Virtual Private Networks with Linux 2.4 and FreeS/WAN 2.01

Standard Tunnel

IPsec is the de facto standard for building virtual private networks (VPNs). FreeS/WAN integrates this security protocol with the Linux kernel and even provides support for opportunistic encryption. On the downside the installation and configuration are non-trivial tasks. We guide you through the steps needed to establish secure connections.

BY RALF SPENNEBERG


ADAC

If you need a secure connection between two geographically remote networks without investing in a leased line, a virtual private network is your only option. Internet Protocol Security (IPsec) has become a widely accepted standard, and it is now possible to create interoperable VPNs that use widely different hardware platforms and software solutions. The FreeS/WAN Project [1] implements IPsec with IPv4 and the Linux 2.2 or 2.4 kernels.

Although FreeS/WAN provides the whole range of functions required by a VPN, most users are not happy with this. Many programmers have written supplemental code in the form of patches for FreeS/WAN, see box "FreeS/WAN Patches". As FreeS/WAN is a kernel patch itself, we are really talking about patches for a patch – and installing this unusual combination is never easy.

In April of this year, a new 2.0 version of FreeS/WAN 2.0 was released. It is the first version to support opportunistic encryption by default (this is described later in this article). To allow for this, the developers added new policy groups to FreeS/WAN. In this article we will be focusing on the current 2.01 version and its major changes.

Installing FreeS/WAN 2.0

There are two or three ways to install FreeS/WAN. Some distributions include FreeS/WAN (for example SuSE, Mandrake, Debian 3.0); but these versions are often obsolete. Also, you have the choice between RPM packages and installing directly from the sources. To install from the sources you need the current kernel source, the current FreeS/WAN package [1], and the appropriate X.509 patch [2]. Before it will compile, FreeS/WAN expects you to have compiled the kernel once without FreeS/WAN and, more importantly, without any errors. Having done this, the admin can go on to patch and re-compile the kernel:

```
cd /usr/local/src
```

IPsec Basics

IPsec was originally designed for version 6 of the Internet Protocol. It guarantees the integrity, authenticity, and trustworthiness of information in data communications. As these characteristics are obviously desirable in IPv4 environments, IPsec was backported to support this protocol . FreeS/WAN is an implementation of this variant.

IPsec uses two protocols: AH (Authentication Header) and ESP (Encapsulating Security Payload). Both of these are independent IP protocols that use the numbers 50 and 51 respectively (see */etc/protocols*). The IKE protocol is used additionally to handle secure authentification and key exchanges. IKE (Internet Key Exchange) listens on UDP port 500.

THE AUTHOR

Ralf Spenneberg is a freelance Unix/Linux trainer and author. Last year saw the release of his first book: "Intrusion Detection Systems for Linux Servers". Ralf has also developed various training materials.

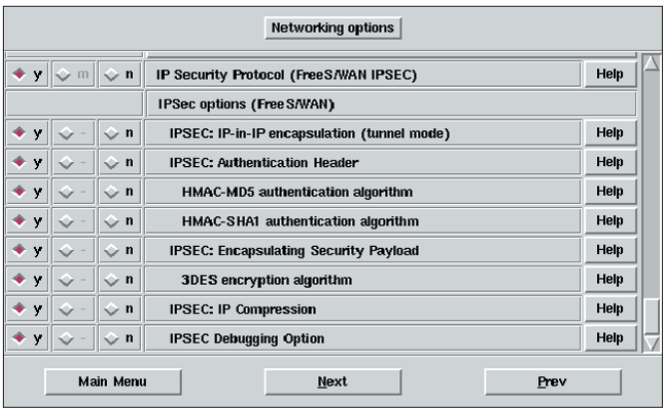

Figure 1: You can change the kernel configuration when you call *make xgo* for the FreeS/WAN sources

