Paranoid Penguin - Linux VPN Technologies

By Mick Bauer
Created 2005-01-05 02:00

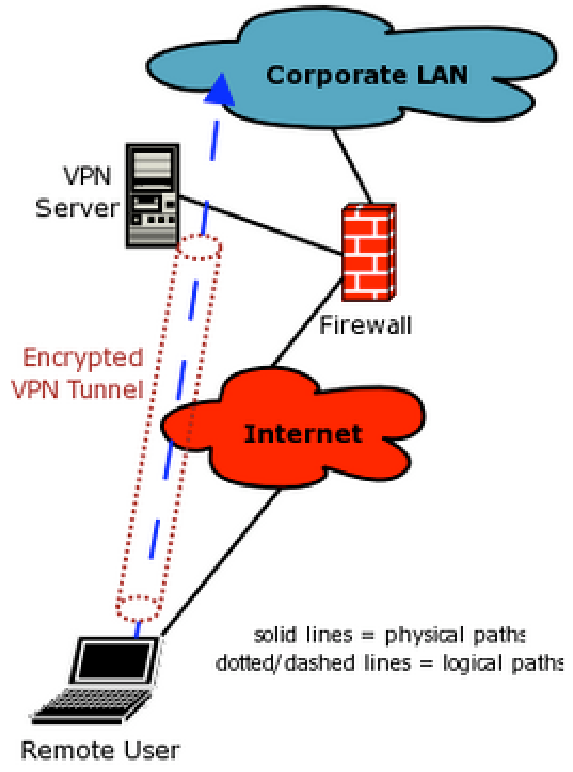
Which virtual private network is right for you? Mick runs down the options and comes up with some winners and some warnings.

Virtual private networks, or VPNs, are useful and convenient things. Road warriors use them to connect to their home networks securely while traveling; geographically dispersed organizations use them to encrypt WAN links that use public bandwidth; and wireless LAN users use them to add a layer of security to their WLAN connections.

A number of VPN packages are available for Linux: FreeS/WAN, OpenS/WAN, PoPToP, OpenVPN and tinc, just to name a few. But how do you choose the right one for a given job? I show you how in this month's column.

VPN Architecture

VPNs generally address two different needs. The first is the need to allow users to connect to a private network with an encrypted connection through some untrusted medium, such as the Internet or a wireless LAN (WLAN). Figure 1 illustrates the remote-access scenario.



[1]

Figure 1. Remote-access VPNs allow one remote system to connect to a network.

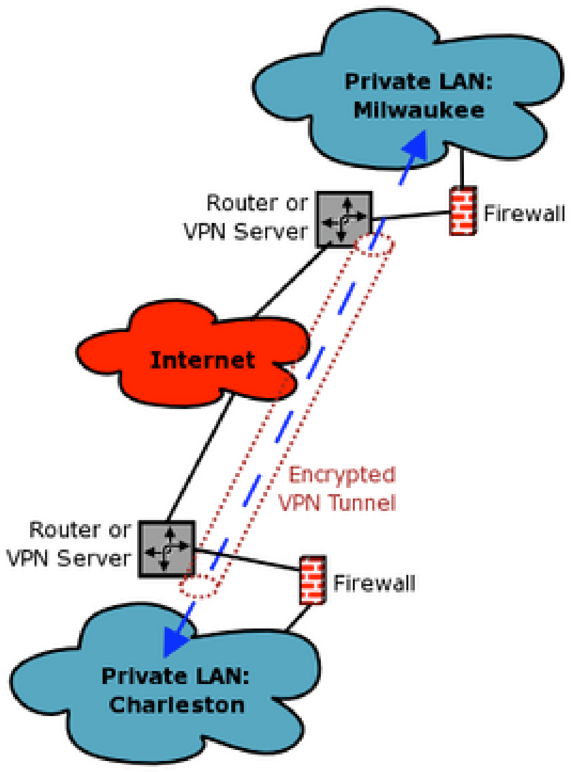
In Figure 1, the dashed-blue data flow implies access to the entire corporate LAN. In practice, a remote-access VPN tunnel can limit that access through access control lists (ACLs) or firewall rules. Access can even, in the case of SSL-VPN, be limited to a single application on a single host (I'll explain SSL-VPN shortly).

