Linux VPN Masquerade HOWTO

Table of Contents

Linux VPN Masquerade HOWTO.....1
John D. Hardin <jhardin@wolfenet.com>.....1
1. Introduction.....1
2. Background Knowledge.....1
3. Configuring the Linux firewall.....1
4. Configuring the VPN client.....2
5. Troubleshooting.....2
6. IPsec masquerade technical notes and special security considerations.....2
1. Introduction.....2
1.1 Introduction.....2
1.2 Feedback, Credits & Resources.....3
1.3 Copyright & Disclaimer.....4
2. Background Knowledge.....5
2.1 What is a VPN?.....5
2.2 What is IPsec?.....5
2.3 What is PPTP?.....6
2.4 What is FWZ?.....6
2.5 Why masquerade a VPN client?.....6
2.6 Can several clients on my local network use IPsec simultaneously?.....7
2.7 Can several clients on my local network use PPTP simultaneously?.....7
2.8 Can I access the remote network from my entire local network?.....8
2.9 Why masquerade the VPN server?.....8
2.10 Why patch the Linux kernel?.....8
2.11 Current Status.....9
3. Configuring the Linux firewall.....9
3.1 Example network.....9
3.2 Determining what needs to be done on the firewall.....10
3.3 Patching and configuring the 2.0.x kernel for VPN Masquerade support.....10
3.4 Patching and configuring the 2.2.x kernel for VPN Masquerade support.....13
3.5 ipfwadm setup for a Private−IP VPN Client or Server.....15
3.6 ipchains setup for a Private−IP VPN Client or Server.....17
3.7 A note about dynamic IP addressing.....18
3.8 Additional setup for a Private−IP VPN Server.....18
3.9 ipfwadm setup for a Registered−IP VPN Server.....20
3.10 ipfwadm setup for a Registered−IP VPN Client.....21
3.11 ipchains setup for a Registered−IP VPN Server.....21
3.12 ipchains setup for a Registered−IP VPN Client.....22
3.13 VPN Masq and LRP.....22
3.14 VPN Masq on a system running FreeS/WAN or PoPToP.....23
4. Configuring the VPN client.....23
4.1 Configuring a MS W'95 client.....23
4.2 Configuring a MS W'98 client.....24
4.3 Configuring a MS W'ME client.....24
4.4 Configuring a MS NT client.....24
4.5 Configuring for network−to−network routing.....25
4.6 Masquerading Checkpoint SecuRemote−based VPNs.....25
5. Troubleshooting.....26

Table of Contents

5.1 Testing....26
5.2 Possible problems....26
5.3 Troubleshooting....28
5.4 MS PPTP Clients and domain−name issues....29
5.5 MS PPTP Clients and Novell IPX....29
5.6 MS network password issues....30
5.7 If your IPsec session always dies after a certain amount of time....30
5.8 If VPN masquerade fails to work after you reboot....31
5.9 If your second PPTP session kills your first session....31
6. IPsec masquerade technical notes and special security considerations....31
6.1 Limitations and weaknesses of IPsec masquerade....31
6.2 Proper routing of inbound encrypted traffic....33

Linux VPN Masquerade HOWTO

John D. Hardin `<jhardin@wolfenet.com>`

$Revision: 2.19 $ $Date: 2000–10–22 12:07:43–07 $

How to configure a Linux firewall to masquerade IPsec− and PPTP−based Virtual Private Network traffic, allowing you to establish a VPN connection without losing the security and flexibility of your Linux firewall's internet connection and allowing you to make available a VPN server that does not have a registered internet IP address. Brief information on configuring the VPN client and server is also given.

---

1. Introduction

- 1.1 Introduction
- 1.2 Feedback, Credits & Resources
- 1.3 Copyright & Disclaimer

2. Background Knowledge

- 2.1 What is a VPN?
- 2.2 What is IPsec?
- 2.3 What is PPTP?
- 2.4 What is FWZ?
- 2.5 Why masquerade a VPN client?
- 2.6 Can several clients on my local network use IPsec simultaneously?
- 2.7 Can several clients on my local network use PPTP simultaneously?
- 2.8 Can I access the remote network from my entire local network?
- 2.9 Why masquerade the VPN server?
- 2.10 Why patch the Linux kernel?
- 2.11 Current Status

3. Configuring the Linux firewall

- 3.1 Example network
- 3.2 Determining what needs to be done on the firewall
- 3.3 Patching and configuring the 2.0.x kernel for VPN Masquerade support
- 3.4 Patching and configuring the 2.2.x kernel for VPN Masquerade support
- 3.5 ipfwadm setup for a Private−IP VPN Client or Server
- 3.6 ipchains setup for a Private−IP VPN Client or Server
- 3.7 A note about dynamic IP addressing
- 3.8 Additional setup for a Private−IP VPN Server
- 3.9 ipfwadm setup for a Registered−IP VPN Server
- 3.10 ipfwadm setup for a Registered−IP VPN Client
- 3.11 ipchains setup for a Registered−IP VPN Server
- 3.12 ipchains setup for a Registered−IP VPN Client

- 3.13 VPN Masq and LRP
- 3.14 VPN Masq on a system running FreeS/WAN or PoPToP

4. Configuring the VPN client

- 4.1 Configuring a MS W'95 client
- 4.2 Configuring a MS W'98 client
- 4.3 Configuring a MS W'ME client
- 4.4 Configuring a MS NT client
- 4.5 Configuring for network−to−network routing
- 4.6 Masquerading Checkpoint SecuRemote−based VPNs

5. Troubleshooting

- 5.1 Testing
- 5.2 Possible problems
- 5.3 Troubleshooting
- 5.4 MS PPTP Clients and domain−name issues
- 5.5 MS PPTP Clients and Novell IPX
- 5.6 MS network password issues
- 5.7 If your IPsec session always dies after a certain amount of time
- 5.8 If VPN masquerade fails to work after you reboot
- 5.9 If your second PPTP session kills your first session

6. IPsec masquerade technical notes and special security considerations

- 6.1 Limitations and weaknesses of IPsec masquerade
- 6.2 Proper routing of inbound encrypted traffic

---

1. Introduction

1.1 Introduction

This document describes how to configure masquerading of IPsec and PPTP VPN traffic. SSH−based VPNs (such as that sold by F−Secure and outlined in the VPN mini−HOWTO) are based on standard TCP traffic and do not need any special kernel modifications.

VPN Masquerade allows you to establish one or more IPsec and/or PPTP sessions to internet−accessible VPN servers via your Linux internet firewall without forcing you to connect to your ISP directly from the VPN client system − thus retaining all of the benefits of your Linux internet firewall. It also allows you to set up a VPN server with a Private Network IP address (as described in RFC1918) behind a masquerading Linux firewall, permitting you to provide relatively secure access to a private network via only one registered IP address − even if that IP address represents a dynamic dial−up link.

It is strongly recommended that you understand, configure and test regular IP Masquerading before you attempt to set up VPN masquerading. Please see the IP Masquerade HOWTO and the IP Masquerade Resource page at http://ipmasq.cjb.net/ before proceeding. Planning and setting up your VPN and firewall is beyond the scope of this document. Here are some resources:

- http://www.linux.org/help/ldp/howto/Firewall−HOWTO.html
- http://hal2000.hal.vein.hu/~mag/linux−security/VPN−HOWTO.html

The patch for the 2.0.x−series kernels works well on Linux kernel version 2.0.36, has been incorporated into the 2.0.37 release, may work on versions earlier than 2.0.36, and should work on Linux kernels up to about version 2.1.102. The IP masquerade code in the kernel was restructured at about version 2.1.103, requiring a different patch for the 2.1.105+ and 2.2.x series of kernels. A patch is available for kernels from 2.2.5 to 2.2.17, and it may work on earlier kernels.

1.2 Feedback, Credits & Resources

The home page for the Linux VPN Masquerade kernel patches is http://www.impsec.org/linux/masquerade/ip_masq_vpn.html

Please feel free to send any feedback or comments regarding this document to me at <jhardin@wolfenet.com>. The current version can be found at:

- HTML: ftp://ftp.rubyriver.com/pub/jhardin/masquerade/VPN−howto/VPN−Masquerade.html
- Postscript: ftp://ftp.rubyriver.com/pub/jhardin/masquerade/VPN−howto/VPN−Masquerade.ps.gz
- SGML source: ftp://ftp.rubyriver.com/pub/jhardin/masquerade/VPN−Masquerade.sgml

If you are working with a kernel whose version number is higher than any mentioned in this document, *please* see if there is an updated version of the HOWTO at the above site before contacting me directly.

It can also be found via the Linux Documentation Project's HOWTO repository and in the `/usr/doc/HOWTO/` directory on your nearest Linux system. These copies are not directly updated by me, so they may be somewhat out of date.

I personally have experience with masquerading IPsec and PPTP clients running on MS W'98 and NT, configuring a registered−IP PPTP server, and using PPTP for network−to−network routing.

The information on masquerading a Private−IP PPTP server is from discussions with Len Bayles <len@isdi.com>, Simon Cocking <simon@ibs.com.au> and C. Scott Ananian <cananian@lcs.mit.edu>.

The home page for the PPTP−only Masquerade kernel patch for the 2.1.105+ and early 2.2.x kernel series is http://bmrc.berkeley.edu/people/chaffee/linux_pptp.html.

The home page for the `ipportfw` port−forwarding kernel patch and configuration tool for 2.0.x kernels is http://www.ox.compsoc.org.uk/~steve/portforwarding.html. Port forwarding is built into the 2.2.x kernel, and the `ipmasqadm` configuration tool for controlling 2.2.x port forwarding can be obtained at http://juanjox.kernelnotes.org/.

The home page for the `ipfwd` generic IP redirector is http://www.pdos.lcs.mit.edu/~cananian/Projects/IPfwd/.

Profuse thanks to Gordon Chaffee <chaffee@cs.berkeley.edu> for coding and sharing a patch to traceroute that allows tracing GRE traffic. It should prove invaluable in troubleshooting if your GRE traffic is being blocked somewhere. The patch is available at http://www.wolfenet.com/~jhardin/pptp−traceroute.patch.gz

More thanks to Steve Chinatti <chinatti@alumni.Princeton.EDU> for contributing his original IPsec masquerade hack, from which I shamelessly stole some very important ideas...

More information on setting up firewall rules to run automatically − including how to automatically use the correct IP address in a dynamic−IP environment − can be found at http://www.wolfenet.com/~jhardin/ipfwadm/invocation.html

The home page for Linux FreeS/WAN (IPsec for Linux) is http://www.xs4all.nl/~freeswan/ − this is the preferred Linux VPN solution.

A native Linux PPTP server called PoPToP is available at http://www.moretonbay.com/vpn/pptp.html − for the most up−to−date information about PPTP on Linux, go there.

Paul Cadach <paul@odt.east.telecom.kz> has made patches that add MS−CHAP−v2, MPPE and Multilink support to Linux pppd. See ftp://ftp.east.telecom.kz/pub/src/networking/ppp/ppp−2.3.5−my.tgz for MS−CHAP and MPPE, and ftp://ftp.east.telecom.kz/pub/src/networking/ppp/multilink/ppp−2.3.5−mp.tgz for Multilink. Another (possibly related) set of pppd patches are available at the PoPToP download site at http://www.moretonbay.com/vpn/download_pptp.html.