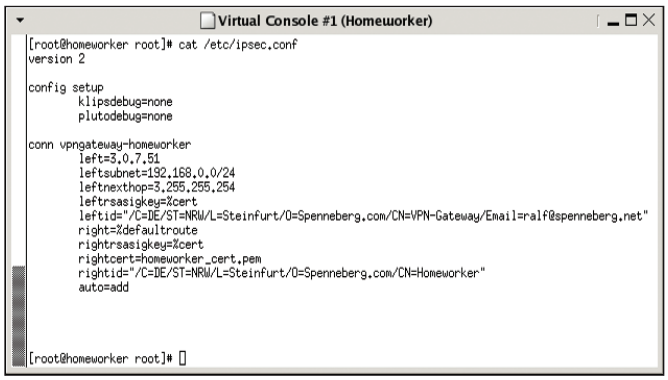

Figure 2: A road warrior uses a dynamic IP address to connect to the VPN gateway. The FreeS/WAN configuration for this user thus contains a *right=%defaultroute* entry, whereas the VPN gateway needs *right=%any*

```
tar xzf freeswan-2.01.tar.gz
tar xzf x509-1.4.2-freeswan➥
-2.01.tar.gz
cd freeswan-2.01
patch -p1 < ../x509-1.4.2-➥
freeswan-2.01/freeswan.diff > ➥
/dev/null
make oldgo KERNELSRC➥
=/usr/src/linux
```

If you need to change the configuration, you should use *make xgo* instead of *make oldgo* (see Figure 1). After successfully compiling, you can use the *make kinstall* to launch the installation – doing so will call *make install* for the kernel image, among other things.

Digging the Tunnel

After booting the new kernel, the admin can start configuring the VPN tunnel. Our lab scenario uses a VPN gateway, where the VPN client has a dynamic IP address that is unknown when configuring the tunnel (see Figure 1). Clients of this type are known as road warriors, the networking approach is referred to as client-to-site VPN.

To authenticate with FreeS/WAN, both systems need RSA keypairs comprising a private and public key each. The admin can generate these keys using *ipsec newhostkey* and store them in */etc/ipsec.secrets*. Each end of the connection will need the other's end public key. Transferring public keys is complex and causes confusion in the case of many VPN clients.

Certificates to Make it Simple

Certificates can facilitate authentication as both the client and the server send their public keys to the other end of the connection when opening a connection. The receiver then refers to the issuing CA's (Certificate Authority's) certificate to validate the incoming certificate. No matter how many road warriors you need to support, the VPN server only needs to refer to the CA certificate to authenticate each road warrior. Refer to the "Issuing Certificates" insert for more details.

Listing 1 shows the configuration file for the VPN gateway. The two VPN endpoints are called *left* and *right*; in our example the VPN gateway is on the left and the road warrior on the right (see Figure 2).

The entries for *left* and *right* are the IP addresses of the tunnel's endpoints. The address of the remote endpoint is unknown; the admin has entered the *%any* keyword to reflect this. If you do not know your own IP address, you have to type *%defaultroute* instead.

The Network at the End of the Tunnel

leftsubnet and *rightsubnet* refer to the networks at both ends that will be using the tunnel. In our example this is 192.168.0.0/24 on the left. *leftnexthop* and *rightnexthop* (if known) define the router used to build the tunnel. *leftid* and *rightid* specify the unique IDs of the VPN hosts, and must correspond to the subject of the certificates used by those hosts. The admin can extract these values from the certificates as follows:

```
openssl x509 -in Cert-File.pem ➥
  -subject -noout
```

OpenSSL responds with the distinguished name in X.500 format:

```
subject=/C=DE/ST=NRW➥
/L=Steinfurt/O=Spenneberg.com➥
/CN=Homeworker
```

The *leftcert* and *rightcert* parameters specify the file that contains the certificate for the host, if it is not transferred automatically on opening up the tunnel. To allow FreeS/WAN to use certificates, the admin needs to add *leftrsasigkey = %cert* and *rightrsasigkey = %cert*.

The configuration for the homeworker is similar, and is shown in Figure 3. In addition to the configuration file */etc/ipsec.conf*, both systems require a */etc/*

FreeS/WAN Patches

Without additional patches, FreeS/WAN only provides the basic functionality required of any IPsec implementation. The most important addition has to be a patch by Andreas Steffen [3], which supports authentification with X.509 certificates, among other things. The current 1.4.2 version also supports DHCP on IPsec connections, dynamic loading of certificate revocation lists (CRLs) via LDAP or HTTP, and storing private keys on Smartcards.