For simplicity's sake, Figure 1 shows a single client; however, this scenario nearly always involves many clients. In other words, the remote-access scenario requires a client-server architecture in which a single VPN server or concentrator can build tunnels with hundreds or even thousands of remote users. (In this article I'm using the term client-server in a very broad sense, not in the specific software development sense.)

Although Figure 1 shows a VPN server acting as the corporate LAN's VPN endpoint, the firewall also could be used for this-both commercial and free firewalls, including Linux iptables/Netfilter, support VPN protocols.

Important: in this article when I say tunnel, I mean encrypted tunnel. Yes, technically the term tunnel simply means one data stream encapsulated into another. But the whole point of VPNs is encryption, so in this context, tunnel equals encryption.

The second VPN need is to create an encrypted point-to-point connection between two different networks over some untrusted medium. Whereas remote-access VPNs use a client-server model, point-to-point tunnels use a peer-to-peer model. Figure 2 shows a point-to-point VPN architecture.



[2]

Figure 2. Point-to-point VPNs connect two networks.

Routers often are used in the point-to-point VPN scenario. Cisco's IOS router operating system, for example, supports several different VPN protocols. Firewalls and dedicated VPN concentrators/servers, however, also can be used as VPN endpoints.

Those are the two problems that VPN architectures address. Two more architectural considerations are worth mentioning, network address translation (NAT) and performance.

With most VPN protocols, NAT can be problematic. That is, your VPN servers generally can't have translated addresses. This is why, in both Figures 1 and 2, none of the VPN endpoints are in corporate LANs, except for the remote client in Figure 1-remote-access clients are the exception to this rule.

Using your firewall as a VPN server is one way to get around the NAT problem, but that brings us to the second consideration: VPN tunnels can be CPU-intensive. Unless your firewall has a crypto-accelerator card or doesn't need to support many concurrent VPN tunnels, you're probably better off using a dedicated VPN server than you are using your firewall for VPNs.

Now that we've covered the basics, let's look at specific VPN software for Linux.

FreeS/WAN and OpenS/WAN

The IPSec protocol, which really is a set of security headers in the Internet Protocol (IP) v6 back-ported to IPv4, is the most open, powerful and secure VPN protocol. It's also the most ubiquitous. IPSec support is now part of virtually all important computer and network-device operating systems. On Linux, it's provided by FreeS/WAN and OpenS/WAN.

I covered FreeS/WAN in depth in "An Introduction to FreeS/WAN", Parts I and II [in the January and February 2003 issues of *LJ*, respectively]. In a nutshell, FreeS/WAN adds a couple of kernel modules and user-space commands to your Linux system. Because the IP protocol is part of your kernel, it follows that extensions to the IP protocol also must be incorporated into your kernel.

The Linux 2.6 kernel includes these IPSec modules, called the 26sec modules. The Linux 2.4 kernels included with Red Hat Enterprise Linux do as well-they contain backported versions of the 26sec modules. If you already have IPSec kernel modules, you need install only FreeS/WAN's user-space commands.

FreeS/WAN may be included with your Linux distribution of choice (SuSE, which is mine, includes it). However, the FreeS/WAN Project recently folded, so if your distribution doesn't include FreeS/WAN and you need to compile it from source, you're better off using OpenS/WAN.

OpenS/WAN was started by a group of FreeS/WAN developers who were unhappy with how things were going with the FreeS/WAN Project. Thus, when FreeS/WAN ended, OpenS/WAN succeeded it. Eventually, we can expect the major Linux distributors to replace their FreeS/WAN packages with OpenS/WAN. In the meantime, you can obtain the latest OpenS/WAN source code from the OpenS/WAN Web site (see the on-line Resources).

