

# Linux IPsec Gateways Using FreeS/Wan

*Peter Mueller* 2001-08-08

## Linux IPsec Gateways Using FreeS/Wan

by *Peter Mueller*

last updated Aug. 8, 2001

---

Open source projects are now refined enough that they provide us with the most configurable and reliable solutions of a vast array of refined products. Recently I was told to implement a VPN solution for my company and keep costs down. Since my company is one of the 'survivors' of the dot-com world, this definitely means no highly expensive commercial products (those of us left have a budget.) It also means we don't want to cut off our own head with an open-source solution that doesn't live up to our specifications.

### VPN Options

The absolute first thing administrators need to do after deciding they want to implement a VPN is to figure out what type of VPN they need. Essentially there are three main types:

1. remote user;
2. bridging of segregate networks; and,
3. a combination of the two.

Remote-user VPNs exist to allow users from a segregated network to securely access services in the home network. A classic example of this topology is the tireless sysadmin working from home at 3am (guzzling Mountain Dew) or perhaps the salesman at some remote location connecting over the Internet via a local dial-up. The open-source solution that most fits the bill here is [PoPTOP](#), which is essentially an open-source version of Microsoft's PPTP implementation that runs on Linux.

Bridging of networks hosts a wider array of choices, although there are only two commonly used open source variants - some kind of IPsec solution or SSH-PPpd trick. Compared to IPsec, SSH-PPpd is quite an inferior solution, both from a technical viewpoint and from PITAF (pain-in-the-ass-factor). I recommend that users avoid it, but if they insist on being stubborn, a good source of documentation is the [Linux Documtentation Project](#) who have a HOWTO and a MINI-HOWTO on the matter. Remember, you have been warned!

By far the most viable VPN solution is an IPsec variant. Not only is IPsec built in to IPV6, but also

all the major vendors and software consortiums are gearing their products towards this standard. It is probably a reasonable bet that in a few years users will be hard pressed to find any major VPN software that isn't utilizing some IPsec variant.

There's only one real choice here for IPsec and open-source on Linux and that is [FreeS/WAN](#). (If users prefer BSD, I hear the [OpenBSD](#) and [FreeBSD](#) variants, which are available at [the Kame Project](#) are pretty good!). FreeS/WAN is a well-established IPsec solution that has both a large support base (well into the thousands for implementations) and strong technical features. About the only drawback I can think of related to FreeS/WAN is their "road-warrior" (remote user) support. Currently the X.509 support is a bit of a pain to setup. Therefore if users require the "combination" setup mentioned above, I recommend utilizing both FreeS/WAN and PoPTOP software. Both can be placed on the same box since they utilize distinct ports.

Since PoPTOP is really quite simple to setup, the rest of this article will be devoted mainly to IPsec. IPsec very much appears to be the future of VPN software for a few years to come, so this choice appears to be appropriate. Now, on to a more in-depth view of IPsec.

## What is IPsec?

IPsec is perhaps best explained by this quotation from the FreeS/WAN homepage:

"IPSEC is **I**nternet **P**rotocol **SEC**urity. It uses strong cryptography to provide both authentication and encryption services. Authentication ensures that packets are from the right sender and have not been altered in transit. Encryption prevents unauthorized reading of packet contents. "These services allow users to build **secure tunnels through distrusted networks**. Everything passing through the distrusted net is encrypted by the IPSEC gateway machine and decrypted by the gateway at the other end. The result is **Virtual Private Network** or VPN. This is a network which is effectively private even though it includes machines at several different sites connected by the insecure Internet."

IPsec is actually an open standard that is being implemented by the vast majority of security vendors: it should continue to become more influential for years to come. FreeS/WAN is just one implementation, and vendors such as Cisco and Checkpoint are other (albeit substantially more expensive!) implementations. Users should expect various IPsec boxes to work together, although it might take a bit of work.

## IPsec and FreeS/Wan

FreeS/WAN works with the following IPsec modes: AH (authentication), ESP (encryption), AH (auth) + ESP (encrypt), ESP (auth and encryption - recommended), and double Authentication - AH + ESP. What all that means is that FreeS/WAN is flexible enough to work with most implementations, even though some modes like AH don't really make sense these days. (Why authenticate but not encrypt, especially with the relatively low cost of encryption?)

FreeS/WAN has one interesting feature that makes it distinct from most other IPsec implementations: DES encryption is unsupported. According to the FreeS/WAN home page, "DES is, unfortunately, a mandatory part of the IPSEC standard. Despite that, we will not implement DES. We believe it is more important to provide security than to comply with a standard which has been subverted into allowing weak algorithms."