The home page for the original Linux PPTP project is http://www.pdos.lcs.mit.edu/~cananian/Projects/PPTP and a patch to add PPTP server capability to it is available at http://debs.fuller.edu/cgi−bin/display?list=pptp&msg=222

Thanks to Eric Raymond for maintaining the Jargon File, and Denis Howe for The Free On−line Dictionary of Computing.

1.3 Copyright & Disclaimer

The information presented in this document is correct to the best of my knowledge. IP Masquerading is *experimental*, and it is possible that I have made a mistake in writing or testing the kernel patch or composing the instructions in this document; you should determine for yourself if you want to make the changes outlined in this document.

> **THE AUTHOR IS NOT RESPONSIBLE FOR ANY DAMAGES INCURRED DUE TO ACTIONS TAKEN BASED ON THE INFORMATION IN THIS DOCUMENT. BACK UP ANY AND ALL CRITICAL INFORMATION BEFORE IMPLEMENTING THE CHANGES OUTLINED IN THIS DOCUMENT. MAKE SURE YOU HAVE A WORKING, BOOTABLE KERNEL AVAILABLE BEFORE PATCHING AND RECOMPILING YOUR KERNEL AS OUTLINED IN THIS DOCUMENT.**

In other words, take sensible precautions.

2. Background Knowledge

2.1 What is a VPN?

A Virtual Private Network, or "VPN", is a tunnel that carries private network traffic from one endpoint system to another over a public network (such as the Internet) without the traffic being aware that there are intermediate hops between the endpoints, or the intermediate hops being aware they are carrying the network packets that are traversing the tunnel. The tunnel may optionally compress and/or encrypt the data, providing enhanced performance and some measure of security.

The "Virtual" part stems from the fact that you are constructing a private link over a public network, rather than actually buying a direct hardwired link over leased lines. The VPN allows you to pretend you are using a leased line or direct telephone call to communicate between the endpoints.

You may find the VPN FAQ at http://kubarb.phsx.ukans.edu/~tbird/vpn/FAQ.html informative.

2.2 What is IPsec?

IPsec is a set of standard protocols for implementing secure communications and encryption key exchange between computers. It can be used to implement a VPN.

An IPsec VPN generally consists of two communications channels between the endpoint hosts: a key−exchange channel over which authentication and encryption key information is passed, and one or more data channels over which private network traffic is carried.

The key−exchange channel is a standard UDP connection to and from port 500. The data channels carrying the traffic between the client and server use IP protocol number 50 (ESP).

More information is available in F−Secure's IPsec FAQ at http://www.Europe.F−Secure.com/support/vpn+/faq/techfaq.html, and in RFC2402 (the AH protocol, IP protocol number 51), RFC2406 (the ESP protocol, IP protocol number 50), and RFC2408 (the ISAKMP key−exchange protocol).

IPsec is a peer−to−peer protocol. However, since most people will be exposed to it in the form of an originate−only Windows client being used to access a central network security gateway, "client" will be used to refer to the endpoint host that the user is sitting in front of and "server" will be used to refer to the central network security gateway.

Important note: If your VPN is based on the AH protocol (including AH+ESP), it cannot be masqueraded. The AH protocol specifies a cryptographic checksum across portions of the IP header, including the IP addresses. IP Masquerade is implemented by modifying the source IP address for outbound packets and the destination IP address for inbound packets. Since the masquerading gateway cannot participate in the encryption key exchange, it cannot generate the correct cryptographic checksums for the modified IP headers. Thus the modified IP packets will be discarded by the recipient as invalid, because they fail the cryptographic checksum test.

2.3 What is PPTP?

PPTP stands for ***P***oint−to−***P***oint ***T***unnelling ***P***rotocol. It is a Microsoft−proposed protocol for implementing a VPN.

The PPTP VPN protocol consists of two communications channels between the client and server: a control channel over which link−management information is passed, and a data channel over which (possibly encrypted) private network traffic is carried.

The control channel is a standard TCP connection to port 1723 on the server. The data channel carrying the private network traffic uses IP protocol number 47 (GRE), a generic encapsulation protocol described in RFC1701. The transparent transmission of data over the data channel is achieved by negotiating a standard PPP connection over it, just as if it were a dialup connection directly from the client to the server. The options negotiated over the tunnel by PPP control whether the data is compressed and/or encrypted, thus PPTP itself has nothing to do with encryption.

The details of the PPTP protocol are documented in RFC2637.

Microsoft's implementation of the PPTP protocol is not considered very secure. If you're interested in the details, here are three separate analyses:

http://www.counterpane.com/pptp.html

http://www.geek−girl.com/bugtraq/1999_1/0664.html

http://oliver.efri.hr/~crv/security/bugs/NT/pptp2.html

2.4 What is FWZ?

FWZ is a proprietary encryption protocol developed by Check Point Software Technologies. It is used in VPNs that are built around their Firewall−1 product.

A Checkpoint−based firewall can be configured in several modes. The "FWZ Encapsulation" mode *cannot* be masqueraded. The "IKE" mode, which uses standard IPsec protocols, can be masqueraded with minor configuration changes on the VPN gateway.

2.5 Why masquerade a VPN client?

Most current VPN clients assume you will be connecting the client computer directly to the internet. Doing this when you have only a single connection for internet access bypasses your Linux firewall and the security and access−sharing capabilities that it provides. Extending the Linux firewall to also masquerade VPN traffic allows you to retain the firewalling security provided by the Linux firewall as well as permitting the other systems on your local network to access the internet regardless of whether or not the VPN network connection is active.

If your firewall is being used in a corporate setting you may also wish to require your VPN client users to go through that firewall for security reasons, rather than providing them with modems so they can dial out on their own when they need to use VPN. VPN Masquerade allows you to do so even if the desktops do not have registered IP addresses.

2.6 Can several clients on my local network use IPsec simultaneously?

Yes, though there may occasionally be minor problems.

The IPsec protocols define a method for identifying the traffic streams called the *Security Parameters Index* ("SPI"). Unfortunately the SPI used by outbound traffic is different from the SPI used by inbound traffic, and there is no other identifying information available that is not encrypted, so association of the inbound and outbound data streams is difficult and not perfectly reliable.

IPsec Masquerade attempts to associate inbound and outbound ESP traffic by serializing new connections. While this has worked well in testing, it cannot be guaranteed to be perfectly reliable, and the serialization of new traffic may result in some timeouts if the link is saturated or if many local IPsec hosts attempt to initiate communications or rekey with the same remote IPsec host simultaneously.

It is also assumed that should this association scheme fail to associate the traffic streams correctly, the IPsec hosts themselves will discard the incorrectly routed traffic because it will have the wrong SPI values. This is required by the IPsec RFCs.

These problems could be eliminated if there was some way to sniff the new SPI values from the ISAKMP key exchange before any ESP traffic appears, but unfortunately that portion of the key exchange is encrypted.

To minimize the problems associated with this, it is recommended that you open a command window on your masqueraded IPsec host and run the "ping" program pinging a host on the remote network for as long as you have the tunnel up.

See the IPsec technical notes at the end of the document for more details.

2.7 Can several clients on my local network use PPTP simultaneously?

Yes.

You must enable PPTP Call ID masquerade when configuring your kernel in order to distinguish between multiple data streams from the same server. PPTP masq with Call ID masq enabled will support many concurrent masqueraded sessions with no restrictions on which server a client can call.

The PPTP RFC specifies in section 3.1.3 that there may only be one control channel connection between two systems. This *should* mean that you can only masquerade one PPTP session at a time with a given remote server, but in practice the MS implementation of PPTP does not enforce this, at least not as of NT 4.0 Service Pack 4. If the PPTP server you're trying to connect to only permits one connection at a time, it's following the protocol rules properly. Note that this does not affect a masqueraded server, only multiple masqueraded clients attempting to contact the same remote server.

For another alternative, see the next question...

2.8 Can I access the remote network from my entire local network?

Yes. However, your VPN client must be able to forward IP traffic.

This means that you'll either have to use a Linux VPN client or a MS NT VPN client. The IP stack in W'95 and W'98 does not support IP forwarding. NT Workstation will work for this, and is less expensive than NT Server if you're only using it to route encrypted traffic.

If you cannot install a Linux or NT−based VPN client, then you'll have to enable PPTP Call−ID masquerade if you are using PPTP, and install VPN client software on every system you want to provide access for. This is inefficient, aesthetically revolting, a security weakness, and may not work if the PPTP server correctly implements the protocol, but it's cheaper than licensing NT.

Network−to−network routing this way works very well. This is how I have my home network set up for telecommuting. It does require a little more networking knowhow than simply giving everybody their own VPN client.

In my experience, network−to−network routing in a pure−MS environment requires RRAS be installed at both ends of the tunnel.

2.9 Why masquerade the VPN server?

If your VPN server has a registered IP address you do not need to masquerade it, simply configure your firewall to route the VPN traffic properly as described below.

If your VPN server has a Private−Network IP address you will need to redirect the inbound traffic to it and masquerade the outbound traffic from it. Masquerading allows you to make a VPN server available to the internet even if you only have one assigned IP address. This should work even if your IP address is dynamically assigned: you would publicize the IP address for clients through the use of a third−party dynamic DNS service such as that provided by DDNS.ORG or CJB.NET and configure the clients to connect to a system named `our−company.ddns.org` or something similar. Note that this is a security risk, because it is possible for an incorrect IP address to be retrieved from the dynamic DNS server through timing problems, a failure to properly register the current dynamic IP address, or a third party registering a different IP address under the system name.

2.10 Why patch the Linux kernel?

The largest problem in masquerading VPN traffic is that the stock Linux IP masquerade has no special awareness of IP protocols other than TCP, UDP and ICMP.

All IP traffic may be forwarded and filtered by IP address, but masquerading IP protocols other than TCP, UDP and ICMP requires modifying the kernel.

The PPTP control channel is plain TCP and requires no special setup beyond letting it through the firewall and masquerading it.

Masquerading the IPsec and PPTP data channels requires a modification that adds support for the ESP and GRE protocols to the masquerading code, and masquerading the ISAKMP key exchange protocol requires a modification that prevents masquerade from altering the UDP source port number and adds tracking of the ISAKMP cookie values instead of the port number.

2.11 Current Status

The 2.0.x kernel patch works on kernel 2.0.36 and is incorporated into the standard 2.0.37 and higher kernel releases. It may work on earlier kernels but I have not tested it, and I recommend you upgrade to kernel 2.0.38 anyway for security reasons if you are running an older kernel.

The 2.2.x kernel patch works on kernels from 2.2.5 to 2.2.17 and may work on earlier kernels, but that has not been tested. It has been submitted for inclusion in the standard 2.2.18 release.

I don't have the resources to follow the development kernels, so at this time no work on VPN Masquerade for 2.3.x or 2.4.x has taken place. If you know someone who *is* working on this, please let me know.

The 2.0.x kernel patch has been tested and works on x86 and Sparc systems, and the 2.2.x kernel patch has been tested and works on x86 and PowerPC systems, but there should be no major problems in porting to other architectures. I believe the architecture dependencies would only be in endian−ness within the bitmaps in the GRE header definition used to format debugging log messages. If anyone ports this to a non−Intel architecture I'd appreciate hearing about it so I can merge any changes into the master copy.

A PPTP−only kernel patch for the 2.1.105+ and early 2.2.x kernels is available at http://bmrc.berkeley.edu/people/chaffee/linux_pptp.html.

See the VPN Masquerade home page at http://www.impsec.org/linux/masquerade/ip_masq_vpn.html for the status of the VPN Masq patches, and http://bmrc.berkeley.edu/people/chaffee/linux_pptp.html for the status of the 2.1.105+/2.2.x PPTP−only Masq patch.