Some other patches worthy of mention are the ALG patch [10], which provides additional cryptographic algorithms, the Notify/Delete SA patch [11], and the NAT Traversal patch [11], which allows you to use IPsec despite NAT. As it is quite complex to deploy these patches, Ken Bancoft started to develop Super FreeS/WAN [2] a while back. This is a version of FreeS/WAN complete with all the patches. But unfortunately, Super FreeS/WAN for FreeS/WAN 2.0 is still in the beta phase.

ipsec.secrets file. This file contains a reference to the private keys and the passphrase required to use these keys. The following line shows the entry for the VPN gateway:

```
RSA vpngateway_key.pem➥
 "Passphrase"
```

The administrator can now launch FreeS/WAN by typing *ipsec setup start* on both machines. Typing *ipsec auto --up vpngateway-homeworker* on the road-warrior machine should start up the tunnel. If this works, the *auto = start* entry should allow FreeS/WAN to open up the tunnel automatically at startup.

Opportunistic Encryption

FreeS/WAN has provided experimental support for opportunistic encryption, OE, for quite a while now. Version 2.0 saw the end of the experimental status and the official introduction of this feature. Opportunistic encryption allows the administrator to open up a tunnel automatically, simply by installing FreeS/WAN. There is no need to configure FreeS/WAN, as the required data can be derived from the DNS server's zone files.

Version 2.01 modifies this approach, but sacrifices backward compatibility in doing so. Where FreeS/WAN 2.0 uses

Listing 1: VPN Gateway

```
01 version 2
02 config setup
03     interfaces="ipsec0=eth0"
04     klipsdebug=none
05     plutodebug=none
06
07 conn vpngateway-homeworker
08     left=3.0.7.51
09     leftsubnet=192.168.0.0/24
10     leftnexthop=3.255.255.254
11     leftrsasigkey=%cert
12
leftcert=vpngateway_cert.pem
13
leftid="/C=DE/ST=NRW/L=Steinfurt/
O=Spenneberg.com/CN=VPN-
Gateway/Email=ralf@spenneberg.net
"
14     right=%any
15     rightrsasigkey=%cert
16     auto=add
```

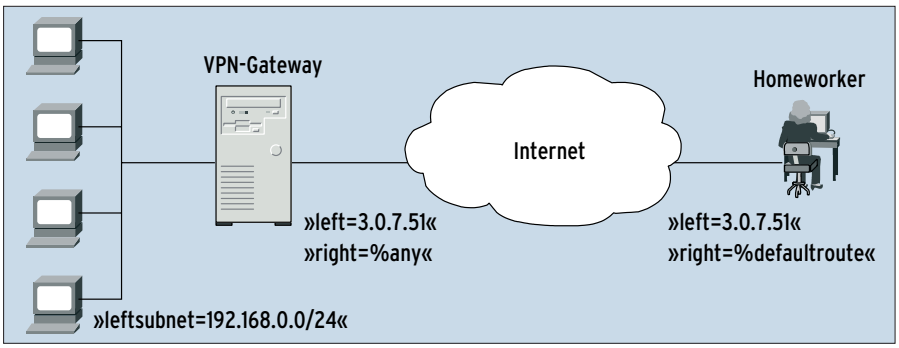

Figure 3: Homeworker configuration

TXT and KEY resource records, FreeS/WAN 2.01 only uses TXT RRs. Opportunistic encryption uses only RSA keys, but not certificates to authenticate. As the keys are distributed via DNS, the authentication will be as secure as your DNS environment – and that means quite insecure in the case of traditional DNS. Without secure DNS, OE can only protect you from passive attacks, but not from man-in-the-middle attacks [12].

OE distinguishes between the initiator and the responder. The initiator is required to register her public key with a domain as a TXT-RR, choosing an appropriate DNS name to do so – for example *initiator.example.com* – and then call *ipsec showhostkey --txt @initiator.example.com*. The initiator then stores the output from this command as a DNS TXT RR for her own address, or more typically would ask the DNS server administrator to do so.

In contrast to this, the responder publishes her key on the DNS server as a reverse DNS TXT record, using the *ipsec*

Issuing Certificates

FreeS/WAN with the X.509 patch can use RSA keys or certificates for authentication purposes. The latter approach is more complex if you simply want to test FreeS/WAN, but it does facilitate key management in larger networks. The first requirement is a CA (Certificate Authority). The OpenSSL package comprises a simple tool – OpenSSL is available with most major Linux distributions. A new CA can be set up quite easily:

```
/usr/share/ssl/misc/CA -newca
```

The tool prompts the admin to enter a password to protect the CA's private key. The next step is to increase the CA's lifetime:

```
openssl -in demoCA/cacert.pem ➥
  -out demoCA/cacert2.pem -days➥
3650   -signkey demoCA/private/➥
cakey.pem
mv demoCA/cacert2.pem demoCA/➥
cacert.pem
```

The CA can now process and sign certificate requests. You also need to supply a passphrase to protect the private key when entering a certificate request (*CA -newreq*).

