paxi , Fotolia

Configuring VPN connections with Linux clients

CLOSE AND SECRET

Linux clients sometimes need a little help to connect to Windows VPN servers. **BY JAMES STANGER**

Two benefits of tunneling are encrypted connections and access to resources behind the firewall. When it comes to interoperability, however, establishing these connections is sometimes difficult for Linux clients. Linux distributions often have issues with establishing virtual private network (VPN) connections with servers based in other environments, mainly because the GUI applications used to establish those connections have trouble staying in sync with the pace of Linux development. It is often two steps forward, and once step back: When each distribution ships, the shared libraries often get changed, and your favorite VPN application that used to work no longer succeeds with cross-platform connections. Recently the situation has improved. In this article, I look at some tips for establishing VPN connections from the Linux desktop.

A VPN creates a point-to-point tunnel over a public network. A number of protocols support VPN connections, including the following popular options:

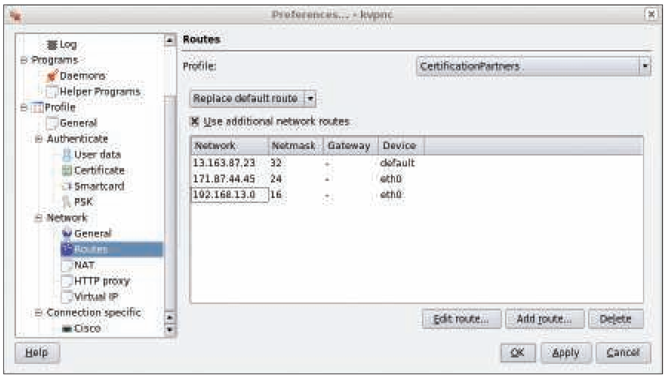
- L2TP over IPsec – Cisco's primary tunneling protocol. L2TPv3 is the latest version, but make sure you choose a version appropriate for your network. Remember that two major implementations of IPsec are available in Linux systems. For example, older systems use FreeS/WAN or Openswan for IPsec. Newer systems with any version of the standard 2.6 kernel have native IPsec support.
- Point-to-Point Tunneling Protocol (PPTP) – An older protocol that still is used in many Microsoft environments.
- Secure Sockets Layer/Transport Layer Security (SSL/TLS) – One of the most powerful interoperability protocols available SSL/TLS supports many types of VPN connections. OpenVPN [1], for example, is an SSL/TLS-based tunneling solution.

Over the years, I've followed many discussions about which protocol is the most secure or open source–friendly. Many of these discussions border on the quasi-religious. As I've grown older, I've discovered that such discussions are less useful than simply finding out which protocol works best for your particular network and adopting it.

Whatever protocol you use to establish your tunnel, it's possible to place packets through this tunnel, and those packets will be regarded by the firewall as internal packets. The key, as you will see below, is making sure the right packets go through the right interface and tunnel.

The Linux environment provides several tools for configuring and managing VPN connections. KDE's KVpnc utility supports various VPN techniques (from Cisco, to Microsoft, to OpenVPN). Vpnc is a command-line VPN client for Cisco systems. Many Linux distributions use Red Hat's NetworkManager [2], which allows you to add VPN plugins for Cisco and Microsoft networks. For the VPN you want to use, you will have to install the appropriate NetworkManager plugin.

Pptpconfig [3] is an older client that works especially well for many Red Hat



Routes

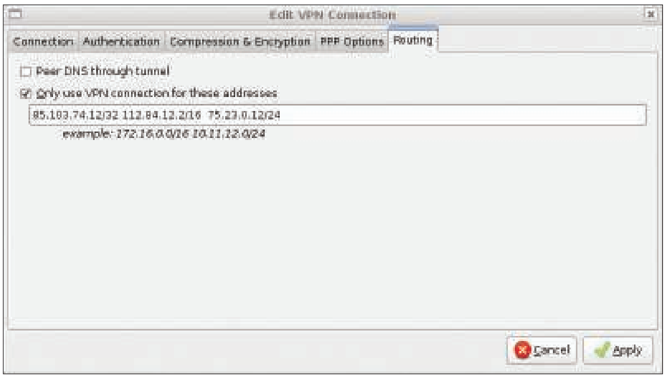
Profile: CertificationPartners

Replace default route

☒ Use additional network routes

Network	Netmask	Gateway	Device
13.163.87.23	32	-	default
171.87.44.45	24	-	eth0
102.168.13.0	16	-	eth0

Figure 1: Adding routes in a KVpnc VPN client.