---

3. Configuring the Linux firewall

3.1 Example network

For the Private−IP configuration examples in this document we will use this sample network:

```
Internet-------- 200.200.200.*   ppp0 or  200.200.200.200 eth1
                                 Dual-Homed Linux Firewall
            .--- 10.0.0.1        eth0
            |
            |--- 10.0.0.2        VPN client or server
            |
```

For the registered−IP configuration examples in this document we will use this sample network:

```
Internet-------- 200.200.200.200 eth1
                                 Dual-Homed Linux Firewall
            .--- 222.0.0.1       eth0
            |
            |--- 222.0.0.2       VPN client or server
            |
```

The VPN server that the example clients connect to will be `199.0.0.1`

The VPN clients that the connect to the example server will be `199.0.0.2` and `199.0.0.3`

3.2 Determining what needs to be done on the firewall

If your VPN client or server has a registered internet IP address you do *not* need to masquerade or modify your kernel – the stock kernel will successfully route all VPN traffic. You can skip directly to the registered–IP setup sections below.

If your VPN client or server has a Private–Network IP address as described in RFC1918 you will need to patch your kernel (unless your kernel is 2.0.37 or higher in the 2.0.x series).

If you are setting up a masqueraded VPN server, you will also have to obtain and install the following two packages:

- To redirect the inbound TCP/UDP traffic (the 1723/tcp PPTP control channel or the 500/udp ISAKMP channel), you need the appropriate `ipportfw` port–forwarding kernel patch and configuration tool from http://www.ox.compsoc.org.uk/~steve/portforwarding.html. Port forwarding has been incorporated into the 2.2.x kernel. See `man ipmasqadm` for configuration details. If `ipmasqadm` is not included with your distribution it can be obtained at http://juanjox.kernelnotes.org/.
- To redirect the initial inbound tunnel traffic (GRE for PPTP and ESP for IPsec), you need the `ipfwd` generic–IP redirector from http://www.pdos.lcs.mit.edu/~cananian/Projects/IPfwd/.

You *do not* need port forwarding or ipfwd if you are masquerading only clients.

3.3 Patching and configuring the 2.0.x kernel for VPN Masquerade support

1. Install the kernel source (preferably version 2.0.37), which you can obtain from http://www.kernel.org/ or a mirror. The sources should be automatically extracted into a directory named `/usr/src/linux`.
2. Configure and test standard IP Masquerading (see the IP Masquerade HOWTO). Doing this will familiarize you with recompiling your kernel and introduce you to IP Masquerading in general.
3. *Back up your kernel sources.*
4. Obtain the kernel patch if necessary.

   If your kernel version is 2.0.36 or lower, obtain the 2.0.x VPN Masquerade kernel patch from the VPN Masquerade home page in the "Resources" section above.

   If your kernel version is 2.0.37 or higher in the 2.0.x series, you do not need to apply any patches. The VPN Masquerade code is included in the kernel. Skip the discussion of patching the kernel.

   For the purposes of this document we'll assume you've saved the appropriate patch in

`/usr/src/ip_masq_vpn.patch.gz`.

5. Apply the VPN Masquerade patch to your kernel if necessary:

   ♦ Change to the kernel source directory:
     `cd /usr/src/linux`
   ♦ Apply the patch:
     `zcat ../ip_masq_vpn.patch.gz | patch -l -p0 > vpn-patch.log 2>&1`

     Note that the options are "dash lowercase L, dash lowercase P zero". You may get odd results if you change the order of the arguments, as patch seems to be sensitive to the order they appear on the command line.
   ♦ Check the `vpn-patch.log` file to see if any hunks failed. If you get failed hunks, then you probably either omitted the options or ran the patch program from the wrong directory. Restore your kernel from the backup and try again.

6. If you are masquerading a VPN server, obtain and install the `ipportfw` patch from the site given above.

   There is a known conflict between the VPN Masquerade patch and two other networking patches: the IP Firewall Chains patch and the ipportfw patch. They are all trying to add options at the same location in `net/ipv4/Config.in`, and the changes made by one patch alter the context that the other patches are looking for.

   If you're applying the VPN Masquerade patch and the IP Firewall Chains or ipportfw patches to your 2.0.x kernel, you will have to manually edit `net/ipv4/Config.in` and add the block of configuration options from the patch file that fails to work. Looking at the patch file should show you where in `net/ipv4/Config.in` the new options should be added.

   The syntax of patch files is simple. For each block of changes to make, there are two sections: the first shows the "before" state, with an indication of lines to be changed or deleted; the second shows the "after" state, with an indication of the lines that have been changed or added. Use the first section to find where to add the lines, and add the lines that are indicated in the second section.

   This should not be a problem once those patches are updated for 2.0.37+

7. Configure your kernel and select the following options – say *YES* to the following:

```
* Prompt for development and/or incomplete code/drivers
  CONFIG_EXPERIMENTAL
  - You must enable this to see the VPN Masq options.

* Networking support
  CONFIG_NET

* Network firewalls
  CONFIG_FIREWALL

* TCP/IP networking
  CONFIG_INET

* IP: forwarding/gatewaying
  CONFIG_IP_FORWARD
```

```
* IP: firewalling
  CONFIG_IP_FIREWALL

* IP: masquerading (EXPERIMENTAL)
  CONFIG_IP_MASQUERADE
  - This is required.

* IP: PPTP masq support (EXPERIMENTAL)
  CONFIG_IP_MASQUERADE_PPTP
  - Enables PPTP data channel masquerading, if you are
    masquerading a PPTP client or server.

* IP: PPTP Call ID masq support (EXPERIMENTAL)
  CONFIG_IP_MASQUERADE_PPTP_MULTICLIENT
  - Enables PPTP Call ID masquerading; only necessary if
    you will be masquerading more than one client trying
    to connect to the same remote server. DO NOT enable
    this option if you will be masquerading a PPTP server.

* IP: IPsec ESP & ISAKMP masq support (EXPERIMENTAL)
  CONFIG_IP_MASQUERADE_IPSEC
  - Enables IPsec masquerade, if you are masquerading an
    IPsec host.

* IP: IPSEC masq table lifetime (minutes)
  - See your network administrator to determine what the
    "rekey interval" or "key lifetime" is set to. The
    default lifetime of masq table entries is thirty
    minutes.  If your rekey interval is greater than
    thirty minutes, then you should increase the lifetime
    to a value slightly greater than the rekey interval.

* IP: always defragment
  CONFIG_IP_ALWAYS_DEFRAG
  - Highly recommended for a firewall.
```

NOTE: These are just the settings you need for masquerading. Select whatever other options you need for your specific setup.

8. Recompile the kernel and install it for testing. Don't replace a known working kernel with your new kernel until you have proven it works.

To determine whether the running kernel includes VPN Masquerade support, run the following command:

```
grep -i masq /proc/ksyms
```

...and look for the following entries:

- IPsec masquerade: `ip_masq_out_get_isakmp`, `ip_masq_in_get_isakmp`, `ip_fw_masq_esp` and `ip_fw_demasq_esp`
- PPTP masquerade: `ip_fw_masq_gre` and `ip_fw_demasq_gre`
- PPTP Call–ID masquerade: `ip_masq_pptp`

If you don't see these entries, VPN Masquerade support is probably not available. If you get complaints about `/proc/ksyms` not being available or `/proc` not being available, make sure that you have enabled the `/proc` filesystem in your kernel configuration.

See the Kernel HOWTO for more details on configuring and recompiling your kernel.

If you are using IPsec masquerade and your system is generating General Protection errors (see `/var/log/messages`) or is locking up, see the VPN Masquerade home page for an update. This patch is for 2.0.38, but should work on earlier kernels. It has been submitted to Alan Cox for inclusion in the 2.0.39 kernel.

3.4 Patching and configuring the 2.2.x kernel for VPN Masquerade support

1. Install the kernel source (preferably version 2.2.17 or later), which you can obtain from http://www.kernel.org/ or a mirror. The sources should be automatically extracted into a directory named `/usr/src/linux`.

2. Configure and test standard IP Masquerading (see the IP Masquerade HOWTO). Doing this will familiarize you with recompiling your kernel and introduce you to IP Masquerading in general.

3. *Back up your kernel sources.*

4. Obtain the kernel patch from the VPN Masquerade home page in the "Resources" section above.

   For the purposes of this document we'll assume you've saved the appropriate patch in `/usr/src/ip_masq_vpn.patch.gz`.

5. Apply the VPN Masquerade patch to your kernel if necessary:

   ♦ Change to the source directory:
   ```
   cd /usr/src
   ```
   ♦ Apply the patch:
   ```
   zcat ip_masq_vpn.patch.gz | patch -l -p0 >
   vpn-patch.log 2>&1
   ```

   Note that the options are "dash lowercase L, dash lowercase P zero". You may get odd results if you change the order of the arguments, as patch seems to be sensitive to the order they appear on the command line.

   Also note that the directory you run the patch command in is different for the 2.2.x kernel patch

   ♦ Check the `vpn-patch.log` file to see if any hunks failed. If you get failed hunks, then you probably either omitted the options or ran the patch program from the wrong directory. Restore your kernel from the backup and try again.

6. If you are masquerading a VPN server you do *not* need the `ipportfw` patch as port forwarding is now built–in. See the `ipmasqadm` man page for more details. If `ipmasqadm` is not included with your distribution it can be obtained at http://juanjox.kernelnotes.org/.

7. Configure your kernel and select the following options – say *YES* to the following:
```
* Prompt for development and/or incomplete code/drivers
  CONFIG_EXPERIMENTAL
  - You must enable this to see the VPN Masq options.
```

```
* Networking support
  CONFIG_NET

* Network firewalls
  CONFIG_FIREWALL

* TCP/IP networking
  CONFIG_INET

* IP: firewalling
  CONFIG_IP_FIREWALL

* IP: always defragment
  CONFIG_IP_ALWAYS_DEFRAG
  - Required for masquerading. This may or may not
    be in your kernel config. If not, you should
    run this in your startup scripts:
      echo 1 > /proc/sys/net/ipv4/ip_always_defrag

* IP: masquerading (EXPERIMENTAL)
  CONFIG_IP_MASQUERADE
  - This is required.

* IP: masquerading special modules support
  CONFIG_IP_MASQUERADE_MOD
  - This is required.

* IP: ipportfw masq support (EXPERIMENTAL)
  CONFIG_IP_MASQUERADE_IPPORTFW
  - Enable this if you will be masquerading a VPN server.

* IP: PPTP masq support
  CONFIG_IP_MASQUERADE_PPTP
  - Enables PPTP data channel masquerading, if you are
    masquerading a PPTP client or server. This is now
    available as a module.
    Note that you no longer need to specify Call-ID masquerade.

* IP: IPsec ESP & ISAKMP masq support (EXPERIMENTAL)
  CONFIG_IP_MASQUERADE_IPSEC
  - Enables IPsec masquerade, if you are masquerading an
    IPsec host. This is now available as a module.

* IP: IPsec masq table lifetime (minutes)
  - See your network administrator to determine what the
    "rekey interval" or "key lifetime" is set to. The default
    lifetime of masq table entries is thirty minutes. If
    your rekey interval is greater than thirty minutes,
    then you should increase the lifetime to a value
    slightly greater than the rekey interval.

* IP: Enable parallel sessions (possible security risk - see help)
  CONFIG_IP_MASQUERADE_IPSEC_PAROK
  - See the IPsec masquerade technical notes and special
    security considerations section of the HOWTO for
    security considerations to be aware of when
    masquerading IPsec traffic. If you are only
    masquerading one IPsec client this setting has no
    effect.
```

Say *NO* to the following:

```
* IP: GRE tunnels over IP
  CONFIG_NET_IPGRE
  - This, confusingly, has *NOTHING* to do with PPTP.
    It enables support for GRE tunnels as used by Cisco
    routers. The fact that you see this option does not
    imply that PPTP support is available. You still need
    to apply the VPN Masquerade patch if the PPTP options
    listed above do not appear when you are configuring
    your kernel. DO NOT enable this unless you are setting
    up a GRE tunnel to a Cisco router.
```

NOTE: These are just the settings you need for masquerading. Select whatever other options you need for your specific setup.