Requesting a Certificate

To sign the request (option *-sign*) and thus issue the certificate, the admin enters the CA passphrase, thus issuing your private key.

```
/usr/share/ssl/misc/CA -newreq
/usr/share/ssl/misc/CA -sign
```

It makes sense to rename keys and certificates to prevent you confusing them:

```
mv newreq.pem vpngateway_key.pem
mv newcert.pem vpngateway_➥
cert.pem
```

Finally, the admin needs to copy these file to the appropriate machines:

```
cp demoCA/cacert.pem /etc/➥
ipsec.d/cacerts
cp vpngateway_key.pem /etc/➥
ipsec.d/private
cp vpngateway_cert.pem /etc/➥
ipsec.d/certs
```

Each machine on the VPN needs a Certificate

Each machine that needs access to the VPN also needs a certificate. After issuing certificates, make sure that you store the CA's private key in a safe place.

All the machines now have their own keys and the CA certificate. When opening up the tunnel, each endpoint transfers its own certificate to the other end. The receiving endpoint can now validate the certificate against the CA certificate before authenticating its counterpart at the other end of the tunnel.

showhostkey --txt IP address to create a text-based version of the key.

Full Opportunism

A scenario where all the machines are capable of acting as initiators and responders, is referred to as full opportunism. No further configuration steps are required in this case. FreeS/WAN acquires the required keys when communicating with a remote machine, and if possible, automatically sets up an encrypted connection.

The new policy groups and the *packetdefault* connection type are used to set up opportunistic encryption automatically. FreeS/WAN defines five groups: *private*, *private-or-clear*, *clear-or-private*, *clear* and *block*. Groups are configured as IP addresses or IP address blocks added to the appropriate */etc/ipsec.d/policies/group* files by the admin.

FreeS/WAN uses only encrypted communication for the *private* group. *private-or-clear* attempts to set up encrypted communication first, and falls back to a non-encrypted state if this fails. The *clear-or-private* group does exactly the opposite. FreeS/WAN uses only cleartext communication with the *clear* group, and the *block* prevents connections from being established.

Internal Network Gateway

In addition to these groups, FreeS/WAN 2.0 also enables the *packetdefault* connection type, which configures the host as an OE gateway for any machines behind it.

The integrated opportunistic encryption feature can cause problems if it is not used. For example, it is impossible to route non-encrypted connections past the tunnel, or delays occur at least; additionally FreeS/WAN will log error messages in your logfiles. If you do not need opportunistic encryption, you should disable the internal OE connections. To do so, simply add the lines in Listing 2 to your configuration file, */etc/ipsec.conf*.

Listing 2: Disabling OE

```
01 conn block
02     auto=ignore
03 conn private
04     auto=ignore
05 conn private-or-clear
06     auto=ignore
07 conn clear-or-private
08     auto=ignore
09 conn clear
10     auto=ignore
11 conn packetdefault
12     auto=ignore
```

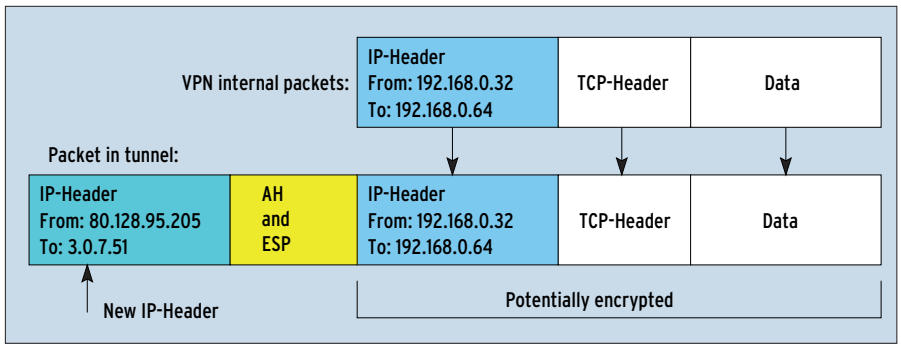

Figure 4: In IPsec tunnel mode, a new IP header is added to the packet: While the client uses the 192.168.0.32 IP inside the VPN (top), the external network sees only the public address, 80.128.95.205, (bottom)

When a road warrior opens up a connection via a VPN gateway, the connection is typically routed across the Internet. To do so, the user supplies the temporary IP address supplied by her Internet service provider. This is the IP address used for communicating across the tunnel.