☐ Peer DNS through tunnel

☑ Only use VPN connection for these addresses

85.103.74.12/32 112.84.12.2/16 75.23.0.12/24

example: 172.16.0.0/16 10.11.12.0/24

Figure 2: Adding routes in the NetworkManager client.

and Novell Linux implementations, along with BSD and various others. As the name suggests, pptpconfig is used for managing Microsoft-based VPNs.

Creating a Tunnel

Regardless of which client or protocol you use, you'll need to gather some basic information to establish a connection. For instance:

- Authentication information – Depending on what your network administrator wants, you'll have to provide user-specific information to authenticate yourself to the VPN server. The least secure (yet most common) option remains providing a username and password. Additional options include secret keys and certificates. Cisco, for example, prefers keys or certificates.
- Gateway (also known as the VPN server name) – The IP address or hostname of the VPN device that provides the tunnel once you authenticate.
- Protocol type – As discussed above, you will need to choose between protocols such as L2TP, PPTP, OpenVPN, and so forth. The configuration tool will ask you to specify options specific to the protocol(s) your network administrator has chosen for the VPN.
- Universal settings – With any authentication protocol, you'll need to provide information specific to your connection. For example, some networks require you to set a specific Maximum Transmission Unit (MTU); your application will allow you to make such connection-specific changes.

- Routing information – No matter what protocol or authentication setting you choose, you will often have to take specific steps to make sure the right packets are sent across the right interface.

Also, you will need a cooperative network administrator to help you get the information listed above.

Configuring Microsoft PPTP Connections

Even though PPTP is often considered less secure than Cisco or OpenVPN connections, it is nevertheless popular. When configuring Microsoft PPTP, you are presented with quite an array of authentication, compression, and encryption options.

One of the challenges is to decide on an authentication method. Peer authentication means that the server will ask the host to identify itself. Options include:

- Challenge Handshake Authentication Protocol (CHAP): The RFC-compliant standard protocol. All you need to provide to CHAP is a username and a password. The protocol uses this information to enable authentication. MS-CHAP is Microsoft's implementation of the CHAP protocol. If you can't get definitive word from a network administrator, use MS-CHAP to connect to a Microsoft VPN server.
- EAP: An extension of the original PPP authentication protocol that allows the use of a certificate instead of a username and password. EAP is not as common as MS-CHAP and CHAP.

Many clients provide the option of refusing each of these authentication techniques. If you want to work and play well with the remote VPN server, you might need to do this explicitly for the PPP daemon on your system.

VPN connections generally compress packets to tunnel them more efficiently. Generally, you have three compression options:

- Microsoft Point-to-Point Compression (MPPC) – An older protocol based on a Lempel-Ziv (LZ) algorithm and usually reserved for ancient Windows clients (e.g., Windows 95 or NT). However, you might find that this form of compression works for your VPN connection.
- Deflate compression – A patentless protocol similar to MPPC. It is more universal than MPPC, but most Microsoft-oriented VPNs won't use it.
- BSD compression: Explained in RFC 1977, BSD compression is the traditional compression protocol.

I've found that compression often causes problems in connections. If you can't get definitive word from your admin, specify no compression at all at first, then experiment with these settings later.

The encryption setting is also an important consideration. Microsoft Point-to-Point Encryption (MPPE) is a subset of MPPC. You can use two different key levels: 40-bit or 128-bit. Companies in many countries use only 40-bit encryption. As you configure your PPTP client, either ask your network administrator for the key length, or else experiment with the key length setting.

