VPN HOWTO

Matthew D. Wilson

VPN HOWTO

Matthew D. Wilson

Publication date Dec 1999

Abstract

This HOWTO describes how to set up a Virtual Private Network with Linux.

Table of Contents

Chapter 1. Introduction

Why I wrote this HOWTO

I work at Real Networks, and we needed VPN service. This was my first real project, and I truly learned more about Linux with this than with any other task. I ended up using my experience with that project to write this document, to share with others what I learned, so that they can do ultra-nifty things with Linux too!

Acknowledgements and Thanks

I want to first and foremost thank my wife Julie, without her, I wouldn't be where I am today. I also want to thank Arpad Magosanyi, the author of the first VPN mini-howto and pty-redir, the utility that makes all of this possible. Jerry, Rod, Glen, Mark V., Mark W., and David, You guys rock! Thanks for all your help.

Format of this document

This document is broken down into 5 chapters.

Section 1: Introduction	This section
Section 2: Theory	Basic VPN theory. What is a VPN, and how does it work. Read this if you are entirely new to VPN.
Section 3: Server	This section describes how a VPN server is set up.
Section 4: Client	This section describes how a VPN client is set up.
Section 5: Implementation	A step by step implementation of a sample VPN setup.
Section 6: Addenda	Other bits and pieces of info that you might find helpful.

Legal Information

Copyright

Disclaimer

The author assumes no responsibility for anything done with this document, nor does he make any warranty, implied or explicit. If you break it, it's not my fault. Remember, what you do here could make very large holes in the security model of your network. You've been warned.

GNU Free Documentation License

Version 1.1, March 2000

PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of

Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from

their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Document History

The original “VPN mini-HOWTO” was written by Arpad Magosanyi, <mag@bunuel.tii.matav.hu>, in 1997. He has since allowed me to take up the document and extend it into a full HOWTO. All of this would not be possible without his original document. Thanks again Arpad. :)

Version 1.0 of this HOWTO was completed on December 10, 1999.

Related Documents

- Networking Overview HOWTO [/HOWTO/Networking-Overview-HOWTO.html]
- Networking HOWTO [/HOWTO/NET3-4-HOWTO.html]
- VPN-Masquerade HOWTO [/HOWTO/VPN-Masquerade-HOWTO.html]

Chapter 2. Theory

What is a VPN?

VPN stands for Virtual Private Network. A VPN uses the Internet as it's transport mechanism, while maintaining the security of the data on the VPN.

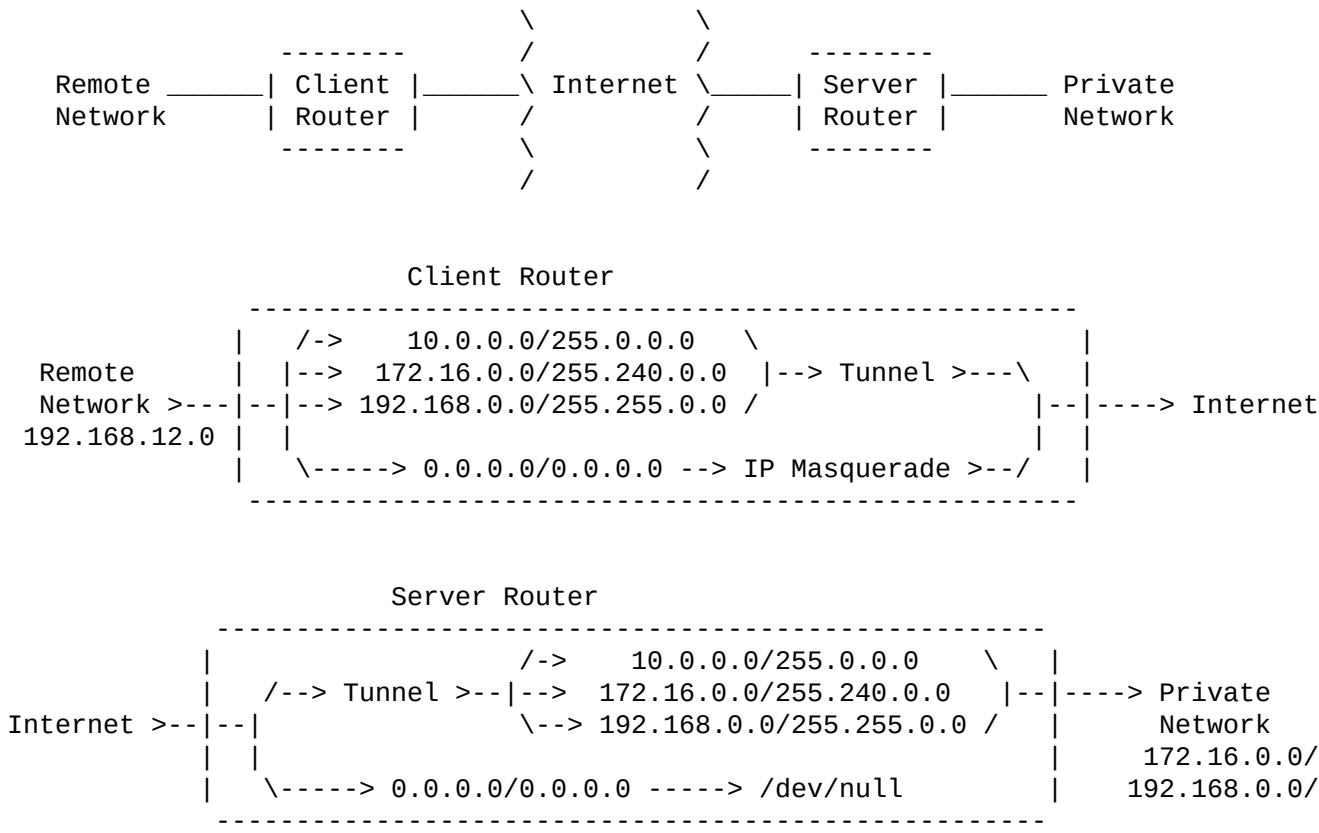
But really, what IS a VPN?