DHCP via IPsec

Administrators often need to assign their road warriors internal IP addresses to support tunneled communications. The encrypted packets are sent to the official IP address, but inside the tunnel the road warrior uses a different IP, assigned by DHCP (see Figure 4). The X.509 patch (see the "FreeS/WAN Patches" box) adds a facility to FreeS/WAN to allow IP addresses of this kind. To leverage this facility you also need to install Mario Strasser's DHCP Relay [3] on your VPN gateway. DHCP Relay accepts DHCP requests from your clients and forwards them to your enterprise's internal DHCP server.

Unfortunately, FreeS/WAN itself does not provide DHCP-over-IPsec client functionality, although it supports the server side. Many clients support this approach, such as the commercial IPsec client alternative, SSH Sentinel [6], see Figure 5.

Automatic CRL Subscriptions

If a VPN uses X.509 for authentication purposes, problems can occur if a laptop

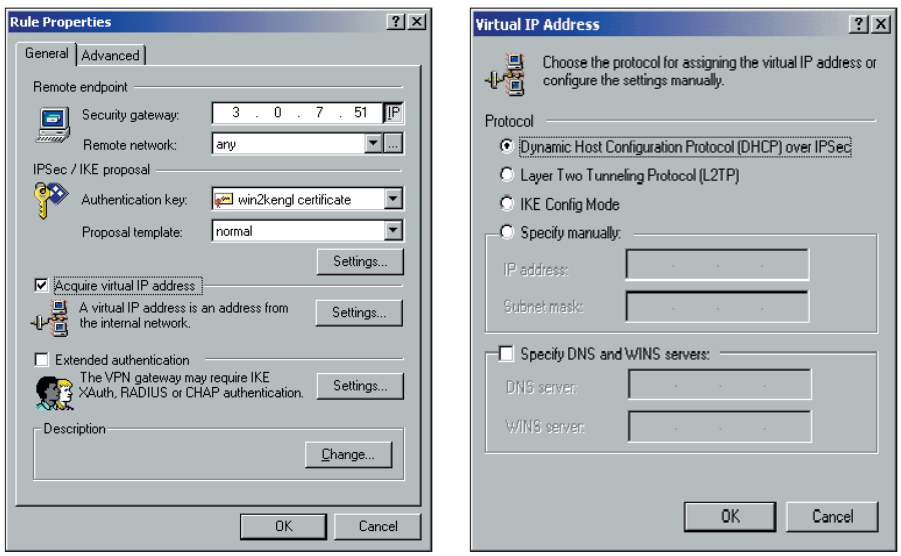

Figure 5: The SSH Sentinel (on Windows) supports DHCP-over-IPsec. *Rule Properties* allow the user to specify a virtual IP address for use inside the tunnel (left); the settings (on the right) are used to specify that this IP address should be assigned by DHCP

Listing 3: Resetting the Defaults

```
01 config setup
02 interfaces=%none # new default is %defaultroute
03 plutoload=%none # new default is %search
04 plutostart=%none # new default is %search
05
06 conn %default
07 uniqueids=no # new default is yes
08 keyingtries=3 # new default is %forever
09 disablearrivalcheck=yes # new default is no
10 authby=secret # new default is rsasig
11 leftrsasigkey=%none # new default %dnsondemand
12 rightrsasigkey=%none # new default %dnsondeman
```

is stolen. The certificate on the laptop is no longer trustworthy, although it is still valid and the VPN gateway will not reject the certificate, as it has not yet expired.

Certificate authorities use certificate revocation lists, or CRLs, to revoke compromised certificates before they expire. If a certificate has been compromised, the certificate authority can use the following commands to revoke it:

```
openssl ca -revoke ➥
certificate.pem
openssl ca -gencrl -out ca.crl
```

In the past, VPN administrators had no alternative but to distribute CRLs manually to machines that required them. The X.509 patch supports automatic downloading of CRLs via FTP, HTTP, or LDAP. To allow this, the CRL distribution source must be specified in the CA's X.509 certificate. In addition, the admin has to install Curl (for FTP and HTTP) or enable LDAP in the *programs/pluto/Makefile* when compiling FreeS/WAN.