Stateful MPPE is usually the best encryption option in that it provides the

Troubleshooting Issues

When troubleshooting connections, make sure you have the right modules installed and running. For example, I have the modules from the following list running on my Linux system (Ubuntu 8.04.1) when I connect as a client to a PPTP VPN:

```
ppp_mppe     8068  2
ppp_async    13312  1
crc_ccitt     3072  1 ppp_async
ppp_generic   29588  6
ppp_mppe
ppp_async
slhc          7040  1 ppp_generic
ppdev         10372  0
```

First, check your VPN documentation to determine which modules are necessary; then, use the *insmod* command to install these modules. Also, you can take the necessary steps to have them added to the */etc/modules* file.

The Maximum Transmission Unit (MTU) is another important setting, and I've never had to change it for my VPN connections. However, a friend of mine who has worked with VPNs for about 15 years has found that, in some cases, a VPN user might need to change the client MTU from the standard *1500* to a smaller value, such as *1490*.

When you encounter troubles, remember to enable debugging options in your client – you can always disable them later. In one case years ago, I found that enabling the debugging options in a client caused problems with the connections, but this is quite rare. Debugging options can include specifying ICMP packets in order to test a connection, as well as determining the size of the echo interval. I've found that the *ping* command is just as good, if not better.

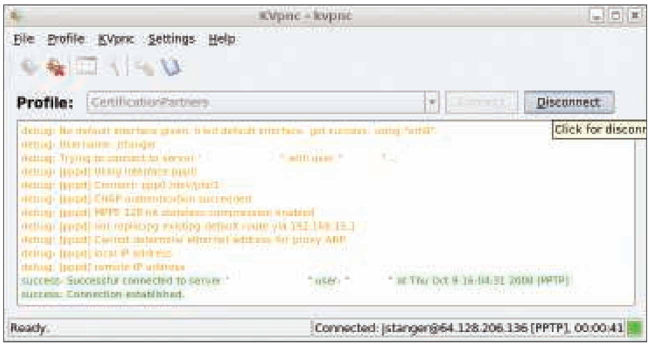
A strange but sometimes useful troubleshooting option is to disable encryption. Of course, if you do this, you'll lose one of the key benefits of having a VPN connection. Most servers reject unencrypted connections; however, I have seen cases in which an unencrypted VPN connection is a possible option. I'm not saying this makes any sense; I'm just saying I've seen it. So, if none of the possible encryption options work, try disabling encryption and seeing if you can make the connection. A successfully connection without encryption narrows the list of possible problems.

IP masquerading and other forms of Network Address Translation (NAT) pose a number of additional problems for VPN connections. If you're using ESP and Authentication Headers (AH), for instance, you will have problems with NAT because AH runs a checksum that includes values such as the IP address for the connection. Because masquerading/NAT modifies the IP address, the checksum run by AH will be different. Because most NAT firewalls/masquerading schemes can't forward the AH keys, the checksums won't be correct.

Account Lockout

As you experiment with your VPN connection, make sure your account doesn't get locked out. Most VPNs have a hard time differentiating between legitimate experimentation with VPN settings and someone trying to break in. If you're locked out, you won't be able to get in, even if you have finally specified the correct settings.

If you find yourself unable to connect, pay close attention to logfile and debug file messages to see whether authentication and connection messages have changed in any way; a subtle change might indicate that you are, in fact, locked out of the account. If this is the case, you'll have to wait until the account gets unlocked. Hopefully, the lockout is temporary. (My account is re-enabled automatically after 30 minutes.) Otherwise, you'll have to contact your network administrator each time you get locked out in order to get your account enabled again.



```
debug: No default interface given. tried default interface, got success, using "eth0".
debug: Username: jstanger
debug: Trying to connect to server "                " with user "        " ...
debug: [pppd] Using interface ppp0
debug: [pppd] Connect: ppp0 <--> /dev/pts/1
debug: [pppd] CHAP authentication succeeded
debug: [pppd] MPPE 128 bit stateless compression enabled
debug: [pppd] not replacing existing default route via 192.168.13.1
debug: [pppd] Cannot determine ethernet address for proxy ARP
debug: [pppd] local IP address
debug: [pppd] remote IP address
success: Successful connected to server: "              " user: "        " at Thu Oct 9 16:04:31 2008 (PPTP)
success: Connection established.
```

```
Ready.    Connected: jstanger@64.128.206.136 [PPTP], 00:00:41
```

Figure 3: KVpnc running a legacy PPTP session.

fastest network connections. As I discuss later in this article, you must enable the MPPE module for any of these settings.