There are several answers to that question. It really depends on your network layout. The most common configuration is to have a single main internal network with remote nodes using VPN to gain full access to the central net. The remote nodes are commonly remote offices or employees working from home. You can also link two small (or large) networks to form an even larger single network.

So how does it work?

Put simply, to make a VPN, you create a secure tunnel between the two networks and route IP through it. If I've lost you already, you should read The Linux Networking Overview HOWTO [http://www.tldp.org/HOWTO/Networking-Overview-HOWTO.html] to learn more about networking with Linux.

Here are some diagrams to illustrate this concept:



The above diagram shows how the network might be set up. If you don't know what IP Masquerading is, you should probably read the The Linux Networking Overview HOWTO [/HOWTO/Networking-Overview-HOWTO.html] and come back once you understand how it works.

The Client Router is a Linux box acting as the gateway/firewall for the remote network. The remote network uses the local IP address 192.168.12.0. For the sake of a simple diagram, I left out the local routing information on the routers. The basic idea is to route traffic for all of the private networks (10.0.0.0, 172.16.0.0, and 192.168.0.0) through the tunnel. The setup shown here is one way. That is, while the remote network can see the private network, the private network cannot necessarily see the remote network. In order for that to happen, you must specify that the routes are bidirectional.

From the diagram you should also note that all of the traffic coming out of the client router appears to be from the client router, that is, all from one IP address. You could route real numbers from inside your network but that brings all sorts of security problems with it.

SSH and PPP

The system that I describe to implement VPN uses SSH and PPP. Basically I use ssh to create a tunnel connection, and then use pppd to run TCP/IP traffic though it. That's what makes up the tunnel.

The real trick to getting ssh and pppd to play well together is the utility written by Arpad Magosanyi that allows the redirection of standard in and standard out to a pseudo tty. This allows pppd to talk through ssh as if it were a serial line. On the server side, pppd is run as the users shell in the ssh session, completing the link. After that, all you need to do is the routing.

Alternative VPN Systems

There are of course other ways of setting up a VPN. Here are a couple of other systems:

PPTP

PPTP is a Microsoft protocol for VPN. It is supported under Linux, but is known to have serious security issues. I do not describe how to use it here since it is covered by the Linux VPN Masquerade HOWTO [http://www.tldp.org/HOWTO/VPN-Masquerade-HOWTO.html].

IP Sec

IP Sec is a different set of protocols from SSH. I don't actually know all that much about it, so if someone wants to help me out with a description, I'd be most appreciative. Again, I do not describe how to use it here since it is covered by the Linux VPN Masquerade HOWTO [http://www.tldp.org/HOWTO/VPN-Masquerade-HOWTO.html].

CIPE

CIPE is a kernel level network encryption system that may be better suited to enterprise setups. You can find out more about it at the CIPE homepage [http://sites.inka.de/sites/bigred/devel/cipe.html].

Chapter 3. Server

This section tells you how to set up the server side of things. I figured that this should go first since without a server, your client is kind of useless.

Security - keeping people out

Security is very important for a VPN. That's why you're building one in the first place, isn't it? You need to keep a few things in mind while setting up your server.

Trim your daemons

Since this server is going to be on both sides of your firewall, and set up to forward traffic into your network, it's a good idea to secure the box as well as you possibly can. You can read up more on Linux security in the Linux Security HOWTO [/HOWTO/Security-HOWTO.html]. In this case I killed everything but sshd and a Roxen Web server. I use the web server to download a couple of files (my scripts, etc) for setting up new machines to access the VPN. I don't use an FTP server since it's harder to configure one to be secure than it is to just make a few files available with a web server. Plus, I only need to be able to download files. If you really want to run different servers on your gateway, you might want to think about restricting access to them to only those machines on your private network.

Don't allow passwords

Yes, it sounds kind of silly, but it got your attention, didn't it? No, you don't use passwords, you disable them completely. All authentication on this machine should be done via ssh's public key authentication system. This way, only those with keys can get in, and it's pretty much impossible to remember a binary key that's 530 characters long.

So how do you do that? It requires editing the `/etc/passwd` file. The second field contains either the password hash, or alternatively 'x' telling the authentication system to look in the `/etc/shadow` file. What you do is change that field to read “*” instead. This tells the authentication system that there is no password, and that none should be allowed.

Here's how a typical /etc/passwd file looks:

```
...
nobody:x:65534:100:nobody:/dev/null:
mwilson:x:1000:100:Matthew Wilson,,,:/home/mwilson:/bin/bash
joe:*:504:101:Joe Mode (home),,,:/home/vpn-users:/usr/sbin/pppd
bill:*:504:101:Bill Smith (home),,,:/home/vpn-users:/usr/sbin/pppd
frank:*:504:101:Frank Jones (home),,,:/home/vpn-users:/usr/sbin/pppd
...
```

Note that I've done more than just editing the second field. I'll explain the other fields later on.

