allow key management and key exchange. These keys are used for different cryptographic algorithms.

For Linux kernels 2.5.47 and higher, IPsec support is a part of the kernel itself. However, this support is not enabled by default. If you have a Linux distribution such as Suse 9.1 or Fedora Core 2, it already comes with a 2.6 kernel and IPsec is enabled by default. If you use some other Linux distribution, for example, Fedora Core 1, you need to install a 2.6.x version of the kernel--the higher the better. This new kernel must be compiled with the following options enabled. Go to Device drivers -> Networking support -> Networking options to enable:

- PF_KEY sockets
- IP: AH transformation
- IP: ESP transformation
- IP: IPComp transformation
- IPsec user configuration interface

You also must include all the cryptographic algorithms you plan to use for your IPsec setup.

On the user-land side, the only thing you need is setkey and Racoon, which are part of the IPsec-tools Project (see Resources). The installation of these tools is straightforward: download the source code and proceed as usual with configure, make and make install commands. There even might be a precompiled package for your distribution of choice.

Setting Up a Secure Connection

You can use IPsec in two modes, transport or tunnel. Briefly, transport mode is used to secure host-to-host communications, and tunnel mode is used to tunnel securely site-to-site communications. In transport mode, a special header for ESP and AH is added to the normal IP header. In tunnel mode, the IP packet of transport mode with an ESP and AH header is encapsulated in a normal IP packet. That way, the ESP and AH header is not visible directly to routers that might discard a packet with unknown options.

IPsec can be configured in different ways. Here are three ways to configure an IPsec secure connection between two hosts:

- Shared Secret Keys: Start with a shared key on two nodes. Upon initialization of a secure connection between two nodes, this common shared secret is used for specified encryption or authentication algorithms. Using shared keys is the easiest way to configure but it also is less secure, as the shared secret most probably is contained in a configuration or script file on both machines. Also, if you do not change your keys often, it is possible that someone could capture enough packets to be able to retrieve the key.

- Pre-Shared Key: In this mode, you need to run Racoon. Its functionality is similar to the shared secret key. The only difference is Racoon uses the pre-shared key as a seed to negotiate a complete key and periodically change that key.

- X.509 Certificate: The most secure method to manage keys securely is to use the X.509 certificate. This solution requires access to a trusted certification authority (CA); otherwise, you need to set up your own CA. IPsec configuration in this case is not much more complicated, but interactions with a trusted certificate might be a problem.

In our simple scenario, we are more interested in discussing IPsec implementation performance rather than secure connection issues. So here we discuss the configuration of shared and pre-shared keys only.

A Simple Scenario

The following listing is used to illustrate how to set up a secure connection between two computers on different LANs. Below, we provide the script file of gouda that sets a secure connection between two systems. gouda is at IP address 192.168.0.1 and reblochon is at IP address 192.168.0.2.

```
 1: #!/usr/local/sbin/setkey -f
 2: flush;
 3: spdflush;
 4:
 5: # AH gouda to reblochon
 6: add 192.168.0.1 192.168.0.2 ah 1000
 7:         -A hmac-md5 "1234567890123456";
 8: add 192.168.0.2 192.168.0.1 ah 2000
 9:         -A hmac-md5 "1234567890123456";
10:
11: # ESP gouda to reblochon
12: add 192.168.0.1 192.168.0.2 esp 1001
13:         -E 3des-cbc "123456789012345678901234"
14:         -A hmac-sha1 "12345678901234567890";
15: add 192.168.0.2 192.168.0.1 esp 2001
16:         -E 3des-cbc "123456789012345678901234"
17:         -A hmac-sha1 "12345678901234567890";
18:
19: spdadd 192.168.0.1 192.168.0.2 any -P out IPsec
20:         esp/transport//require
21:         ah/transport//require;
22:
23: spdadd 192.168.0.2 192.168.0.1 any -P in IPsec
24:         esp/transport//require
25:         ah/transport//require;
```

For reblochon, we use the same script file with only the following differences on line 19:

```
19: spdadd 192.168.0.1 192.168.0.2 any -P in IPsec
```

and on line 23:

```
23: spdadd 192.168.0.2 192.168.0.1 any -P out IPsec
```

Note the in and out keywords are reversed.

In line 1, the setkey command is invoked. This program inserts or deletes IPsec rules in the kernel. In lines 2 and 3, we use the setkey command to clear all security associations (SA) and security policies (SP), because we want to begin from a clean state.

Before diving into more technical details, we need to become familiar with two basic concepts in IPsec protocol, security association (SA) and security policy (SP). An SA defines the security parameters, for example, the crypto algorithm to be used, to create a secure connection between two systems. An SP, on the other hand, is the security rule defining the security context to be used between the two systems. For example, an SP can specify that we need to use encryption between my desktop and a remote system on the Internet. An SA then is the effective secure connection created between my desktop and that system. Be aware that SAs are unidirectional.