DNS Issues

Most DNS clients will ask you whether you want to use the standard DNS information found in */etc/resolv.conf* or the DNS server information provided by the VPN server. This, of course, is up to you. But many times, I've found that establishing a VPN connection causes unexpected changes to my DNS resolution, mainly because the VPN client will still make changes to my */etc/resolv.conf* file, even if I tell it not to.

The best solution is to use a VPN client that works the way in which I expect. But failing that, I simply create a script that copies the right */etc/resolv.conf* file. If your VPN client asks you whether you want to "peer DNS through the tunnel," this is simply the client asking you whether you want it to update your */etc/resolv.conf* file with DNS server information. Make your own choice here, depending on the information from your network administrator. If you don't use information from the VPN server, chances are that the host using the VPN won't have access to the DNS names of your internal sources. However, make sure that if the VPN server provides the DNS information, the client's routing tables are updated to access the internal VPN server. Otherwise, the client host will experience a DNS resolution problem for internal resources – and possibly external resources.

Enabling GRE Support

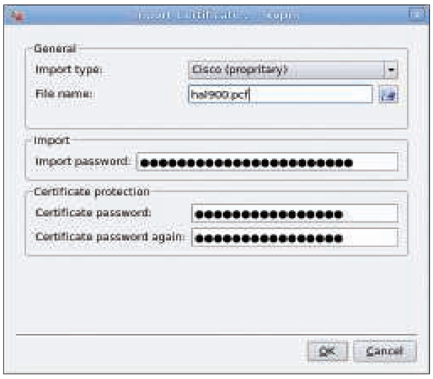
If you're trying to connect a VPN client to a Microsoft PPTP connection and you are using a Linux box as a firewall for your broadband connection, you'll have to take an additional step. Allow Generic Route Encapsulation (GRE) protocol to pass through the firewall. If, for example, you are using iptables on your Linux firewall and your VPN server has the IP address *189.44.45.3*, you would enter the following:

```
iptables -I FORWARD -p 47 -d ⮐
189.44.45.3 -j ACCEPT
```

Routing Packets through the Connection

Sometimes you'll find it necessary to explicitly route packets through a specific interface. Many Windows administrators consider this one of the biggest challenges in working with Linux clients.

The need for explicit routing is especially important when your remote network is using public IP addresses. Even



General
Import type: Cisco (proprietary)
File name: hal900.pcf
Import
Import password: ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
Certificate protection
Certificate password: ●●●●●●●●●●●●●●●●●●●
Certificate password again: ●●●●●●●●●●●●●●●●●●●

Figure 4: Importing a Cisco certificate in KVpnc.

though you have a VPN tunnel, your network interface might still try to route packets across the Internet, rather than through the VPN tunnel. Many times when I have helped troubleshoot "failed" VPN connections, all I had to do was simply add a few alternative routes to the default routing table. Sometimes I would do this with the *route* command (as root). Other times I used the VPN GUI application.

The *route* command is the standard:

```
route add -net 13.163.97.23 ↩
netmask 255.255.255.255 ↩
dev ppp0
```

Also, you can use the *ip* command

```
ip route add 171.87.44.54/24 ↩
dev ppp0
```

or specify the routes with the use of the VPN software GUI interface. In some cases, if you don't add these routes, the packets that you intended to go through the VPN tunnel will be routed through your wireless or Ethernet card instead of your VPN interface.

GUI Options

The GUI VPN applications are getting much better at adding routes on their own. Figure 1 shows the settings for KVpnc. Figure 2 provides a similar configuration in NetworkManager.

As these images show, packets that match the IP address and subnet mask will not be sent across a standard network connection; rather, they will be sent through the VPN tunnel.