User Access - letting people in

User access is done via ssh's authentication scheme. As stated above, this is how users get access to the system, while maintaining a high level of security. If you're not familiar with ssh, check out http://www.ssh.org/. Note that I am using ssh version 1, not version 2. There is a big difference, notably that version 1 is free, and 2 isn't.

Configuring sshd

You'll need to configure **sshd**. The idea is to disable password authentication and rhosts authentication. The following options should be present in your `/etc/sshd_config` file.

```
PermitRootLogin yes
IgnoreRhosts yes
StrictModes yes
QuietMode no
CheckMail no
IdleTimeout 3d
X11Forwarding no
PrintMotd no
KeepAlive yes
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
UseLogin no
```

Restricting Users

Now that you're keeping the bad people out, and only letting the good people in, you may need to make sure that the good people behave themselves. This is most easily done by not letting them do anything except run pppd. This may or may not be necessary. I restrict users because the system that I maintain is dedicated to VPN, so users have no business doing anything else on it.

sudo or not sudo

There is this neat little program called sudo that allows the admin on a Unix system to grant certain users the ability to run certain programs as root. This is necessary in this case since pppd must be run as root. You'll need to use this method if you want to allow users shell access. Read up on how to setup and use sudo in the sudo man page. Using sudo is best on multi-use systems that typically host a small number of trusted users.

If you decide to not allow users to have shell access, then the best way to keep them from gaining it is to make their shell pppd. This is done in the /etc/passwd file. You can see /etc/passwd file that I did this for the last three users. The last field of the /etc/passwd file is the user's shell. You needn't do anything special to pppd in order to make it work. It gets executed as root when the user connects. This is certainly the simplest setup to be had, as well as the most secure, and ideal for large scale and corporate systems. I describe exactly what all needs to be done later in this document. You can the section called “Server: Set up user accounts” if you like.

Networking

Now that your users have access to the system, we need to make sure that they have access to the network. We do that by using the Linux kernel's firewalling rules and routing tables. Using the **route** and **ipfwadm** commands, we can set up the kernel to handle network traffic in the appropriate ways. For more info on **ipfwadm**, **ipchains** and **route** see the Linux Networking HOWTO [http://www.tldp.org/HOWTO/Linux-Networking-HOWTO.html].

The Kernel

In order for any of this to work, you must have your kernel configured correctly. If you don't know how to build your own kernel, then you should read the Kernel HOWTO [http://www.tldp.org/HOWTO/Kernel-HOWTO.html]. You'll need to make sure that the following kernel options are turned on in addition to basic networking. I use a 2.0.38 kernel in my system.

For 2.0 kernels:

- CONFIG_FIREWALL
- CONFIG_IP_FORWARD
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_IP_MASQUERADE (optional)
- CONFIG_IP_MASQUERADE_ICMP (optional)
- CONFIG_PPP

For 2.2 kernels:

- CONFIG_FIREWALL
- CONFIG_IP_ADVANCED_ROUTER
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_IP_MASQUERADE (optional)
- CONFIG_IP_MASQUERADE_ICMP (optional)
- CONFIG_PPP

Filter Rules

First, we write firewall filter rules that allow our users to access our internal nets, while restricting them from accessing the outside internet. This sounds strange, but since the users already have access to the internet, why let them use the tunnel to access the net? It wastes both bandwidth and processor resources.

The filter rules that we use depend upon which internal nets we use, but translate to: "Allow traffic coming from our VPNs that is destined for our internal nets to go there." So how do we do that? As always, it depends. If you are running a 2.0 kernel, you use the tool called **ipfwadm**, but if you are using a 2.2 kernel, you use the utility called **ipchains**.

To set the rules with **ipfwadm**, run it with options similar to the following:

```
# /sbin/ipfwadm -F -f
# /sbin/ipfwadm -F -p deny
# /sbin/ipfwadm -F -a accept -S 192.168.13.0/24 -D 172.16.0.0/12
```

To set the rules with **ipchains**, run it with options similar to the following:

```
# /sbin/ipchains -F forward
# /sbin/ipchains -P forward DENY
# /sbin/ipchains -A forward -j ACCEPT -s 192.168.13.0/24 -d 172.16.0.0/12
```

For those using 2.2 kernels, please read the section called “IPv4 Forwarding and 2.2 kernels”.

Routing

Now that users are allowed to access our nets, we need to tell the kernel where to send the packets. On my system, I have two ethernet cards, one is on the external network, while the other is on the internal network. This helps keep things secure, as outbound traffic is masqueraded by the gateway, and any incoming traffic is filtered and routed by the Cisco Router. For most setups, the routing should be simple.

Next, route all traffic destined for the private networks out the internal interface, and all other traffic out the external interface. The specific routing commands depend on which internal nets you are using. Below is an example of what they might look like. These lines are of course in addition to your basic routes for your local nets. I also doubt that you are using all 3 groups of internal numbers:

```
Assuming that 172.16.254.254 is the internal gateway:

# /sbin/route add -net 10.0.0.0 netmask 255.0.0.0 gw 172.16.254.254 dev eth1
# /sbin/route add -net 172.16.0.0 netmask 255.240.0.0 gw 172.16.254.254 dev eth1
# /sbin/route add -net 192.168.0.0 netmask 255.255.0.0 gw 172.16.254.254 dev eth1
```

One additional note on routing. If you are using two way routing for say, a remote office, then you will need to do one more thing. You need to set up routes on the server that point back to the client. The easiest way to accomplish this is to run a cron job every minute that quietly sets back routes. If the client is not connected, **route** will just spit out an error (that you've conveniently sent to `/dev/null`.)