In our scenario, we define two SPs between reblochon and gouda. An SP is defined as:

```
source | destination | on which kind of traffic to apply the policy (TCP, UDP,
port, any) |
the direction in/out | what to do (IPsec/discard/none) | (esp/ah) /
(transport/tunnel) /
(IP address of both ends of the tunnel) not required in transport mode /
require.
```

For example, these lines:

```
spdadd 192.168.0.1 192.168.0.2 any -P out IPsec
         esp/transport//require
         ah/transport//require;
```

declare a security policy stating that any packets coming from 192.168.0.1 and going to 192.168.0.2 should use IPsec on transport mode with ESP and AH functionality.

Now that you defined the policy between your systems, you need to define SAs in order to be able to achieve that policy. You need two SAs for communication, one from gouda to reblochon and one from reblochon to gouda. The two SAs do not need to use the same algorithm. In fact, unlike this example, for better security you should not use the same key for both SAs.

You define a SA as

```
source | destination | ah/esp | SPI (Security Policy Index) any number but
should be unique
| algorithm and associated secret key.
```

For example, these lines:

```
add 192.168.0.1 192.168.0.2 esp 1001
         -E 3des-cbc "123456789012123456789012"
         -A hmac-sha1 "12345678901234567890";
```

define that if you want to use ESP on a packet going from gouda to reblochon, you should use 3DES as the encryption algorithm with the quoted text as the key and SHA1 as the authentication algorithm.

Now, you finally can run the script on both nodes. You can check the status of different SAs established by using `setkey -D`. If you want to see existing policies on your system, you can use `setkey -DP`. At the end, to be sure that the traffic between two systems actually is encrypted, you can use Ethereal to monitor the traffic between two nodes. For example, in Figure 1, we show the traffic between two systems exchanging messages containing the "hello world" text. As you see, the message is encrypted between gouda and reblochon.

Figure 1: Ethereal Snapshot Showing Encrypted Communications

Automatic Keying

Now, you certainly can understand why this type of configuration is less secure. Keeping a system secure with secret keys in clear text in a configuration file is not practical. But don't worry, the people behind IPsec have thought about it and have come up with a protocol to negotiate keys and set up secure connections automatically. This functionality is implemented by Racoon. With Racoon, you do not need to specify any SAs; you need to specify only the SP. Racoon dynamically defines the SAs, and of course, you need to configure Racoon. The Racoon daemon runs on UDP port 500, which means that your firewall rules should not block this port on your system. We are not going into the details of setting up Racoon here, but refer to the HOWTO listed in Resources for more information.

Benchmark Results

We used Netio benchmarking software to test the new native IPsec implementation on a 2.6.7 kernel. We used two Pentium IV 2.2GHz machines with 512MB of memory. Netio measures the throughput of a network by way of TCP and UDP and different packet sizes. We established a secure connection using transport mode with the encryption algorithm 3DES (key size = 192 bits) and SHA1 for integrity checks. You can see the results in Table 1 for TCP and in Table 2 for UDP.

Table 1. Results from Netio TCP Benchmark

Packet Size	Bandwidth without IPsec	Bandwidth with IPsec
1KB	10905KB	5157KB
2KB	10832KB	5222KB
4KB	10827KB	5305 KB
8KB	10811KB	5263KB
16KB	10814KB	5345KB
32KB	10729KB	5374KB

Table 2. Results from Netio UDP Benchmark

Packet Size	Bandwidth without IPsec	Bandwidth with IPsec
1KB	11479KB	4806KB
2KB	11244KB	4320KB
4KB	11698KB	4985KB
8KB	11714KB	5116KB
16KB	11725KB	5152KB
32KB	11743KB	5271KB

We can see that IPsec diminishes by half the throughput of a TCP or UDP connection. Although the absolute bandwidth still is high--worst case is 4.3MB/sec--it is enough for most applications.

Conclusion

The new native IPsec implementation has a user-friendly interface to enable users to set up secure connections easily among different Linux systems. The results of our tests show that even though there is need for some improvements with regards to the stability of the implementation, the performance of the new native IPsec implementation makes it a good candidate for use by SOHOs as well as mid-sized enterprises. The new Linux IPsec implementation pushes Linux farther along the path to becoming a natural choice for many security needs of SOHOs and mid-sized companies.

Resources

IETF RFCs

IPsec-tools Project

Setting up Racoon

Netio Project

Vincent Roy and Makan Pourzandi work at the Open Systems Lab at Ericcson Canada.