8. Recompile the kernel and install it for testing. Don't replace a known working kernel with your new kernel until you have proven it works.

To determine whether the running kernel includes VPN Masquerade support, run the following command:

```
grep -i masq /proc/ksyms
```

...and look for the following entries:

- IPsec masquerade: `ip_masq_esp` and `ip_demasq_esp`
- PPTP masquerade: `ip_masq_pptp_tcp` and `ip_demasq_pptp_tcp`

Or run:

```
lsmod
```

...and look for the following entries:

- IPsec masquerade: `ip_masq_ipsec`
- PPTP masquerade: `ip_masq_pptp`

If you don't see these entries, VPN Masquerade support is probably not available – did you remember to `modprobe ip_masq_pptp.o` or `modprobe ip_masq_ipsec.o` if you compiled them as modules? If VPN masquerade stops working after you reboot, did you remember to add the `modprobe` commands into your `/etc/rc.d/rc.local` startup script?

If you get complaints about `/proc/ksyms` not being available or `/proc` not being available, make sure that you have enabled the `/proc` filesystem in your kernel configuration.

See the Kernel HOWTO for more details on configuring and recompiling your kernel.

3.5 ipfwadm setup for a Private–IP VPN Client or Server

The firewall must now be configured to masquerade the outbound VPN traffic. You may wish to visit http://www.wolfenet.com/~jhardin/ipfwadm.html to take a look at a GUI wrapper around the ipfwadm command that automates a lot of security–related packet filtering setup.

The minimum firewall rules are:

```
# Set the default forwarding policy to DENY:
ipfwadm -F -p deny
```

```
# Allow local-network traffic
ipfwadm -I -a accept    -S 10.0.0.0/8 -D 0.0.0.0/0  -W eth0
ipfwadm -O -a accept    -S 0.0.0.0/0  -D 10.0.0.0/8 -W eth0
# Masquerade traffic for internet addresses and allow internet traffic
ipfwadm -F -a accept -m -S 10.0.0.0/8 -D 0.0.0.0/0  -W ppp0
ipfwadm -O -a accept    -S 0.0.0.0/0  -D 0.0.0.0/0  -W ppp0
ipfwadm -I -a accept    -S 0.0.0.0/0  -D 0.0.0.0/0  -W ppp0
```

or, if you have a permanent connection,

```
ipfwadm -F -a accept -m -S 10.0.0.0/8 -D 0.0.0.0/0  -W eth1
ipfwadm -O -a accept    -S 0.0.0.0/0  -D 0.0.0.0/0  -W eth1
ipfwadm -I -a accept    -S 0.0.0.0/0  -D 0.0.0.0/0  -W eth1
```

This is a completely open setup, though. It will masquerade *any* traffic from *any* host on the local network destined for *any* host on the internet, and provides *no* security at all.

A tight firewall setup would only allow traffic between the client and the server, and would block everything else:

```
# Set the default policy to DENY:
ipfwadm -I -p deny
ipfwadm -O -p deny
ipfwadm -F -p deny
# Allow local-network traffic
ipfwadm -I -a accept -S 10.0.0.0/8 -D 0.0.0.0/0  -W eth0
ipfwadm -O -a accept -S 0.0.0.0/0  -D 10.0.0.0/8 -W eth0
# Masquerade only VPN traffic between the VPN client and the VPN server
ipfwadm -F -a accept -m -P udp -S 10.0.0.2/32 500 -D 199.0.0.1/32 500  -W ppp0
ipfwadm -F -a accept -m -P tcp -S 10.0.0.2/32     -D 199.0.0.1/32 1723 -W ppp0
ipfwadm -F -a deny      -P tcp -S 10.0.0.2/32     -D 199.0.0.1/32      -W ppp0
ipfwadm -F -a deny      -P udp -S 10.0.0.2/32     -D 199.0.0.1/32      -W ppp0
ipfwadm -F -a accept -m -P all -S 10.0.0.2/32     -D 199.0.0.1/32      -W ppp0
ipfwadm -O -a accept    -P udp -S 200.200.200.0/24 500 -D 199.0.0.1/32 500  -W ppp0
ipfwadm -O -a accept    -P tcp -S 200.200.200.0/24     -D 199.0.0.1/32 1723 -W ppp0
ipfwadm -O -a deny      -P tcp -S 200.200.200.0/24     -D 199.0.0.1/32      -W ppp0
ipfwadm -O -a deny      -P udp -S 200.200.200.0/24     -D 199.0.0.1/32      -W ppp0
ipfwadm -O -a accept    -P all -S 200.200.200.0/24     -D 199.0.0.1/32      -W ppp0
ipfwadm -I -a accept    -P udp -S 199.0.0.1/32 500     -D 200.200.200.0/24 500 -W ppp0
ipfwadm -I -a accept    -P tcp -S 199.0.0.1/32 1723    -D 200.200.200.0/24     -W ppp0
ipfwadm -I -a deny      -P tcp -S 199.0.0.1/32         -D 200.200.200.0/24     -W ppp0
ipfwadm -I -a deny      -P udp -S 199.0.0.1/32         -D 200.200.200.0/24     -W ppp0
ipfwadm -I -a accept    -P all -S 199.0.0.1/32         -D 200.200.200.0/24     -W ppp0
```

or, if you have a permanent connection,

```
ipfwadm -F -a accept -m -P udp -S 10.0.0.2/32 500 -D 199.0.0.1/32 500  -W eth1
ipfwadm -F -a accept -m -P tcp -S 10.0.0.2/32     -D 199.0.0.1/32 1723 -W eth1
ipfwadm -F -a deny      -P tcp -S 10.0.0.2/32     -D 199.0.0.1/32      -W eth1
ipfwadm -F -a deny      -P udp -S 10.0.0.2/32     -D 199.0.0.1/32      -W eth1
ipfwadm -F -a accept -m -P all -S 10.0.0.2/32     -D 199.0.0.1/32      -W eth1
ipfwadm -O -a accept    -P udp -S 200.200.200.200/32 500 -D 199.0.0.1/32 500  -W eth1
ipfwadm -O -a accept    -P tcp -S 200.200.200.200/32     -D 199.0.0.1/32 1723 -W eth1
ipfwadm -O -a deny      -P tcp -S 200.200.200.200/32     -D 199.0.0.1/32      -W eth1
ipfwadm -O -a deny      -P udp -S 200.200.200.200/32     -D 199.0.0.1/32      -W eth1
ipfwadm -O -a accept    -P all -S 200.200.200.200/32     -D 199.0.0.1/32      -W eth1
ipfwadm -I -a accept    -P udp -S 199.0.0.1/32 500  -D 200.200.200.200/32 500 -W eth1
ipfwadm -I -a accept    -P tcp -S 199.0.0.1/32 1723 -D 200.200.200.200/32     -W eth1
ipfwadm -I -a deny      -P tcp -S 199.0.0.1/32      -D 200.200.200.200/32     -W eth1
```

```
ipfwadm -I -a deny      -P udp -S 199.0.0.1/32      -D 200.200.200.200/32     -W eth1
ipfwadm -I -a accept    -P all -S 199.0.0.1/32      -D 200.200.200.200/32     -W eth1
```

Note: these rules only allow VPN traffic and block *everything else*. You will have to add rules for any other traffic you wish to permit, such as DNS, HTTP, POP, IMAP, etc.

3.6 ipchains setup for a Private−IP VPN Client or Server

The minimum ipchains firewall rules are:

```
# Set the default forwarding policy to DENY:
ipchains -P forward DENY
# Allow local-network traffic
ipchains -A input   -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0  -i eth0
ipchains -A output  -j ACCEPT -s 0.0.0.0/0  -d 10.0.0.0/8 -i eth0
# Masquerade traffic for internet addresses and allow internet traffic
ipchains -A forward -j MASQ   -s 10.0.0.0/8 -d 0.0.0.0/0  -i ppp0
ipchains -A output  -j ACCEPT -s 0.0.0.0/0  -d 0.0.0.0/0  -i ppp0
ipchains -A input   -j ACCEPT -s 0.0.0.0/0  -d 0.0.0.0/0  -i ppp0
```

or, if you have a permanent connection,

```
ipchains -A forward -j MASQ   -s 10.0.0.0/8 -d 0.0.0.0/0  -i eth1
ipchains -A output  -j ACCEPT -s 0.0.0.0/0  -d 0.0.0.0/0  -i eth1
ipchains -A input   -j ACCEPT -s 0.0.0.0/0  -d 0.0.0.0/0  -i eth1
```

This is a completely open setup, though. It will masquerade *any* traffic from *any* host on the local network destined for *any* host on the internet, and provides *no* security at all.

A tight firewall setup would only allow traffic between the client and the server, and would block everything else:

```
# Set the default policy to DENY:
ipchains -P input   DENY
ipchains -P output  DENY
ipchains -P forward DENY
# Allow local-network traffic
ipchains -A input  -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0  -i eth0
ipchains -A output -j ACCEPT -s 0.0.0.0/0  -d 10.0.0.0/8 -i eth0
# Masquerade only VPN traffic between the VPN client and the VPN server
# IPsec
ipchains -A forward -j MASQ   -p udp -s 10.0.0.2/32 500      -d 199.0.0.1/32 500     -i ppp0
ipchains -A output  -j ACCEPT -p udp -s 200.200.200.0/24 500 -d 199.0.0.1/32 500     -i ppp0
ipchains -A input   -j ACCEPT -p udp -s 199.0.0.1/32 500     -d 200.200.200.0/24 500 -i ppp0
ipchains -A forward -j MASQ   -p 50  -s 10.0.0.2/32          -d 199.0.0.1/32         -i ppp0
ipchains -A output  -j ACCEPT -p 50  -s 200.200.200.0/24     -d 199.0.0.1/32         -i ppp0
ipchains -A input   -j ACCEPT -p 50  -s 199.0.0.1/32         -d 200.200.200.0/24     -i ppp0
# PPTP
ipchains -A forward -j MASQ   -p tcp -s 10.0.0.2/32       -d 199.0.0.1/32 1723 -i ppp0
ipchains -A output  -j ACCEPT -p tcp -s 200.200.200.0/24  -d 199.0.0.1/32 1723 -i ppp0
ipchains -A input   -j ACCEPT -p tcp -s 199.0.0.1/32 1723 -d 200.200.200.0/24  -i ppp0
ipchains -A forward -j MASQ   -p 47  -s 10.0.0.2/32       -d 199.0.0.1/32      -i ppp0
ipchains -A output  -j ACCEPT -p 47  -s 200.200.200.0/24  -d 199.0.0.1/32      -i ppp0
ipchains -A input   -j ACCEPT -p 47  -s 199.0.0.1/32      -d 200.200.200.0/24  -i ppp0
```