Advantages of FreeS/WAN and OpenS/WAN include:

- Maturity: this is one of the older Linux VPN technologies.
- Security: IPSec is a robust, powerful and well-designed protocol.
- Interoperability: client systems running other OSes probably have IPSec client software that interoperates with Free/OpenS/WAN.
- Flexibility: IPSec is ideal for both remote-access and point-to-point VPNs.

Disadvantages include:

- Complexity: IPSec is not easy to understand, and it requires digital certificates.
- Power: if all you need to do is provide remote users with access to one application running on one internal system, IPSec may be overkill. IPSec is designed to connect entire networks to each other.

Having said that, if after reading this entire article you're still confused as to which VPN solution is best for you, I recommend that you default to FreeS/WAN or OpenS/WAN. IPSec is by far the most mature and secure VPN technology for Linux. In my opinion, these advantages outweigh the disadvantage of being complex. See the FreeS/WAN and OpenS/WAN Web sites for more information on configuring and using these packages.

OpenSSH

It's tempting to think of OpenSSH purely as a remote shell tool. But the SSH protocol supports the secure tunneling of *any* single-TCP-port service, not only shells, by using the -L and -R options.

For example, suppose I have a secure shell server in a firewalled but publicly accessible DMZ network and a Microsoft SQL server in my internal network. If I create a firewall rule allowing MS-SQL transactions from the SSH server to the MS-SQL server and if my SSH server allows port forwarding, I could create an SSH tunnel between some remote host and my SSH server that allows remote database clients to send queries to the remote host that are tunneled to the SSH server and forwarded to the MS-SQL server. The SSH command on my remote host would look like this:

```
        bash-#> ssh -L 11433:ms-sql.server.name:1433 myaccount@remote.ssh-server.name
```

where `ms-sql.server.name` is the name or IP address of the MS-SQL server, and `remote.ssh-server.name` is the name or IP address of the DMZed SSH server.

It's even possible to tunnel PPP over SSH, which technically achieves the same thing as IPSec-that is, the ability to tunnel *all* traffic between two networks. However, this is one of the least efficient means of doing so; it involves much more administrative overhead than the other tools and methods described in this article.

In summary, OpenSSH is a good tool for tunneling traffic from specific applications running on specific hosts; it can be used in this way in both remote-access and point-to-point VPN scenarios. It is less useful, however, for tunneling all traffic between remote networks or users.

See the ssh(1) and sshd_config(5) man pages for more information on using OpenSSH for port forwarding.

Stunnel

Conceptually, Stunnel, an SSL wrapper, provides functionality equivalent to SSH port forwarding. Stunnel is a standard package on most Linux distributions nowadays.

The main difference between Stunnel and SSH is that Stunnel is much more limited; *all* it does is encrypted port forwarding. Also, because Stunnel really is a sort of front end for OpenSSL, Stunnel requires you to configure and install digital certificates, which perhaps offsets some of its simplicity. Otherwise, Stunnel shares OpenSSH's limitations as a VPN tool.

See the stunnel(8) man page, the Stunnel Web site and my article "Rehabilitating Cleartext Network Applications with Stunnel" (*LJ*, September 2004) for information on configuring and using Stunnel.

OpenVPN

OpenVPN is an SSL/TLS-based user-space VPN tool that encapsulates all traffic between VPN endpoints inside ordinary UDP or TCP packets (ordinary in the sense that they don't require any modifications to your kernel's IP stack). OpenVPN was created because in the opinion of its author, James Yonan, the world needed a less complex alternative to IPSec.

Because no special kernel modules or modifications are necessary, OpenVPN runs purely in user space, making it much easier to port across operating systems than IPSec implementations. And, by virtue of using the standard OpenSSL libraries, OpenVPN, like Stunnel, does a minimum of wheel re-invention. Unlike homegrown cryptosystems, such as those used in the CIPE and tinc VPN packages (see below), all of OpenVPN's critical operations are handled by OpenSSL. OpenSSL itself certainly isn't flawless, but it's under constant scrutiny for security flaws and is maintained by some of the Open Source community's finest crypto programmers.