Chapter 4. Client

Now we examine the client end. In practice, when used to allow access to a remote network, this box can easily serve as a Samba (Windows Networking) server, DHCP server, and even an internal web server. The important thing to remember is that this box should be as secure as possible, as it runs your whole remote network.

The Kernel

First things first, you must have ppp available in your kernel. If you are going to allow multiple machines to use the tunnel, then you need to have firewalling and forwarding available too. If the client is going to be a single machine, ppp is enough.

Bring up the link

The link is created by running **pppd** through a pseudo terminal that is created by **pty-redir** and connected to **ssh**. This is done with something similar to the following sequence of commands:

```
# /usr/sbin/pty-redir /usr/bin/ssh -t -e none -o 'Batchmode yes' -c blowfish -i /r
# sleep 10

# /usr/sbin/pppd `cat /tmp/vpn-device`
# sleep 15

# /sbin/route add -net 172.16.0.0 gw vpn-internal.mycompany.com netmask 255.240.0.
# /sbin/route add -net 192.168.0.0 gw vpn-internal.mycompany.com netmask 255.255.0
```

What this does is run ssh, redirecting the input and output to pppd. The options passed to ssh configure it to run without escape characters (-e), using the blowfish crypto algorithm (-c), using the identity file specified (-i), in terminal mode (-t), with the options 'Batchmode yes' (-o). The sleep commands are used to space out the executions of the commands so that each can complete their startup before the next is run.

Scripting

If you don't want to have to type those commands in every time that you want to get the tunnel running, I've written a set of bash scripts that keep the tunnel up and running. You can download the package from here [http://www.shinythings.com/vpnd/vpnd.tar.gz]. Just download and uncompress it into /usr/local/vpn. Inside you'll find three files:

- vpnd: The script that controls the tunnel connection.
- check-vpnd: a script to be run by cron to check that vpnd is still up.
- pty-redir: a small executable needed to initialize the tunnel.

You'll need to edit the **vpnd** script to set things like the client's username and the server's names. You may also need to modify the starttunnel section of the script to specify which networks you are using. Below is a copy of the script for your reading enjoyment. You'll note that you could put the script in a different directory, you just need to change the VPN_DIR variable.

```
#! /bin/bash
#
# vpnd: Monitor the tunnel, bring it up and down as necessary
#

USERNAME=vpn-username
IDENTITY=/root/.ssh/identity.vpn

VPN_DIR=/usr/local/vpn
LOCK_DIR=/var/run
VPN_EXTERNAL=vpn.mycompany.com
VPN_INTERNAL=vpn-internal.mycompany.com
PTY_REDIR=${VPN_DIR}/pty-redir
SSH=${VPN_DIR}/${VPN_EXTERNAL}
PPPD=/usr/sbin/pppd
ROUTE=/sbin/route
CRYPTO=blowfish
PPP_OPTIONS="noipdefault ipcp-accept-local ipcp-accept-remote local noauth nocrtsc
ORIG_SSH=/usr/bin/ssh


starttunnel () {
   $PTY_REDIR $SSH -t -e none -o 'Batchmode yes' -c $CRYPTO -i $IDENTITY -l $USERN
   sleep 15

   $PPPD `cat /tmp/vpn-device` $PPP_OPTIONS
   sleep 15

   # Add routes (modify these lines as necessary)
   /sbin/route add -net 10.0.0.0 gw $VPN_INTERNAL netmask 255.0.0.0
   /sbin/route add -net 172.16.0.0 gw $VPN_INTERNAL netmask 255.240.0.0
   /sbin/route add -net 192.168.0.0 gw $VPN_INTERNAL netmask 255.255.0.0
}

stoptunnel () {
   kill `ps ax | grep $SSH | grep -v grep | awk '{print $1}'`
}

resettunnel () {
   echo "reseting tunnel."
   date >> ${VPN_DIR}/restart.log
   eval stoptunnel
   sleep 5
   eval starttunnel
}

checktunnel () {
   ping -c 4 $VPN_EXTERNAL 2>/dev/null 1>/dev/null

   if [ $? -eq 0 ]; then
      ping -c 4 $VPN_INTERNAL 2>/dev/null 1>/dev/null
      if [ $? -ne 0 ]; then
         eval resettunnel
      fi
```

```
   fi
}

settraps () {
   trap "eval stoptunnel; exit 0" INT TERM
   trap "eval resettunnel" HUP
   trap "eval checktunnel" USR1
}

runchecks () {
   if [ -f ${LOCK_DIR}/tunnel.pid ]; then
      OLD_PID=`cat ${LOCK_DIR}/vpnd.pid`
      if [ -d /proc/${OLD_PID} ]; then
         echo "vpnd is already running on process ${OLD_PID}."
         exit 1
      else
         echo "removing stale pid file."
         rm -rf ${LOCK_DIR}/vpnd.pid
         echo $$ > ${LOCK_DIR}/vpnd.pid
         echo "checking tunnel state."
         eval checktunnel
      fi
   else
      echo $$ > ${LOCK_DIR}/vpnd.pid
      eval starttunnel
   fi
}

case $1 in
    check)  if [ -d /proc/`cat ${LOCK_DIR}/vpnd.pid` ]; then
               kill -USR1 `cat ${LOCK_DIR}/vpnd.pid`
               exit 0
            else
               echo "vpnd is not running."
               exit 1
            fi ;;

    reset)  if [ -d /proc/`cat ${LOCK_DIR}/vpnd.pid` ]; then
               kill -HUP `cat ${LOCK_DIR}/vpnd.pid`
               exit 0
            else
               echo "vpnd is not running."
               exit 1
            fi ;;

   --help | -h)
            echo "Usage: vpnd [ check | reset ]"
            echo "Options:"
            echo "     check    Sends running vpnd a USR1 signal, telling it to ch
            echo "              the tunnel state, and restart if neccesary."
            echo "     reset    Sends running vpnd a HUP signal, telling it to res
            echo "              it's tunnel connection." ;;
esac
```

```
ln -sf $ORIG_SSH $SSH
settraps
runchecks

while true; do
   i=0
   while [ $i -lt 600 ]; do
      i=((i+1))
      sleep 1
   done
   eval checktunnel
done
```