Some of users might be thinking, "Hmm that's all well and good to run a secure VPN, but doesn't that kill performance?" Simply put, any tales of 'software encryption' not being able to perform are myths - even with 3DES. If users have any kind of decent processor they will be just fine. I've seen many reports on the mailing list indicating that a P60 does about 1.5mbit - 2mbit/sec with 3DES. Scaling conservatively, one could easily see a modern dual-processor computer surpassing 10mbit. Last time I looked the cost for a fast dual-processor box was right around \$2000. How much are those expensive firewall products again?

Let's talk about interoperability with other IPsec implementations for a bit. Assuming users have support for 3DES in their other IPsec gateway, getting most other implementations to talk to FreeS/WAN is not going to be too difficult. Most popular configurations have been done and have clear documentation readily available on the Internet. Examples of this type of documentation are available at FreeS/Wan's [Interoperation with other IPSEC implementations site](#), which contains HOWTO's for Cisco, Checkpoint, BSD(s), SSH sentinel, PGP, etc. I also highly recommend that users check their manufacturer's site or utilize the [Freeswan mailing list](#).

## Installing FreeS/Wan

Let's walk through a sample installation of FreeS/Wan.

FreeS/Wan currently compiles on 2.2 and 2.4 kernels. 2.0 kernels have been known to still compile, although personally I wouldn't count on it happening without it being a major pain (a la reverse bleeding-edge). Since I haven't personally installed a 2.0.x Freeswan, I cannot comment anymore with accuracy. If users are running a 2.2.x kernel, the Freeswan team recommends updating to the [latest kernel](#) (or from the [Linux Kernel Archive Mirror System](#)) due to some

security improvements released in 2.2.19. At the time of publication of this article, 2.4 kernels were, in my opinion, still too new to be put into production without a specific need.

Regardless of the user's Linux distribution, I recommend downloading kernel source directly from Linus and pals. It's been my experience that things almost always go right with the fresh kernel, whereas with a modified vendor kernel things are always a little iffy. All the usual reservations about upgrading the user's kernel apply: saving user.config somewhere, setting up LILO with backup kernels, etc.! Don't skimp out and proceed recklessly or you will definitely be sorry. I know from experience that it always seems to work out for the worst the one time users are not fully prepared. If this is your first time compiling the kernel, don't fret, as a [Linux Kernel HOWTO](#) is available.

In order for FreeS/Wan to install there needs to be a precompiled kernel present. Users shouldn't worry about setting up FreeS/WAN settings in the initial config, instead they should concentrate on properly configuring the kernel for their machine. FreeS/Wan is setup so that it will plug-in the settings it needs automatically for users.

There are only three steps required after compiling the kernel freshly - the 'make oldgo' installation option from the FreeS/Wan tar extraction directory, the 'make kinstall' script from the same directory, and a reboot to enable the new FreeS/Wan kernel. For more in-depth installation procedures, check out the [install guide](#) that is available from the FreeS/Wan site.

## Configuring Firewalls for Use With FreeS/Wan

Once FreeS/Wan is compiled properly, users will need to setup their firewall rules to allow IPsec communications to pass properly. In addition to their other rules, users should configure their firewall to allow UDP port 500 and protocols 50 & 51 bi-directionally between the gateways. Other required settings include:

```
Allow all traffic from the ipsec interface (usually ipsec0).
Allow forwarding of traffic from the ipsec interface from/to all
Disable rp_filter (on relevant interfaces).
Enable IP forwarding (on relevant interfaces).
Allow traffic between to-be-joined private networks.
Deny everything users don't explicitly need.
```

It is beyond the scope of this article to go into more details on how to secure the user's system; however, they should be aware that most default Linux distributions come with tons of

unnecessary services (especially for an IPsec gateway!) enabled and not much safeguard against Denial of Service attacks.

## Configuring FreeS/Wan

There are only two main configuration files to be concerned with, both located in /etc. One of those two files, ipsec.secrets, contains both public and private key exponents and a pre-shared key. The other file contains the relevant configuration information. This file is ipsec.conf.

The ipsec.secrets file contains a bunch of lines full of random characters. The keys contained in those lines are generated utilizing the 'ipsec rsasigkey' command. See the [IPSEC\\_RSASIGKEY](#) document on the FreeS/Wan site for specifics. An example command would be 'ipsec rsasigkey -verbose 2048 > mykey'. Since the installation program should have generated a key for users, this step is unnecessary unless users wish to change their key size.

If users examine their ipsec.secrets, they should see a line near the top that contains 'PSK:'. This is the pre-shared key I was talking about earlier. Users should change the default PSK-line to contain the IP addresses of the two gateways and a new random pre-shared key. To facilitate the creation of a random pre-shared key, utilize the 'ipsec ranbits' tool. Something like 'ipsec ranbits 60' will generate 60 random characters for users to use. Simply replace the default PSK with the newly generated random characters (I recommend cut & paste). Be sure to place this key inside the quotations. Both of the user's gateways must utilize this same PSK. (The rest of ipsec.secrets should be unique to each gateway.)