or, if you have a permanent connection,

```
# IPsec
ipchains -A forward -j MASQ   -p udp -s 10.0.0.2/32 500        -d 199.0.0.1/32 500       -i eth1
ipchains -A output  -j ACCEPT -p udp -s 200.200.200.200/32 500 -d 199.0.0.1/32 500       -i eth1
ipchains -A input   -j ACCEPT -p udp -s 199.0.0.1/32 500       -d 200.200.200.200/32 500 -i eth1
ipchains -A forward -j MASQ   -p 50  -s 10.0.0.2/32            -d 199.0.0.1/32           -i eth1
ipchains -A output  -j ACCEPT -p 50  -s 200.200.200.200/32     -d 199.0.0.1/32           -i eth1
ipchains -A input   -j ACCEPT -p 50  -s 199.0.0.1/32           -d 200.200.200.200/32     -i eth1
# PPTP
ipchains -A forward -j MASQ   -p tcp -s 10.0.0.2/32        -d 199.0.0.1/32 1723  -i eth1
ipchains -A output  -j ACCEPT -p tcp -s 200.200.200.200/32 -d 199.0.0.1/32 1723  -i eth1
ipchains -A input   -j ACCEPT -p tcp -s 199.0.0.1/32 1723  -d 200.200.200.200/32 -i eth1
ipchains -A forward -j MASQ   -p 47  -s 10.0.0.2/32        -d 199.0.0.1/32       -i eth1
ipchains -A output  -j ACCEPT -p 47  -s 200.200.200.200/32 -d 199.0.0.1/32       -i eth1
ipchains -A input   -j ACCEPT -p 47  -s 199.0.0.1/32       -d 200.200.200.200/32 -i eth1
```

Note: these rules only allow VPN traffic. You will have to add rules for any other traffic you wish to permit, such as DNS, HTTP, POP, IMAP, etc.

Also note how there rules are much neater and easier to make sense of than the equivalent ipfwadm rules. This is because ipchains allows specification of all IP protocols, not just TCP, UDP, ICMP or ALL.

3.7 A note about dynamic IP addressing

If your firewall is assigned a dynamic IP address by your ISP (dialup accounts are this way, as are some cable internet services), then you should add the following to the startup script `/etc/rc.d/rc.local`:

```
echo 7 > /proc/sys/net/ipv4/ip_dynaddr
```

This enables dynamic IP address following, which means that should your connection drop and be reestablished, any active sessions will be updated to the new IP address rather than using the old IP address. This does not mean that the session will continue across the interruption, rather that it will be closed down quickly.

If you do not do this, then there may be a "dead period" after you redial and before old masq table entries expire where you're being masqueraded with the wrong IP address, which will prevent your establishing a connection.

This is particularly helpful if you are using a demand−dial daemon such as `diald` to manage your dialup connection.

See `/usr/src/linux/Documentation/networking/ip_dynaddr.txt` for more details.

3.8 Additional setup for a Private−IP VPN Server

If you are setting up VPN masquerade for a Private−IP VPN server (that is, you wish to provide for *inbound* connections as well as *outbound* connections), you also need to install two packet−forwarding utilities. One (`ipportfw`) forwards inbound TCP or UDP traffic addressed to a specific port on the firewall system to a system on the local network behind the firewall. This is used to redirect the initial inbound 1723/tcp PPTP control channel or 500/udp ISAKMP traffic to the VPN server. The other (`ipfwd`) is a more generic forwarding utility that allows you to do this for any IP protocol. It is used to forward the initial inbound 47/ip (GRE) or 50/ip (ESP) data channel traffic to the VPN server.

Outbound responses to the inbound 1723/tcp or 500/udp traffic are masqueraded using the normal

IP−Masquerade facilities in the Linux kernel. The outbound 47/ip or 50/ip traffic is masqueraded using the VPN−Masquerade kernel patch you installed earlier.

Once these utilities are installed, you must configure them to forward the traffic to the VPN server.

- Configuring `ipportfw` under 2.0.x kernels

  The following commands will set up `ipportfw` to forward the initial inbound 500/udp traffic to the IPsec server:

  ```
  # Static−IP ipportfw setup for IPsec
  # Clear the ipportfw forwarding table
  /sbin/ipportfw −C
  # Forward traffic addressed to the firewall's 500/udp port
  # to the IPsec server's 500/udp port
  /sbin/ipportfw −A −u 200.200.200.200/500 −R 10.0.0.2/500
  ```

  The following commands will set up `ipportfw` to forward the initial inbound 1723/tcp traffic to the PPTP server:

  ```
  # Static−IP ipportfw setup for PPTP
  # Clear the ipportfw forwarding table
  /sbin/ipportfw −C
  # Forward traffic addressed to the firewall's 1723/tcp port
  # to the PPTP server's 1723/tcp port
  /sbin/ipportfw −A −t 200.200.200.200/1723 −R 10.0.0.2/1723
  ```

  Note that the ipportfw command line requires the internet IP address of the firewall, and you cannot specify the interface (e.g. `ppp0`) as you can with ipfwadm. This means that for a dynamic−IP connection (such as a typical dialup PPP connection) you have to run these commands every time you connect to the internet and are assigned a new IP address. You can do this quite easily − simply add the following to your `/etc/ppp/ip−up` or `/etc/ppp/ip−up.local` script:

  ```
  # Dynamic−IP ipportfw setup for IPsec
  # Clear the ipportfw forwarding table
  /sbin/ipportfw −C
  # Forward traffic addressed to the firewall's 500/udp port
  # to the IPsec server's 500/udp port
  /sbin/ipportfw −A −u ${4}/500 −R 10.0.0.2/500
  ```

  or:

  ```
  # Dynamic−IP ipportfw setup for PPTP
  # Clear the ipportfw forwarding table
  /sbin/ipportfw −C
  # Forward traffic addressed to the firewall's 1723/tcp port
  # to the PPTP server's 1723/tcp port
  /sbin/ipportfw −A −t ${4}/1723 −R 10.0.0.2/1723
  ```

  See http://www.wolfenet.com/~jhardin/ipfwadm/invocation.html for more information on firewalling with a dynamic IP.

- Configuring `ipfwd` under both 2.0.x and 2.2.x kernels

  The following command will set up `ipfwd` to forward the initial inbound 50/ip traffic to the IPsec server:

```
/sbin/ipfwd --masq 10.0.0.2 50 &
```

The following command will set up `ipfwd` to forward the initial inbound 47/ip traffic to the PPTP server:

```
/sbin/ipfwd --masq 10.0.0.2 47 &
```

It should only be run once, from your `/etc/rc.d/rc.local` script.

The techniques described here can be generalized to allow masquerading of most any type of server − HTTP, FTP, SMTP, and so forth. Servers that are purely TCP− or UDP−based will not require `ipfwd`.

If you are masquerading a PPTP server you also need to make sure that you have *not* enabled PPTP Call ID masquerade in the kernel. Enabling PPTP Call ID masquerade builds in some assumptions that you're masquerading only PPTP clients, so enabling it will prevent proper masquerade of the PPTP server traffic. This also means that with the 2.0.x version of the patch you cannot simultaneously masquerade a PPTP server and PPTP clients.

3.9 ipfwadm setup for a Registered−IP VPN Server

Setting up a registered−IP VPN server behind a Linux firewall is a simple matter of making sure the appropriate routing and packet−filter commands are in place. Masquerading is not required.

Unfortunately the 2.0.x−series kernels will not let us specify IP protocol 47 or 50 directly, so this firewall is less secure than it could be. If this is a problem for you, then install the IP Firewall Chains kernel patch or move to the 2.1.x or 2.2.x series kernel, where you can filter by IP protocol.

The firewall rules will look something like this:

```
# This section should follow your other firewall rules.

# Specify the acceptable clients explicitly for tighter security.
# Allow the IPsec ISAKMP traffic in and out.
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P udp -S 199.0.0.2/32 500 -D 222.0.0.2/32 500
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P udp -D 199.0.0.2/32 500 -S 222.0.0.2/32 500
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P udp -S 199.0.0.3/32 500 -D 222.0.0.2/32 500
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P udp -D 199.0.0.3/32 500 -S 222.0.0.2/32 500
# Allow the PPTP control channel in and out.
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P tcp -S 199.0.0.2/32 -D 222.0.0.2/32 1723
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P tcp -D 199.0.0.2/32 -S 222.0.0.2/32 1723
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P tcp -S 199.0.0.3/32 -D 222.0.0.2/32 1723
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P tcp -D 199.0.0.3/32 -S 222.0.0.2/32 1723

# Block all other TCP and UDP traffic from the internet.
# This is essentially a "default deny TCP/UDP" that
# only applies to the internet interface.
ipfwadm -I -a deny -W eth1 -V 200.200.200.200 -P tcp
ipfwadm -I -a deny -W eth1 -V 200.200.200.200 -P udp

# Specify the acceptable clients explicitly for tighter security.
# Note that this is too open since we're forced to
# specify "-P all" rather than "-P 47" or "-P 50"...
# Allow the PPTP data channel and IPsec ESP traffic in and out.
```

```
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P all -S 199.0.0.2/32 -D 222.0.0.2/32
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P all -D 199.0.0.2/32 -S 222.0.0.2/32
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P all -S 199.0.0.3/32 -D 222.0.0.2/32
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P all -D 199.0.0.3/32 -S 222.0.0.2/32

# Block all other traffic from the internet.
# This is essentially a "default deny" that
# only applies to the internet interface.
ipfwadm -I -a deny -W eth1 -V 200.200.200.200
```

If you are installing firewall rules on forwarding and/or rules on the inner interface, you will have do do something similar. The above example only covers VPN traffic; you will have to merge it into your existing firewall setup to allow any other traffic you need.

3.10 ipfwadm setup for a Registered−IP VPN Client

Setting up a registered−IP VPN client behind a Linux firewall is similar to setting up a registered−IP VPN server.

The firewall rules will look something like this:

```
# Allow the IPsec ISAKMP traffic out and in.
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P udp -S 222.0.0.2/32 500 -D 199.0.0.1/32 500
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P udp -D 222.0.0.2/32 500 -S 199.0.0.1/32 500
# Allow the PPTP control channel out and in.
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P tcp -S 222.0.0.2/32 -D 199.0.0.1/32 1723
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P tcp -D 222.0.0.2/32 -S 199.0.0.1/32 1723

# Block all other TCP and UDP traffic from the internet.
# This is essentially a "default deny TCP/UDP" that
# only applies to the internet interface.
ipfwadm -I -a deny -W eth1 -V 200.200.200.200 -P tcp
ipfwadm -I -a deny -W eth1 -V 200.200.200.200 -P udp

# Note that this is too open since we're forced to
# specify "-P all" rather than "-P 47" or "-P 50"...
# Allow the PPTP data channel and IPsec ESP traffic out and in
ipfwadm -O -a accept -W eth1 -V 200.200.200.200 -P all -S 222.0.0.2/32 -D 199.0.0.1/32
ipfwadm -I -a accept -W eth1 -V 200.200.200.200 -P all -D 222.0.0.2/32 -S 199.0.0.1/32

# Block all other traffic from the internet.
# This is essentially a "default deny" that
# only applies to the internet interface.
ipfwadm -I -a deny -W eth1 -V 200.200.200.200
```

3.11 ipchains setup for a Registered−IP VPN Server

Setting up a registered−IP VPN server behind a Linux firewall is a simple matter of making sure the appropriate routing and packet−filter commands are in place. Masquerading is not required.