LRP - Linux Router Project

I actually run this setup on Pentium 90's running the LRP distribution of Linux. LRP is a distribution of Linux that fits in, and boots off of a single floppy disk. You can learn more about it at http://www.linuxrouter.org/ You can download my LRP package for the VPN client from here [http://www.shinythings.com/vpnd/vpnd.lrp]. You will also need both the ppp and ssh packages from the LRP site.

Chapter 5. Implementation

In this section, I explain step by step how to set up your VPN system. I'll start with the server, and then move on to the client. For the purposes of an example, I will invent a situation that would require a couple of different kinds of VPN set up.

Planning

Let's imagine that we have a company, called mycompany.com. At our head office, we are using the 192.168.0.0 reserved network, breaking the class B into 256 class C networks to allow routing. We have just set up two small remote offices, and want to add them to our network. We also want to allow employees who work from home to be able to use their DSL and cable modem connections instead of making them use dialup. To start, we need to plan things out a little.

I decide that I want to give each remote office a class C network range to allow them to expand as necessary. So, I reserve the 192.168.10.0 and 192.168.11.0 nets. I also decide that for home users, I've got enough numbers that I don't need to masquerade them on the VPN server side. Each client gets it's own internal IP. So, I need to reserve another class C for that, say 192.168.40.0. The only thing that I must now do is to add these ranges to my router. Let's imagine that our company owns a small Cisco (192.168.254.254) that handles all of the traffic through our OC1. Just set routes on the Cisco such that traffic headed to these reserved nets goes to our VPN server (192.168.40.254). I put the VPN server into the home user's net for reasons that should become clear later. We'll name the external interface of the server vpn.mycompany.com, and the internal vpn-internal.mycompany.com.

As for external numbers, we don't need to know them explicitly. You should have your own numbers, supplied by your ISP.

Gather the tools

We will need a few pieces of software. Get the following packages, and install them where specified.

For the Server:

- pppd (version 2.3 or greater)
- ssh (version 1.2.26 or better)

For the Client:

- pppd (same version as server)
- ssh
- pty-redir [ftp://ftp.vein.hu/ssa/contrib/mag/pty-redir-0.1.tar.gz]

Server: Build the kernel

To start, you probably need to rebuild your kernel for the server. You need to make sure that the following kernel options are turned on in addition to basic networking and everything else that you might need. If you've never built your own kernel before, read the Kernel HOWTO [/HOWTO/Kernel-HOWTO.html].

For 2.0 kernels:

- CONFIG_FIREWALL
- CONFIG_IP_FORWARD
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_PPP

For 2.2 kernels:

- CONFIG_FIREWALL
- CONFIG_IP_ADVANCED_ROUTER
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_PPP

Server: Configure Networking

If you are building a server that has only one network card, I suggest that you think about buying another, and rewiring your network. The best way to keep your network private is to keep it on it's own wires. So if you do have two network cards, you'll need to know how to configure both of them. We'll use eth0 for the external interface, and eth1 for the internal interface.

Configuring the interfaces

We first should configure the external interface of the server. You should already know how to do this, and probably already have it done. If you don't, then do so now. If you don't know how, go back and read the Networking HOWTO [/HOWTO/NET3-4-HOWTO.html]

Now we bring up the internal interface. According to the numbers that we've chosen, the internal interface of the server is 192.168.40.254. so we have to configure that interface.

For 2.0 kernels, use the following:

```
# /sbin/ifconfig eth1 192.168.40.254 netmask 255.255.255.0 broadcast 192.168.40.25
# /sbin/route add -net 192.168.40.0 netmask 255.255.255.0 dev eth1
```

For 2.2 kernels, use the following:

```
# /sbin/ifconfig eth1 192.168.40.254 netmask 255.255.255.0 broadcast 192.168.40.25
```

That gets our basic interfaces up. You can now talk to machines on both local networks that are attached to the server.

Setting routes

We can now talk to machines on our local nets, but we can't get to the rest of our internal network. That requires a few more lines of code. In order to reach the other machines on other subnets, we need have a route that tells traffic to go to the Cisco router. Here's that line:

```
# /sbin/route add -net 192.168.0.0 gw 192.168.254.254 netmask 255.255.0.0 dev eth1
```

That line tells the kernel that any traffic destined for the 192.168.0.0 network should go out eth1, and that it should be handed off to the Cisco. Traffic for our local net still gets where it is supposed to because the routing tables are ordered by the size of the netmask. If we were to have other internal nets in our network, we would have a line like the above for each net.