KVpnc (Figure 3), which is supported by many distributions, is perhaps the most versatile client in that it supports L2TP, Cisco free and proprietary VPN protocols, and OpenVPN and Microsoft PPTP. KVpnc also lets you import certificates, as shown in Figure 4.

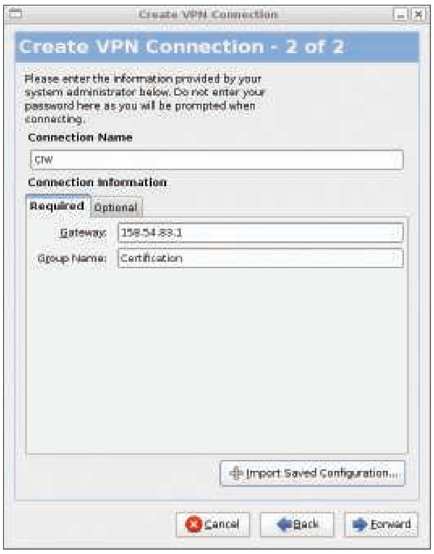
Even though the user interface programmers can't spell particularly well (notice the word "proprietary" is misspelled in my version of the program; Figure 4), the KVpnc team has created an implementation that works particularly well with Cisco devices.

The venerable *pptpconfig* tool also is available with many distributions. The key to getting pptpconfig to work properly is to make sure the encryption settings are configured exactly as your network administrator has them set.

I've often found that requiring MPPE encryption and enabling stateful MPPE encryption are important. In the case of pptpconfig, you would select "Require Microsoft Point-to-Point Encryption (MPPE)" and "Refuse Stateless Encryption" to accomplish this.

Pptpconfig also has the ability to add routes automatically. Simply click the *Routing* tab, then select the *Client to LAN* radio button and enter the routes of the systems you want to connect with through your VPN tunnel.

Many Linux users prefer the NetworkManager client for one simple reason: It tends to work. Plugins are available for NetworkManager that support various protocols, including OpenVPN, Microsoft PPTP, and Cisco's L2TP methods. In my Ubuntu system, I use *apt-get*, but you can also search for the appropriate plugins with Synaptic. Once you've added the plugin, all you have to do is click on the network icon then select VPN connections to begin entering the appropriate information. Figure 5 shows the steps for configuring a Cisco connection on an Ubuntu system.



Create VPN Connection - 2 of 2

Please enter the information provided by your system administrator below. Do not enter your password here as you will be prompted when connecting.

Connection Name
CIW
Connection Information
Required | Optional
Gateway: 158.54.83.1
Group Name: Certification

Figure 5: Configuring a Cisco VPN connection in NetworkManager.

NetworkManager supports both shared-key and X.509 certificate-based encryption. The keys to getting NetworkManager to function properly include installing the racoon daemon software to handle the Internet Key Exchange, creating or otherwise obtaining a shared key from your administrator, and creating or otherwise obtaining signed certificates from your administrator (if you are using certificates). Also, you have the option of importing the saved configuration files of existing connections.

Supporting IPsec via the 2.6 Kernel

Make sure you install the right supporting daemons for your connection. If your system uses any version of the 2.6 kernel, it natively supports IPsec, but if you want to use KVpnc or vpnc, you'll still need to install the racoon daemon, which takes care of the key exchange for IPsec implementations. To install racoon, use your native package manager or look online for instructions on building a racoon connection [4].

To support FreeS/WAN, the older IPsec standard, you'll have to install the *ipsec* daemon. If you fail to install the appropriate daemon, your VPN implementation will fail because it will be impossible for your system to conduct the necessary key exchanges when establishing the tunnel.

Conclusion

Establishing VPN today has gotten much easier, but the GUI VPN clients still don't do it all for you. Although working with Microsoft, Cisco, and OpenVPN servers requires a bit of troubleshooting acumen, if you keep working at it, you'll find success. ■

INFO

[1] OpenVPN: *http://openvpn.net*

[2] *http://www.gnome.org/projects/NetworkManager*

[3] *http://pptpclient.sourceforge.net*

[4] *http://www.netbsd.org/docs/network/ipsec/rasvpn.html*