The firewall rules will look something like this:

```
# Specify the acceptable clients explicitly for tighter security.
# Allow the IPsec ISAKMP traffic in and out.
ipchains -A input  -j ACCEPT -p udp -s 199.0.0.2/32 500 -d 222.0.0.2/32 500 -i eth1
ipchains -A output -j ACCEPT -p udp -d 199.0.0.2/32 500 -s 222.0.0.2/32 500 -i eth1
```

```
ipchains -A input  -j ACCEPT -p udp -s 199.0.0.3/32 500 -d 222.0.0.2/32 500 -i eth1
ipchains -A output -j ACCEPT -p udp -d 199.0.0.3/32 500 -s 222.0.0.2/32 500 -i eth1
# Allow the IPsec ESP traffic in and out.
ipchains -A input  -j ACCEPT -p 50  -s 199.0.0.2/32     -d 222.0.0.2/32     -i eth1
ipchains -A output -j ACCEPT -p 50  -d 199.0.0.2/32     -s 222.0.0.2/32     -i eth1
ipchains -A input  -j ACCEPT -p 50  -s 199.0.0.3/32     -d 222.0.0.2/32     -i eth1
ipchains -A output -j ACCEPT -p 50  -d 199.0.0.3/32     -s 222.0.0.2/32     -i eth1
# Allow the PPTP control channel in and out.
ipchains -A input  -j ACCEPT -p tcp -s 199.0.0.2/32 -d 222.0.0.2/32 1723 -i eth1
ipchains -A output -j ACCEPT -p tcp -d 199.0.0.2/32 -s 222.0.0.2/32 1723 -i eth1
ipchains -A input  -j ACCEPT -p tcp -s 199.0.0.3/32 -d 222.0.0.2/32 1723 -i eth1
ipchains -A output -j ACCEPT -p tcp -d 199.0.0.3/32 -s 222.0.0.2/32 1723 -i eth1
# Allow the PPTP tunnel in and out.
ipchains -A input  -j ACCEPT -p 47  -s 199.0.0.2/32 -d 222.0.0.2/32      -i eth1
ipchains -A output -j ACCEPT -p 47  -d 199.0.0.2/32 -s 222.0.0.2/32      -i eth1
ipchains -A input  -j ACCEPT -p 47  -s 199.0.0.3/32 -d 222.0.0.2/32      -i eth1
ipchains -A output -j ACCEPT -p 47  -d 199.0.0.3/32 -s 222.0.0.2/32      -i eth1
```

If you are installing firewall rules on forwarding and/or rules on the inner interface, you will have do do something similar. The above example only covers VPN traffic; you will have to merge it into your existing firewall setup to allow any other traffic you need.

3.12 ipchains setup for a Registered−IP VPN Client

Setting up a registered−IP VPN client behind a Linux firewall is similar to setting up a registered−IP VPN server.

The firewall rules will look something like this:

```
# Allow the IPsec ISAKMP traffic out and in.
ipchains -A output -j ACCEPT -p udp -s 222.0.0.2/32 500 -d 199.0.0.1/32 500 -i eth1
ipchains -A input  -j ACCEPT -p udp -d 222.0.0.2/32 500 -s 199.0.0.1/32 500 -i eth1
# Allow the IPsec ESP traffic out and in.
ipchains -A output -j ACCEPT -p 50  -s 222.0.0.2/32     -d 199.0.0.1/32     -i eth1
ipchains -A input  -j ACCEPT -p 50  -d 222.0.0.2/32     -s 199.0.0.1/32     -i eth1
# Allow the PPTP control channel out and in.
ipchains -A output -j ACCEPT -p tcp -s 222.0.0.2/32 -d 199.0.0.1/32 1723 -i eth1
ipchains -A input  -j ACCEPT -p tcp -d 222.0.0.2/32 -s 199.0.0.1/32 1723 -i eth1
# Allow the PPTP tunnel out and in.
ipchains -A output -j ACCEPT -p 47  -s 222.0.0.2/32 -d 199.0.0.1/32      -i eth1
ipchains -A input  -j ACCEPT -p 47  -d 222.0.0.2/32 -s 199.0.0.1/32      -i eth1
```

3.13 VPN Masq and LRP

The Linux Router Project at http://www.linuxrouter.org/ provides a Linux−based firewall−on−a−floppy kit. With a '386 PC, two network cards, and a diskette drive, you can set up a full−featured masquerading firewall. No hard disk is needed.

VPN Masquerade is supposed to be included in LRP version 2.2.9 − to verify it is available, see if `ip_masq_ipsec` or `ip_masq_pptp` are listed in the loadable modules in `Package Settings -> Modules`, or grep `/proc/ksyms` as described above. If you want to add VPN masquerade to an earlier version of LRP then somebody on the LRP mailing list may be able to provide a diskette image for you, or you can roll your own kernel using the instructions available on the LRP home page.

The firewall rules would be added to the startup script file in `Network Settings -> Direct`

`Network Setup`.

3.14 VPN Masq on a system running FreeS/WAN or PoPToP

If you are going to be using the firewall as an IPsec gateway with FreeS/WAN, you *must not* enable IPsec masquerade. If you are going to be using the firewall as a PPTP server with PoPToP, or a PPTP client using the Linux PPTP client software, you *must not* enable PPTP masquerade.

VPN masquerade and a VPN client or server using the same protocols cannot at this time coexist on the same computer.

Your firewall *can*, however, be a FreeS/WAN IPsec VPN gateway while masquerading PPTP traffic, or vice–versa.

---

4. Configuring the VPN client

4.1 Configuring a MS W'95 client

1. Set up your routing so that the Linux firewall is your default gateway:
    1. Open `Control Panel/Network` or right–click "Network Neighborhood" and click on `Properties`.
    2. Click on the `Configuration` tab.
    3. In the list of installed network components, double–click on the "TCP/IP –> whatever–NIC–you–have" line.
    4. Click on the `Gateway` tab.
    5. Enter the local–network IP address of your Linux firewall. Delete any other gateways.
    6. Click on the "OK" button.
2. Test masquerading. For example, run "`telnet` *`my.isp.mail.server`* `smtp`" and you should see the mail server's welcome banner.
3. Install and configure the VPN software. For IPsec software follow the manufacturer's instructions. For MS PPTP:
    1. Open `Control Panel/Network` or right–click "Network Neighborhood" and click on `Properties`.
    2. Click on the `Configuration` tab.
    3. Click on the "Add" button, then double–click on the "Adapter" line.
    4. Select "Microsoft" as the manufacturer and add the "Virtual Private Networking Adapter" adapter.
    5. Reboot when prompted to.
    6. If you need to use strong (128–bit) encryption, download the strong encryption DUN 1.3 update from the MS secure site at http://mssecure.www.conxion.com/cgi–bin/ntitar.pl and install it, then reboot again when prompted to.
    7. Create a new dial–up phonebook entry for your PPTP server.
    8. Select the VPN adapter as the device to use, and enter the PPTP server's internet IP address as the telephone number.
    9. Select the `Server Types` tab, and check the compression and encryption checkboxes.
    10. Click on the "TCP/IP Settings" button.
    11. Set the dynamic/static IP address information for your client as instructed to by your PPTP server's administrator.

12. If you wish to have access to your local network while the PPTP connection is up, uncheck the "Use default gateway on remote network" checkbox.
13. Reboot a few more times, just from habit... :)

4.2 Configuring a MS W'98 client

1. Set up your routing so that the Linux firewall is your default gateway and test masquerading as described above.
2. Install and configure the VPN software. For IPsec software follow the manufacturer's instructions. For MS PPTP:
    1. Open `Control Panel/Add or Remove Software` and click on the `Windows Setup` tab.
    2. Click on the `Communications` option and click the "Details" button.
    3. Make sure the "Virtual Private Networking" option is checked. Then click the "OK" button.
    4. Reboot when prompted to.
    5. If you need to use strong (128−bit) encryption, download the strong encryption VPN Security update from the MS secure site at http://mssecure.www.conxion.com/cgi−bin/ntitar.pl and install it, then reboot again when prompted to.
3. Create and test a new dial−up phonebook entry for your VPN server as described above.

4.3 Configuring a MS W'ME client

I haven't seen one of these yet. I expect the procedure is very similar to that for W'98. Could someone who has done this let me know what, if any, differences there are? Thanks.

4.4 Configuring a MS NT client

> Note: this section may be incomplete as it's been a while since I've installed PPTP on an NT system.

1. Set up your routing so that the Linux firewall is your default gateway:
    1. Open `Control Panel/Network` or right−click "Network Neighborhood" and click on `Properties`.
    2. Click on the `Protocols` tab and double−click on the "TCP/IP" line.
    3. Enter the local−network IP address of your Linux firewall in the "Default Gateway" box.
    4. Click on the "OK" button.
2. Test masquerading. For example, run "`telnet` *`my.isp.mail.server`* `smtp`" and you should see the mail server's welcome banner.
3. Install and configure the VPN software. For IPsec software follow the manufacturer's instructions. For MS PPTP:
    1. Open `Control Panel/Network` or right−click "Network Neighborhood" and click on `Properties`.
    2. Click on the `Protocols` tab.
    3. Click on the "Add" button, then double−click on the "Point−to−Point Tunneling Protocol" line.
    4. When it asks for the number of Virtual Private Networks, enter the number of PPTP servers you could possibly be communicating with.
    5. Reboot when prompted to.

6. If you need to use strong (128−bit) encryption, download the strong encryption PPTP update from the MS secure site at http://mssecure.www.conxion.com/cgi−bin/ntitar.pl and install it, then reboot again when prompted to.
7. Create a new dial−up phonebook entry for your PPTP server.
8. Select the VPN adapter as the device to use, and enter the PPTP server's internet IP address as the telephone number.
9. Select the `Server Types` tab, and check the compression and encryption checkboxes.
10. Click on the "TCP/IP Settings" button.
11. Set the dynamic/static IP address information for your client as instructed to by your PPTP server's administrator.
12. If you wish to have access to your local network while the PPTP connection is up, see MS Knowledge Base article Q143168 for a registry fix. (*Sigh.*)
13. Make sure you reapply the most recent Service Pack, to ensure that your RAS and PPTP libraries are up−to−date for security and performance enhancements.

4.5 Configuring for network−to−network routing

Yet to be written.

You really ought to look at FreeS/WAN (IPsec for Linux) at http://www.xs4all.nl/~freeswan/ instead of masquerading.

4.6 Masquerading Checkpoint SecuRemote−based VPNs

It is possible to masquerade Checkpoint SecuRemote−based VPN traffic under certain circumstances.

First, you must configure the SecuRemote firewall to allow masqueraded sessions. On the SecuRemote firewall do the following:

1. Run `fwstop`
2. Edit `$FWDIR/conf/objects.C` and after the "`:props (`" line, add or modify the following lines to read:
   ```
           :userc_NAT (true)
           :userc_IKE_NAT (true)
   ```
3. Run `fwstart`
4. Re−install your security policy.
5. Verify the change took effect by checking both `$FWDIR/conf/objects.C` and `$FWDIR/database/objects.C`

If you use the IPsec protocols (called "IKE" by CheckPoint) you don't have to do anything else special to masquerade the VPN traffic. Simply configure your masquerading gateway to masquerade IPsec traffic as described above.