Making filter rules

Now that we can reach every machine that we could need to, we need to write the firewall filtering rules that allow or deny access through the VPN server.

To set the rules with **ipfwadm**, run it like so:

```
# /sbin/ipfwadm -F -f
# /sbin/ipfwadm -F -p deny
# /sbin/ipfwadm -F -a accept -S 192.168.40.0/24 -D 192.168.0.0/16
# /sbin/ipfwadm -F -a accept -b -S 192.168.10.0/24 -D 192.168.0.0/16
# /sbin/ipfwadm -F -a accept -b -S 192.168.11.0/24 -D 192.168.0.0/16
```

To set the rules with **ipchains**, run it like so:

```
# /sbin/ipchains -F forward
# /sbin/ipchains -P forward DENY
# /sbin/ipchains -A forward -j ACCEPT -s 192.168.40.0/24 -d 192.168.0.0/16
# /sbin/ipchains -A forward -j ACCEPT -b -s 192.168.10.0/24 -d 192.168.0.0/16
# /sbin/ipchains -A forward -j ACCEPT -b -s 192.168.11.0/24 -d 192.168.0.0/16
```

This tells the kernel to deny all traffic except for the traffic that is coming from the 192.168.40.0/24 network and destined for the 192.168.0.0/16 network. It also tells the kernel that traffic going between the 192.168.10.0/24 and 192.168.0.0/16 nets is allowed, and the same for the 192.168.11.0 net. These last two are bidirectional rules, this is important for getting the routing to work going both ways.

Routing

For home users, everything will work fine to here. However for the remote offices, we need to do some routing. First of all, we need to tell the main router, or Cisco, that the remote offices are behind the VPN server. So specify routes on the Cisco that tell it to send traffic destined for the remote offices to the VPN server. Now that is taken care of, we must tell the VPN server what to do with the traffic destined for the remote offices. To do this, we run the **route** command on the server. The only problem is that in order for the **route** command to work, the link must be up, and if it goes down, the route will be lost. The solution is to add the routes when the clients connects, or more simply, to run the route command frequently as it's not a problem to run it more than is necessary. So, create a script and add it to your crontab to be run every few minutes, in the script, put the following:

```
/sbin/route add -net 192.168.11.0 gw 192.168.10.253 netmask 255.255.255.0
/sbin/route add -net 192.168.10.0 gw 192.168.11.253 netmask 255.255.255.0
```

Server: Configure pppd

Now we will configure pppd on the server to handle VPN connections. If you are already using this server to handle dialup users or even dialing out yourself, then you should note that these changes may affect those services. I go over how to avoid conflicts at the end of this section.

/etc/ppp/

This directory may contain a number of files. You probably already have a file called `options`. This file holds all of the global options for **pppd**. These options cannot be overridden by **pppd** on the command line.

/etc/ppp/options

Your `options` file should contain at least the following:

```
ipcp-accept-local
ipcp-accept-remote
proxyarp
noauth
```

The first two lines tell **pppd** to accept what the other end specifies for IP addresses. This is necessary when hooking up remote offices, but can be disabled if you are only connecting home users. It's okay to leave it on, as it does not prevent the server from assigning addresses, it only says it that it's okay to accept what the client asks for.

The third line is very important. From the **pppd** man page:

```
proxyarp
       Add an entry to this system's ARP [Address  Resolu-
       tion  Protocol]  table  with  the IP address of the
       peer and the Ethernet address of this system.  This
       will  have  the effect of making the peer appear to
       other systems to be on the local ethernet.
```

This is important because if it is not done, local traffic will not be able to get back through the tunnel.

The last line is just as important. This tells **pppd** to allow connections without username and password. This is safe since authentication is already handled by **sshd**.

Avoiding conflicts

If you are handling other services with **pppd**, you should consider that the configurations for these other services may not be the same as what the VPN system needs. **pppd** is designed such that the options in the main options file `/etc/ppp/options` cannot be overridden by options specified at runtime. This is done for security reasons. In order to avoid conflict, determine which options cause the conflict, and

move them from the main file into a separate options file that is loaded when the appropriate application of `pppd` is run.

Server: Configure sshd

The following is what my `/etc/sshd_config` file looks like. Yours should look the same or similar:

```
# This is the ssh server system wide configuration file.

Port 22
ListenAddress 0.0.0.0
HostKey /etc/ssh_host_key
RandomSeed /etc/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts yes
StrictModes yes
QuietMode no
FascistLogging yes
CheckMail no
IdleTimeout 3d
X11Forwarding no
PrintMotd no
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
UseLogin no
```

The important points to note are that password authentication is disabled as are all of the “R” services. I have also turned off mail checking and the message of the day as they can confuse **pppd** on the client side. I still allow root login, but as this can only be done with a key, it is adequately safe.

Server: Set up user accounts

Now we'll set up the user accounts.