Smartcard Support

In the past, FreeS/WAN users had to store their private keys on their hard disks. Users who specified a passphrase to protect their keys had to store the passphrase in the clear in */etc/ipsec.secrets* to allow FreeS/WAN to open up the tunnel.

In other words, there was no real way to protect the key. Version 1.4.0 or later with the X.509 patch allows you to store your key on a Smartcard or USB crypto token. The Smartcard protects the key adequately, as it is impossible to extract the key from the card. And the card handles the required signature operations itself. The functionality provided by the X.509 patch is based on the Open SC library [7] and on the PC/SC Lite package [8].

Both are available as RPM packages for Red Hat Linux from the author's homepage [9].

To support Smartcards, the admin needs to modify the *SMARTCARD = 1* entry in *programs/pluto/Makefile* before compiling, and to specify the Smartcard (instead of a file containing the certificate) in the */etc/ipsec.conf* configuration file.

```
leftcert=%smartcard
```

Upgrading

If you want to upgrade from FreeS/WAN 1.9 to version 2, you should start by saving your current configuration. After installing FreeS/WAN, the next step is to modify the */etc/ipsec.conf* configuration file; the line with *version 2* is particularly important. If you do not require opportunistic encryption for your VPN, you will want to use the commands in Listing 2 to disable the integrated connections.

Some issues may occur due to the FreeS/WAN team changing some defaults. To provide complete compatibility, you might like to use the lines in Listing 3; you will need to insert them at the start of your configuration file.

Pluto and the Linux 2.6 Kernel

Ever since the developer kernel 2.5.45, the Linux kernel has had a new IPsec implementation that was developed by Dave Miller and Alexey Kuznetsov. Their work is heavily based on the USAGI *http://www.linux-ipv6.org*, KAME *http://www.kame.net*, and WIDE *http://www.wide.ad.jp* projects. David Miller also backported this implementation [5] to the 2.4.21 kernel, and it is included with the current Red Hat Linux beta distribution, Severn.

Herbert Xu is responsible for a port of the FreeS/WAN IKE daemon, Pluto [4], to the new IPsec implementation. This allows admins to continue using their FreeS/WAN configurations without patching the kernel, as it merely requires userspace tools.

As the Smartcard provides PIN protection for the key, FreeS/WAN needs the PIN to open up the tunnel. The user can either store the PIN in the clear in the */etc/ipsec.secrets* file, or enter the PIN when prompted by FreeS/WAN. The two variants look like this:

```
* PIN %smartcard "12345678"
* PIN %smartcard %prompt
```

Of course, prompting the user for the PIN is more secure as it prevents a stolen server from developing into a major security headache. Having said that, this does prevent FreeS/WAN from starting up automatically.

Conclusion

FreeS/WAN with the latest additions is an extremely powerful IPsec implementation and now provides enhanced functions previously only offered by commercial manufacturers. Automatic CRL updates, and Smartcard support in particular, allow the admin to deploy FreeS/WAN in professional environments. ■

INFO

[1] Official FreeS/WAN homepage: *http://www.freeswan.org*

[2] Unofficial FreeS/WAN homepage: *http://www.freeswan.ca*

[3] X.509 patch and DHCP Relay: *http://www.strongsec.com/freeswan/*

[4] Pluto for Linux Kernel 2.6: *http://gondor.apana.org.au/~herbert/freeswan/*

[5] IPsec backport: *ftp://ftp.kernel.org/pub/linux/kernel/people/davem/IPSEC/*

[6] SSH Sentinel: *http://www.ssh.com/products/security/sentinel/*

[7] Open SC: *http://www.opensc.org*

[8] PC/SC Lite: *http://www.linuxnet.com/middle.html*

[9] FreeS/WAN RPM packages for Red Hat: *http://www.spenneberg.com*

[10] ALG-Patch: *http://www.irrigacion.gov.ar/juanjo/ipsec/*

[11] NAT Traversal Patch: *http://open-source.arkoon.net*

[12] Known OE issues: *http://www.freeswan.org/freeswan_trees/freeswan-2.01/doc/opportunism.known-issues*