Checkpoint's proprietary FWZ protocol is more complicated. There are two modes that FWZ can be used in: encapsulated mode and transport mode. In encapsulated mode, integrity checking is done over the whole IP packet, just as in IPsec's AH protocol. Changing the IP address breaks this integrity guarantee, thus encapsulated FWZ tunnels *cannot* be masqueraded.

In transport mode, only the data portion of the packet is encrypted, and the IP headers are not verified against changes. In this mode, masquerading should work with the modifications described above.

The configuration for encapsulated or transport mode is done in the FireWall−1 GUI. In the network object for the Firewall, under the VPN tab, edit the FWZ properties. The third tab in FWZ properties allows you to set encapsulated mode.

You will only be able to masquerade one client at a time.

Further information can be found at:

- http://www.phoneboy.com/fw1/nat.html,
- http://www.phoneboy.com/fw1/faq/0141.html
- http://www.phoneboy.com/fw1/faq/0372.html

---

5. Troubleshooting

5.1 Testing

To test VPN Masquerade:

1. Bring up your ISP connection from your Linux box and verify that it still works properly.
2. Verify that regular masquerading still works properly by, for example, trying to browse a Web site or access an FTP server from a masqueraded box on your local network.
3. PPTP: Verify that you have masquerading of the PPTP control channel properly configured: try to telnet from the PPTP client system to port 1723 on your PPTP server. Don't expect to see anything, but if you get a timeout or an error saying the connection failed, take a look at the masquerade rules on your Linux box to ensure that you are indeed masquerading traffic from your PPTP client to TCP port 1723 on your PPTP server.
4. PPTP: Attempt to establish a PPTP connection. I recommend you also run `RASMON` if it is available, as this will give you a minimal amount of information about the status of the connection. If you establish a PPTP connection on the first try, congratulations! You're done!
5. IPsec: Attempt to establish an IPsec connection.

5.2 Possible problems

There are several things that may prevent a VPN session from being established. We'll work through them going from the client to the server and back again. We will assume you're using a Windows−based client for the examples, as that's the most common case.

1. Connect information: the "telephone number" in the VPN dialup configuration must be the Internet IP address of the VPN server, or the IP address of the firewall if the server is being masqueraded.
2. PPTP and strong encryption: unless both client and server have the 128−bit `NDISWAN.SYS` or W'95/'98 PPTP software, you will not be able to establish a strongly−encrypted session. Unfortunately in my experience this problem does not generate any obvious error messages, it just keeps trying and trying and trying... The strong encryption update can be obtained from the Microsoft secure site URL given in the "Configuring a MS Client" section.

   This may also affect IPsec clients, if they use the MS−supplied encryption libraries rather than using their own libraries.

3. Routing: verify that the default route on your VPN client is pointing at the Linux masquerade box. Run the `route print` command and look for an `0.0.0.0` entry.

   If other masqueraded services (such as HTTP, FTP, IRC, etc.) work from your VPN client system then this probably is not the problem.

4. Masquerading: there are two parts to the VPN session.

   For IPsec, the authentication and key exchange service (ISAKMP), which is a normal UDP session to port 500 on the remote IPsec host, must be configured for masquerading as you would any other UDP service (such as DNS).

   For PPTP, the control channel, which is a normal TCP session to port 1723 on the PPTP server, must be configured for masquerading as you would any other TCP service (such as HTTP).

   The encrypted data channel in IPsec is carried over ESP, IP protocol 50. The encrypted data channel in PPTP is carried over GRE, IP protocol 47. (Note that these are *not* TCP or UDP port numbers!) Since the 2.0 Linux kernel only lets you specify `TCP`, `UDP`, `ICMP` and `ALL` IP protocols when creating masquerade rules, you must also masquerade `ALL` protocol traffic if you are masquerading only specific services. If you are masquerading everything, you don't need to worry about this.

   In order to isolate the firewall rules from the kernel masquerade code, try establishing a VPN connection with your firewall completely open, then if it works, tighten the firewall rules.

   2.0.x ipfwadm completely open firewall:

   ```
   ipfwadm -I -p accept
   ipfwadm -O -p accept
   ipfwadm -F -a accept -m
   ```

   2.2.x ipchains completely open firewall:

   ```
   ipchains -P input   ACCEPT
   ipchains -P output  ACCEPT
   ipchains -P forward MASQ
   ```

   Do *not* leave your firewall completely open for any longer than it takes to prove that a masqueraded VPN connection can be established!

5. Intermediary hops and the Internet: All routers between your Linux firewall and the remote IPsec host must forward packets carrying IP protocol 50. All routers between your Linux firewall and the PPTP server must forward packets carrying IP protocol 47. If you had IPsec or PPTP working when your VPN client system directly dialled your ISP then this probably is not the problem.

   To isolate whether an intermediary hop is blocking GRE traffic, use a patched `traceroute` to trace the progress of GRE packets. See the resources section for information on the traceroute patch. A similar patch for ESP is in the works.

6. The remote firewall: the firewall at the server end must allow a system with the IP address assigned to your Linux box by your ISP to connect to port 500/udp on the IPsec host or port 1723/tcp on the PPTP server. If you had the VPN working when your VPN client system directly dialled your ISP then this probably is not the problem.

7. The server firewall and ESP: the IPsec encrypted data is carried over IP protocol 50. If the firewall the remote IPsec host is behind does not forward ESP traffic in both directions, IPsec will not work. Again, if you had IPsec working when your IPsec client system directly dialled your ISP then this probably is not the problem.
8. The server firewall and GRE: the PPTP data channel is carried as a GRE−encapsulated (IP protocol 47) PPP session. If the firewall your PPTP server is behind does not forward GRE traffic in both directions, PPTP will not work. Again, if you had PPTP working when your PPTP client system directly dialed your ISP then this probably isn't the problem.
9. The patch: If your IPsec client successfully authenticates you but cannot establish a network connection, the patch may not be masquerading ESP traffic properly. If your PPTP client establishes the control channel (RASMON beeps and the little telephone lights up) and sends GRE traffic (the upper light in RASMON blinks) but gets no GRE traffic back (the lower light in RASMON does not blink in response) the patch may not be masquerading GRE traffic properly.

   Look in `/var/log/messages` for log entries showing that VPN traffic was seen. Turning on VPN debugging may help you to determine whether or not the patch is at fault. Also run a sniffer on your internet connection and look for outbound VPN traffic *(see below)*.

10. Multiple clients: the older PPTP patch does NOT support masquerading of multiple PPTP clients attempting to access the *same* PPTP server. If you're trying to do this, you should take a look at your network design and consider whether you should set up a PPTP router for your local clients. The 2.0 patch incorporates Call−ID masquerading, which allows multiple simultaneous sessions. *Note:* do not enable PPTP Call−ID masquerade if you are masquerading a PPTP Server. At the current time this will prevent the server's outbound traffic from being masqueraded.

5.3 Troubleshooting

Most problems can be localized by running a packet sniffer (e.g. `tcpdump` with the `−v` option) on your VPN firewall. If everything is working properly, you'll see the following traffic:

- Client local network:

  IPsec: UDP (destination UDP port 500) and ESP (IP protocol 50) traffic from your IPsec client local network IP to the remote IPsec host's Internet IP. If you don't see this, your IPsec client is misconfigured.

  PPTP: TCP (destination TCP port 1723) and GRE (IP protocol 47) traffic from your PPTP client local network IP to the PPTP server's Internet IP. If you don't see this, your PPTP client is misconfigured.

- ISP side of client firewall: UDP and ESP or TCP and GRE traffic from the client firewall Internet IP (remember − we're masquerading) to the VPN server's Internet IP. If you don't see this, your masquerade is misconfigured or the patch isn't working.
- ISP side of server firewall: UDP and ESP or TCP and GRE traffic from the client Internet IP to the VPN server's Internet IP. If you don't see this, the Internet is down `:)` or some intermediary is blocking ESP or GRE traffic.
- Boundary network (DMZ) side of server firewall: UDP and ESP or TCP and GRE traffic from the client internet IP to the server IP. If you don't see this, check your firewall rules for forwarding UDP port 500 and IP protocol 50 or TCP port 1723 and IP protocol 47, and the configuration of `ipportfw` and `ipfwd` if you're masquerading the server.

- Boundary network side of server firewall: UDP (source port 500) and ESP or TCP (source port 1723) and GRE traffic from the VPN server IP to the client internet IP. If you don't see this, check the VPN server configuration, including the packet filtering rules on the VPN server.
- ISP side of server firewall: UDP and ESP or TCP and GRE traffic from the VPN server IP (or firewall IP if the server is masqueraded) to the client internet IP. If you don't see this, check your firewall rules for forwarding UDP port 500 and IP protocol 50 or TCP port 1723 and IP protocol 47.
- ISP side of client firewall: UDP and ESP or TCP and GRE traffic from the VPN server IP to the client firewall internet IP. If you don't see this, the Internet is acting up again.
- Client local network: UDP and ESP or TCP and GRE traffic from the VPN server internet IP to the VPN client local network IP. If you see the UDP traffic but not the ESP traffic, or the TCP traffic but not the GRE traffic, the patch isn't working or wasn't properly installed.

You may find it helpful to turn on VPN debugging and recompile your kernel. Add the following to `/etc/syslog.conf`

```
# debugging
*.=debug            /var/log/debug
```

and watch `/var/log/messages` and `/var/log/debug` for log messages about the VPN traffic. Note that logging – especially verbose logging – will cause a great deal of disk activity and will cause the log files to grow very large very quickly. Don't turn on debugging unless you need to, and turn it off when you're done.

5.4 MS PPTP Clients and domain−name issues

Thanks to Charles Curley <ccurley@trib.com> for the following:

> If you use PPTP (Point to Point Tunneling Protocol) to access a Microsoft Networking (SMB) environment and have your own Microsoft Networking environment in your local private network (Samba or Windows), give your local workgroup a name that does not show up in the remote environment. The reason is that while your PPTP client is logged into the remote environment, it will see the remote environment's domain name servers, and will only see the remote computers in that workgroup.
>
> You should avoid the lazy option. Microsoft ships Windows set up for a default workgroup name of WORKGROUP. Some people will be lazy and accept that as their workgroup when they set up their computers. So there is a good chance that the remote environment will have a workgroup called WORKGROUP, administrators willing or not.

I think that this will apply regardless of the VPN in use, as name services aren't dependent on the transport. If your client(s) can see the WINS servers on the remote network then you may experience this, PPTP or no PPTP.

5.5 MS PPTP Clients and Novell IPX

If you're having trouble with IPX traffic over your PPTP link, please see sections 3.5 and 5.2 in this MS Knowledge Base article:
http://microsoft.com/ntserver/nts/downloads/recommended/dun13win95/ReleaseNotes.asp

The same considerations probably apply to Win'98 as well.

Thanks to David Griswold <dgriswol@ix.netcom.com>

5.6 MS network password issues

When you are using a VPN to access a MS network you should remember that you will have to provide two different authentication tokens – one to connect to the VPN server (the VPN password) and the other to access resources on the remote network once the connection is established (the network password).

The VPN password – the username and password you enter into your VPN client when initiating the call to the VPN server – is only used by the VPN server to grant you permission to connect to the network via the VPN. It isn't used for anything else once you're connected.

The VPN password is *not* used to prove your identity to other computers on the remote network. You must provide another username/password pair – your network password – for that.