Add `vpn-users` group

just run:

```
# /usr/sbin/groupadd vpn-users
```

Now cat the `/etc/group` file and look at the last line. It should be the entry for the vpn-users group. Note the third field. This is the group ID (GID). Write it down, as we'll need it in a minute. For this example, the GID is 101.

create the vpn-users home directory

We're going to use a single home directory for all of the users. So just run:

```
# mkdir /home/vpn-users
```

The .ssh directory

Now create the `.ssh` directory in the `vpn-users` home directory.

```
# mkdir /home/vpn-users/.ssh
```

Adding users

Now comes the fun part. We're going to edit the `/etc/passwd` file by hand. Normally you let the system handle this file, but for an unusual setup like this, it is easier to do it yourself. To start, open the `/etc/passwd` file and see what's in there. Here's an example of what you might find:

```
...
nobody:x:65534:100:nobody:/dev/null:
mwilson:x:1000:100:Matthew Wilson,,,:/home/mwilson:/bin/bash
joe:*:1020:101:Joe Mode (home),,,:/home/vpn-users:/usr/sbin/pppd
bill:*:1020:101:Bill Smith (home),,,:/home/vpn-users:/usr/sbin/pppd
frank:*:1020:101:Frank Jones (home),,,:/home/vpn-users:/usr/sbin/pppd
...
```

You'll find the first user on most any system. The second one is me. After that are a few made up vpn-users. The first field is the username, and the second is the password field. The third is user ID (UID) and the fourth is the group ID (GID). After that comes some info on who the people are in the fifth field. The sixth field is the user's home directory, and the last is their shell. As you can see, each field is separated by a colon. Look at the last three lines. The only difference between them is the username in the first field, and the user info in the fifth field. What we want to do is create lines like this for each user. Don't just use one user for all of the connections, you'll never be able to tell them apart if you do. So copy the last line of this file and edit it so that it looks something like the above. Make sure that the second field has an asterisk (*). The second field should be unique to all the other IDs in the file. I used 1020. You should use a number above 1000, since those below are typically reserved for system use. The fourth field should be the group ID for vpn-users. I told you to write it down, now is the time that you need it. So put the group ID in there. Lastly, change the home directory to `/home/vpn-users`, and the shell to `/usr/sbin/pppd`. Now copy that line to make more users. Just edit the first the fifth fields and you're set.

Server: Administration

One of the advantages to using this system for user accounts is that you can take advantage of the UNIX user administration commands. Since each client is logged in as a user, you can use standard methods to get user statistics. The following are a few commands that I like to use to see what all is going on.

who — Prints the users currently logged in, as well as when they logged in, from where (name or IP), and on which port.

w	This command prints a more extensive listing of who is currently logged in. It also tells you uptime and load averages for the system. It also lists the user's current process (which should be -pppd for VPN clients) as well as idle time, and current CPU usage for all processes as well as the current process. Read the `w` man page for more info.
last [username]	This lists the login history for the specified user, or for all users if a username is not provided. It's most useful for finding out how well the tunnels are running as it prints the length of time that the user was logged in, or states that the user is still logged in. I should warn you that on a system that has been up a long time, this list can grow extremely long. Pipe is through `grep` or `head` to find out exactly what you want to know.

You can also control which users are allowed to connect by modifying the `/home/vpn-users/.ssh/authorized_keys` file. If you remove the user's public key line from this file, they won't be able to log in.

Client: Build the kernel

Now we move onto the client. First we must rebuild the kernel so that it can support all of the functions that we need. The minimum requirement is to have ppp in the kernel. You will need forwarding, a firewall, and a gateway only if you are going to allow other machines access to the tunnel. For this example, I will setup one of the remote office machines in my example layout. Add the following options to your kernel. Again, if you've never built a kernel before, read the Kernel HOWTO [/HOWTO/Kernel-HOWTO.html].

For 2.0 kernels:

- CONFIG_PPP
- CONFIG_FIREWALL
- CONFIG_IP_FORWARD
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_IP_MASQUERADE
- CONFIG_IP_MASQUERADE_ICMP

For 2.2 kernels:

- CONFIG_PPP
- CONFIG_FIREWALL
- CONFIG_IP_ADVANCED_ROUTER
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_IP_MASQUERADE
- CONFIG_IP_MASQUERADE_ICMP

Client: Configure Networking

Now we should setup the networking on our client box. Let's assume that we've configured the external network and that it works. Now we will configure the internal interface of the client to service our intranet.

Interface

We need to first bring up the internal network interface. To do this, add the following to your `/etc/rc.d/rc.inet1` (or equivalent) file:

For 2.0 Kernels:

```
/sbin/ifconfig eth1 192.168.10.253 broadcast 192.168.10.255 netmask 255.255.255.0
/sbin/route add -net 192.168.10.0 netmask 255.255.255.0 dev eth1
```

For 2.2 Kernels:

```
/sbin/ifconfig eth1 192.168.10.253 broadcast 192.168.10.255 netmask 255.255.255.0
```

Filter rules

To set up the remote office, we will want to set up our filter rules that allow traffic to go both directions through the tunnel. Add the following lines to your `/etc/rc.d/rc.inet1` (or equivalent) file:

For 2.0 kernels:

```
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p deny
/sbin/ipfwadm -F -a accept -b -S 192.168.10.0/24 -D 192.168.0.0/16
```

For 2.2 kernels:

```
/sbin/ipchains -F forward
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -j ACCEPT -b -s 192.168.10.0/24 -d 192.168.0.0/16
```

You may have noticed that these lines look like what we have on the server. That's because they are the same. These rules just say where traffic is allowed to go between these two networks.

Routing

The only extra routes that are needed are created by the script that bring the tunnel up.

Client: Configure pppd

You may not need to edit the client's `/etc/ppp/options` file at all. You will if the "auth" option is present, or some of the other priveledged options. Try it, and if it fails, a black `/etc/ppp/options` will work. just keep adding the options from the old file to figure out which one broke it (if it's not obvious) and see if you can get around that. Maybe you don't need them at all. You probably don't if you don't use **pppd** for anything else.