It goes without saying that all the contents of the ipsec.secrets file must be guarded with the utmost care. Do not leave this information lying around anywhere! Don't transmit it via unsecured means! If a malicious hacker obtains access to this file you're toast!

## Getting IPsec to Function

My example ipsec.conf file is going to be simple and to the point. There are a number of different methods that one could utilize in getting IPsec to function. However, there's only so much space in this article and I'm going to recommend the Perfect Forward Secrecy + RSASIG + ENCRYPT method. This method contains automatic key regeneration and is considered one of the much preferred choices. (Users may be familiar with the statement that "nobody ever got fired for buying Cisco", well the same applies here - nobody ever got fired for using RSASIG + PFS + ENCRYPT.)

Ipsec.conf can be broken up into two distinct parts - the 'basic config' and a connection-specific section. Let's go over the basic config first.

The only lines users really have to play around with in the basic config is the 'interfaces' line, which goes something like this:

```
interfaces="ipsec0=eth1" or
interfaces=%defaultroute
```

'interfaces=' is essentially asking what interface to bind the ipsec adapter to. The FreeS/Wan group recommends that users try the '%defaultroute' option first. My personal preference is to state the interface explicitly. Either way should be fine.

```
klipsdebug=none
plutodebug=none
```

Users can turn these two up when debugging usersr connection. After users have their VPN setup, they should be sure to turn these back to 'none' or usersr logs will fill quickly. The options I choose for my actual connections are listed below.

```
keyingtries=0      # be very persistent in reconnecting dropped connections
authby=rsasig     # authenticate via rsa public signature
compress=yes      # compress all data flowing between the links
left=left.public.ip.here      # public IP address of the left gateway
leftsubnet=192.168.0.0/24     # left network that we are tunneling
leftnexthop=your.next.hop.out # usually your default gateway
right=right.public.ip.here   # public IP address of the right gateway
rightsubnet=192.168.1.0/24   # right network that we are tunneling
rightnexthop=your.next.hop.out # usually your default gateway
```

It's important to note that left and right are arbitrary terms here. Users can choose either gateway to be either usersr left or right gateway. The gotcha here is that left and right need to be consistent on both gateways.

```
auto=add          # automatically startup this connection
leftid=somevalue  # see the FreeS/Wan Configuration page
rightid=somevalue # see the FreeS/Wan Configuration page
leftrsasigkey=0x? #use "ipsec showhostkey -left" to get value
rightrsasigkey=0x? #use "ipsec showhostkey -right" to get value
```

The left/right rsasigkey values are the public keys of the left and right gateways. That should be it!

At this point all the user should have to do is startup IPsec on both gateways and run the auto keying process on one of the gateways. For example:

```
/etc/rc.d/init.d/ipsec start  
ipsec auto -up connectionname
```

There's an important gotcha here. Essentially what we have done is to create a tunnel between two subnets, NOT a tunnel between the gateways. This means pinging one gateway to the other will not work or be useful. Instead, try these commands:

```
ping -i local.private.interface.address other.gateway.private.address  
traceroute -i public-eth -f 20 otherSGpublic
```

Users will still have to take care of routing in order to utilize their VPN. Everything else should be fully functional at this point. Expect uptime in the months or years.

## Upgrading FreeS/Wan

Recently I decided to upgrade my VPN to the latest FreeS/WAN release (1.91). This latest release fixed quite a few bugs from the 1.8 release I had been using, including bug fixes for both SMP and compression. Since those are features I wanted enabled on both of my gateways, I decided it was worth the risk on an update. Unfortunately there's an interesting catch with FreeS/Wan that I had forgotten about: when upgrading their FreeS/W versions, users need to go back to a 'pre' FreeS/Wan kernel. If users try to upgrade the kernel with FreeS/Wan already installed on it they will end up breaking their VPN and pulling their hair out for a while. I recommend not taking a risk on an upgrade unless it is for security improvements or feature enhancements that you need.

## A Note on Security

It seems silly to me that quite a few reputable sysadmin are willing to undertake the architecting of a VPN solution, but are unwilling to take the extra bit to secure the gateways themselves. Seriously folks, the encryption technology is good enough now that brute forcing just isn't likely to be a concern. A few far more likely points of failure are likely to include an improperly secured gateway, or a security hole through another machine, or failing to turn off a service. These are just some examples - there's definitely more!

Properly securing servers that must offer services to the Internet is definitely a continuous and laborious process (and very much beyond the scope of this article). If users are not familiar with

the intricacies of Linux, I highly recommend reading the excellent free (and online) "[Securing and optimizing RedHat Linux](#)".

[Privacy Statement](#)

Copyright 2006, SecurityFocus