There are two ways to supply a network password. Your network password may be the same username/password pair you supplied when logging onto the local network when you started your computer up. If it is different, you can configure your VPN client to ask you for your network password for the remote network once the VPN connection is established.

If you are successfully connecting to the VPN server but you cannot access any of the resources provided by the remote network, then you aren't providing a valid network username/password pair for the remote network. Verify that the username and password for your local network will also work on the remote network, or set your VPN client to prompt you for a username and password for use on the remote network and "log on" to the remote network once the VPN connection is established.

5.7 If your IPsec session always dies after a certain amount of time

If you're having trouble with your IPsec tunnel regularly dying, particularly if checking the system logs on the firewall shows that ISAKMP packets with "zero cookie" values are being seen, here's what's happening:

Earlier versions of the IPsec Masq patch did not change the timeout for masq table entries for ISAKMP UDP packets. The masq table entries for the ISAKMP UDP traffic would time out fairly quickly (relative to the data channel) and be removed; if the remote IPsec host then decided to initiate rekeying before the local IPsec host did, the inbound ISAKMP traffic for the rekey couldn't be routed to the masqueraded host. The rekey traffic would be discarded, the remote IPsec host would think the link had failed, and the connection would eventually be terminated.

The 2.0.x patch has been modified from its original version to increase the timeout on ISAKMP UDP masq table entries. Get the current version of the patch, available via the sites given in the Resources section, and repatch and recompile your kernel.

Also verify that your `IPsec Masq Table Lifetime` parameter is configured to be the same as or slightly longer than your rekey interval.

5.8 If VPN masquerade fails to work after you reboot

Did you remember to put `modprobe ip_masq_pptp.o` or `modprobe ip_masq_ipsec.o` commands into your `/etc/rc.d/rc.local` startup script if you compiled VPN Masq support as modules?

5.9 If your second PPTP session kills your first session

The PPTP RFC specifies that there may only be one control channel between two systems. This may mean that only one masqueraded client will be able to contact a given PPTP server at a time. See multiclientpptp for more details.

---

6. IPsec masquerade technical notes and special security considerations

6.1 Limitations and weaknesses of IPsec masquerade

Traffic that uses the AH protocol *cannot* be masqueraded. The AH protocol incorporates a cryptographic checksum across the IP addresses that the masquerade gateway cannot correctly regenerate. Thus, all masqueraded AH traffic will be discarded as having invalid checksums.

IPsec traffic using transport−mode ESP also cannot be reliably masqueraded. Transport mode ESP essentially encrypts everything after the IP header. Since, for example, the TCP and UDP checksums include the IP source and destination addresses, and the TCP/UDP checksum is within the encrypted payload and thus cannot be recalculated after the masquerade gateway alters the IP addresses, the TCP/UDP header will fail the checksum test at the remote gateway and the packet will be discarded. Protocols that do not include information about the source or destination IP addresses may successfully use masqueraded transport mode.

Apart from these limitations, IPsec masquerade is secure and reliable when only one IPsec host is being masqueraded at a time, or when each masqueraded host is communicating with a different remote host. When more than one masqueraded host is communicating with the same remote host, a few weaknesses show up:

- Transport−mode communications are subject to collisions.

  If two or more masqueraded hosts are using transport mode to communicate with the same remote host, and the security policy of the remote host permits multiple transport−mode sessions with the same peer, it is possible for sessions to experience collisions. This happens because the IP address of the *masquerading gateway* will be used to identify the sessions, and any other identifying information cannot be masqueraded because it is within the encrypted portion of the packet.

  If the remote host's security policy does not permit multiple transport−mode sessions with the same peer, the situation is even worse: the more−recently−negotiated transport mode session will likely completely take over *all* of the traffic from the older session, causing the older session to "go dead". While the established sessions from the older transport−mode IPsec session may be quickly reset if the remote host isn't expecting to receive the traffic, at least one packet of information will be sent to the wrong host. This information will probably be discarded by the recipient, but it will still be sent.

Thus, a transport−mode collision may result in leaking of information between the two sessions or termination of one or both sessions. Using IPsec in transport mode via a masquerading gateway is *not recommended* if there is the possibility that other transport mode IPsec sessions will be attempted via the same masquerading gateway to the same remote IPsec host.

IPsec using tunnel mode with extruded network addressing (where the masqueraded IPsec host is assigned an IP address from the remote host's network) is *not* subject to these problems, as the IP addresses assigned from the remote network will be used to identify the sessions instead of using the IP address of the masquerading host.

- ISAKMP communications are subject to cookie collisions.

  If two or more masqueraded hosts establishing a session to the same remote host happen to select the same initiator cookie when initiating ISAKMP traffic, the masquerading gateway will route all of the ISAKMP traffic to the second host. There is a 1 in 2^64 (i.e. very small) chance of this collision happening for each host, at the time of establishing the initial ISAKMP connection.

  Correcting this requires including the responder cookie in the key used to route inbound ISAKMP traffic. This modification is incorporated into IPsec masquerade for the 2.2.x kernel, and the short window between the time the masqueraded host initiates the ISAKMP exchange and the remote host responds is covered by discarding any new ISAKMP traffic that would collide with the current outstanding traffic. This modification will be backported to the 2.0.x code soon.
- There may be a collision between SPI values on inbound traffic.

  Two or more masqueraded IPsec hosts communicating with the same remote IPsec host may negotiate to use the same SPI value for inbound traffic. If this happens the masquerading gateway will route all of the inbound traffic to the first host to receive any inbound traffic using that SPI. The possibility of this happening is about 1 in 2^32 for each outstanding ESP session, and may occur on any rekey.

  Since the SPI values refer to different SAs having different encryption keys the first host will not be able to decrypt the data intended for the other hosts, so no data leakage will occur. There is no way for the masquerading gateway to detect or prevent this collision. The only way to prevent this collision is for the remote IPsec host to check the SPI value proposed by the masqueraded host to see if that SPI value is already in use by another SA from the same IP address. It is not likely that this will be done, since it imposes more overhead on an already expensive operation (the rekey) to benefit a small percentage of users in case of a relatively rare event.
- Inbound and outbound SPI values may be misassociated.

  This is discussed in detail in the next section.

To avoid these problems the 2.2.x code by default prevents the establishment of multiple connections to the same remote host. If the weaknesses exposed by multiple connections to the same remote host are acceptable, you can enable "parallel sessions".

Blocking parallel sessions for security reasons can be annoying: there is no way for the IPsec masquerade code to sniff the session and see when it is terminating, so the masquerade table entries will persist for the IPsec Masq Table Lifetime even if the session terminates immediately after it is established. If parallel sessions are prevented, this means that the server will be unavailable to other clients until the masq table entry for the most recent session has timed out and been deleted. This can be up to several hours.

6.2 Proper routing of inbound encrypted traffic

The portion of the ISAKMP key exchange where the ESP SPI values are communicated is encrypted, so the ESP SPI values must be determined by inspection of the actual ESP traffic. Also, the outbound ESP traffic does not contain any indication of what the inbound SPI will be. This means there is no perfectly reliable way to associate inbound ESP traffic with outbound ESP traffic.

IPsec masq attempts to associate inbound and outbound ESP traffic by serializing initial ESP traffic on a by−remote−host basis. What this means is:

- If an outbound ESP packet with an SPI value that has not previously been seen (or whose masquerade table entry has expired) is received (which shall hereafter be called an "initial packet"), a masquerade table entry for that SourceAddr+SPI+DestAddr combination is created. It is marked as "outstanding", that is, no inbound ESP traffic has been received for it yet. This is done by setting the "inbound SPI" value in the masq table entry to zero, which is a value reserved for uses such as this. This will happen at the initiation of a new ESP connection and at regular intervals when an existing ESP connection rekeys.
- As long as the masq table entry is outstanding, no other initial ESP packets for the *same remote host* will be processed. The packets are immediately discarded, and a system log entry is made saying the traffic is temporarily blocked. This also applies to initial traffic from the same masqueraded host going to the same remote host if the SPI values differ. Traffic to other remote hosts, and traffic where both SPI values are known ("established" traffic) is not affected by this.
- This could easily lead to a Denial of Service of the remote host, so this outstanding ESP masq table entry is given a short lifetime, and only a limited number of retries of the same traffic are allowed. This permits round−robin access to the remote host if several masqueraded hosts are attempting to initialize simultaneously and responses aren't coming back very quickly, for example due to network congestion or a slow remote host. The retry limitation begins once there is a collision, so the masqueraded IPsec host can wait as long as necessary for a reply until there's a need for serialization.
- When an ESP packet from the outstanding remote host is received and the SPI value does not appear in any masq table entry, it is assumed that the packet is the response to the outstanding initial packet. The SPI value is stored in that masq table entry, thus associating the SPI values, and the inbound ESP traffic is routed to the masqueraded host. At this point another initial packet for the remote server may be processed.
- Any ESP traffic with a zero SPI value is discarded as invalid, per the RFC requirements.

There are several ways this can fail to associate traffic properly:

- Network delays or a slow remote host can cause the response to the first initial packet to be delayed long enough that the init masq table entry expires and a different masqueraded host is given a chance to initialize. This could cause the response to be associated with the wrong outbound SPI, which would cause inbound traffic to be routed to the wrong masqueraded host. If this happens the masqueraded host receiving the traffic in error will discard it because it has an unexpected SPI value, and everybody will eventually time out, rekey and try again. This can be addressed by editing `/usr/src/linux/net/ipv4/ip_masq.c` (`ip_masq_ipsec.c` in 2.2.x) and increasing the

INIT lifetime or the number of INIT retries permitted, at the cost of increasing the blocking (and DoS) window.

- Sessions idle or semi−idle (with infrequent inbound traffic and no outbound traffic) for a long period of time may be idle long enough for the masq table entry to expire. If the remote host sends traffic to an established yet masq−expired session while an outstanding init to the same remote host is underway, the traffic may be misrouted for the same reason as described above. This can be addressed by making sure the `IPsec Masq Table Lifetime` kernel configuration parameter is slightly longer than the rekey interval, which is the longest time any given SPI pair should be used. The problem here is that you may not know all of the rekey intervals if you're masquerading for many remote servers, or some may have their rekey intervals set to unreasonably high values, such as several hours.

- If there is a delay between a rekey and the transmission of outbound ESP traffic using the new SPI, and during this delay inbound ESP traffic using the new SPI is received, there will be no masq table entry describing how to route the inbound traffic. If another masqueraded host has a pending init with the same remote host, the traffic will be misassociated. Note that serialization of ESP initial traffic *does not* affect ISAKMP rekey traffic.

The best solution is to have some way to preload the masq table with the properly associated out−SPI/in−SPI pair or some other mapping of remote_host + inbound_SPI to masqueraded_host. This cannot be done by inspecting the ISAKMP key exchange, as it is encrypted. It may be possible to use RSIP (a.k.a. Host−NAT) to communicate with the masqueraded IPsec host and request notification of SPI information once it has been negotiated. This is being investigated. If something is done to implement this it will be done no sooner than the 2.3.x series, as RSIP is a fairly complex client/server NAT protocol.

When an inbound ESP packet with a new SPI is received the masquerading firewall attempts to guess which masqueraded host(s) the unassociated inbound traffic is intended for. If the inbound ESP traffic is not matched to an established session or a pending session initialization, then the packet is sent to the masqueraded host(s) who most recently rekeyed with that remote host. The "incorrect" masqueraded hosts will discard the traffic as being improperly encrypted, and the "correct" host will get its data. When the "correct" host responds, the normal ESP init serialization process occurs.