Client: Configure ssh

As root on the client, run the following lines:

```
# mkdir /root/.ssh
# ssh-keygen -f /root/.ssh/identity.vpn -P ""
```

This will create two files, `identity.vpn` and `identity.vpn.pub` in the `.ssh` directory. The first is your private key, and should be kept such. *Never send this over the net* unless it is via an encrypted session. The second file is your public key, and you can send this anywhere you want, it only serves to allow you access to other systems, and cannot be used to get into your own. It is a text file with one line in it that is your actual key. At the end of the line is the comment field which you may change without fear of breaking the key. an example key looks something like this:

```
1024 35 143072373667416261958831427516 7.....250872101150654839 root@vpn-client.m
```

It's actually a lot longer than that, but it wouldn't fit on the page if I showed the whole thing. Copy your key into the `/home/vpn-users/.ssh/authorized_keys` file on the server. Make sure that there is only one key per line, and that each key is not broken onto multiple lines. You may alter the comment field all that you like in order to help you remember which line goes with which user. I highly recommend doing so.

Client: Bring up the connection

Now we'll try to actually make the connection to the VPN server. First we'll need to make a single connection to set up the **ssh** known_hosts file. Run this:

```
# ssh vpn.mycompany.com
```

Answer “yes” when it asks you if you want to continue connecting. The server will tell you “permission denied”, but that's okay. It's important that you use the same name for the server that you are using in your connection scripts. Now run the following lines. You will obviously need to change the options to suit your setup.

```
# /usr/sbin/pty-redir /usr/bin/ssh -t -e none -o 'Batchmode yes' -c blowfish -i /r

 (now wait about 10 seconds)

# /usr/sbin/pppd `cat /tmp/vpn-device` 192.168.10.254:192.168.40.254
```

Note the IP addresses specified on the **pppd** line. The first is the address of the client end of the tunnel. The second is the address of the server end of the tunnel, which is set to the server's internal address. If all of that seemed to work, move on. If not, check that you have all of the options, and that they are spelled right. If something is still going wrong, check the section called “Pitfalls”.

Client: Set the routes

Now set the route to send traffic through the tunnel. Just run this:

```
# /sbin/route add -net 192.168.0.0 gw vpn-internal.mycompany.com netmask 255.255.0
```

You should now be able to communicate with machines on the other side of the tunnel. Give it a try. If it doesn't work, try using **ping** and **traceroute** to figure out where your problem might be. If in fact it does work, move on to setting up scripts to do the work for you.

Client: Scripting

Use the vpnd script. Only you may need to modify it a little. Make the following changes:

- Change the variables at the top to match your setup. Most should be just fine as they are, but you can change them should you need to.
- Line 27: add the local and remote IP addresses before $PPP_OPTIONS
- Line 31: Change this line, and the two after it to set routes for your internal nets.

Keeping it running

While bash scripts are generally stable, they have been known to fail. In order to make sure that the **vpnd** script keeps running, add an entry to the client's crontab that runs the **check-vpnd** script. I run mine every 5 minutes or so. If **vpnd** is indeed running, **check-vpnd** doesn't use much CPU.

Chapter 6. Addenda

Pitfalls

Here are just a few of the snags that I've run into while using this system. I put them here so that you can hopefully avoid them. If you run into any new ones, please email them to me [mailto:matthew@shinythings.com] so that I can keep track, and help others avoid them.

read: I/O error

This error is associated with mis-matched versions off pppd. If you get it, try upgrading both ends of the connection to the latest version of pppd. I've found that pppd version 2.2 has this problem, so use version 2.3.7 or 2.3.8 instead.

SIOCADDRT: Network is unreachable

This error is generated by **route**. I've seen it happen when the sleep time between **ssh** and **ppd** is not long enough. If you get this error, run **ifconfig**, and you may see that there is no pppX interface. This means that **ssh** was not done authenticating before **pppd** was launched, and therefore **pppd** did not make the connection. just increase the delay, and your problems will be solved.

I wonder however if there might be some pppd option that will fix this problem.

IPv4 Forwarding and 2.2 kernels

In the new 2.2 kernel, you must specifically enable IP forwarding in the kernel at boot up. This with the following command:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Without this, the kernel will not forward any packets, and hence the server will not work, nor will any of the gatewaying clients.

Routing

It should go without saying, but be careful when you are routing real numbers that you don't route traffic destined for the VPN server's external address through the tunnel. It won't make it. (yes, this *is* from personal experience.)

Hardware and Software Requirements

Minimum Hardware Requirements

This system has been run on a 486SX33 with 8 megabytes of RAM. It didn't run very well though, it had trouble handling heavy traffic.

It doesn't take much more to make it work though. This system does work very well on a Pentium 75 with 16 megs of RAM, using an LRP distribution running off of a floppy, with a 6 meg ramdisk, and 10 megs of main memory. I've tested this setup by running a 700kbit RealVideo stream through it for over an hour.

I now typically run it on Pentium 90s, as their PCI clocking plays nicer with cheap 100Mbit Ethernet cards.

Software Requirements

This system works with both the 2.0 and 2.2 kernels. The script to keep the tunnel up requires a reasonably modern bash. I have however noticed that certain distribution's versions of bash don't play too well with the script.

Also, if someone could help me refine my scripts (or even write an executable) that would help a lot. I'm not sure why, but even my own bash doesn't follow the rules and doesn't seem to interpret signals correctly. If you do make any improvements, please email me at matthew@shinythings.com [mailto:matthew@shinythings.com].