OpenVPN is a good match for point-to-point VPNs, but until version 2.0 (still in beta as of this writing, November 2004), OpenVPN had the limitation of being able to accommodate only a single tunnel on a given listening port. If you wanted to use OpenVPN to provide remote-access VPN tunnels to ten different users, you needed to run ten different OpenVPN listeners, each using its own UDP port, such as UDP 10201, UDP 10202 and UDP 10203 and seven more. Therefore, if you want to use OpenVPN for remote-access VPNs, you'll be much happier with OpenVPN 2.0 (even in its beta state), unless you have only a handful of users.

OpenVPN is included with SuSE Linux 9.1 and probably other distributions as well. See the OpenVPN Web site for configuration information and for the latest OpenVPN software.

PoPToP and the Linux PPTP Client

IPSec isn't the only low-level VPN protocol used on the Internet. Microsoft's Point-to-Point Tunneling Protocol (PPTP) also has its adherents, mainly because it has been a standard component of Microsoft's server operating systems since Windows NT 4.0 and because, unlike IPSec, which can only tunnel IP packets, PPTP can be used to tunnel not only IP but also other protocols, such as NETBEUI and IPX/SPX.

Linux support for PPTP comes in two flavors, PoPToP on the server side and Linux PPTP Client on the client side.

As handy as it is to tunnel non-IP protocols and as ubiquitous as Windows servers are, PPTP has one big problem. When Bruce Schneier and Dr Mudge analyzed the Windows NT 4.0 implementation of PPTP in 1998, they found serious security flaws that were only partially mitigated by the release of MSCHAPv2 shortly afterward. MSCHAP is an authentication protocol PPTP depends on; it was the source of the worst vulnerabilities Schneier and Mudge found. Schneier has a Web page devoted to their analysis (see Resources).

Schneier and Mudge analyzed Windows NT 4.0; what about a Linux PoPToP server? According to the PoPToP Web site (in "PoPToP Questions and Answers"): "PoPToP suffers the same security vulnerabilities as the NT sever (this is because it operates with Windows clients)."

I do not recommend using PPTP unless you can configure your PPTP server and all PPTP clients to use MSCHAPv2 (not all Windows versions support MSCHAPv2) and you're trying to do something that simply can't be done with IPSec. IPSec is much better designed and is provably more secure. Furthermore, non-IP network protocols aren't as important as they once were; both Windows and Novell Netware can do everything over IP.

I'll stop short of saying something like "you can't use PPTP, because it's lame." As I argued last month, security is about risk management, not about seeking some sort of utopian state of pure security. After you read up on the Schneier and Mudge controversy, Microsoft's response and MSCHAPv2, and after you carefully examine your particular organization's needs and capabilities, you conceivably could decide that for you, PPTP represents a justifiable compromise between security and functionality-just don't tell anyone *I* said you should use it!

Other Linux VPN Packages

Three other Linux VPN tools are worth mentioning here, because you'll occasionally see references to them. Two of them I recommend against using, and the third I'm not sure about.

CIPE and vtun conceptually are similar to OpenVPN. They encapsulate traffic into encrypted UDP or TCP packets. Unlike OpenVPN, however, they use homegrown cryptosystems rather than OpenSSL. That is, they do use standard cryptographic algorithms such as Blowfish and MD5, but in custom implementations (session-key generation, user authentication and so on). Because implementation is one of the hardest parts of cryptographic programming, this is a dangerous thing to do, and sure enough, the cryptographer Peter Gutmann has found serious flaws in both CIPE and vtun.

In neither case have the flaws Gutmann identified been fixed, as far as I can tell. And neither CIPE nor vtun appears to be in active development anymore (CIPE for sure is not), which is reason enough to avoid any security application, except when that application is part of a Linux distribution whose packagers provide patches themselves. I do not, therefore, recommend using either CIPE or vtun.

tinc, like CIPE and vtun, uses a custom cryptographic implementation to encapsulate VPN traffic in encrypted UDP packets. And like those packages, Gutmann found flaws in tinc, in the same analysis I referred to earlier. Unlike CIPE and vtun, however, tinc's developers have responded to Gutmann's findings in a credible manner; at least from my perspective (IANAC-that is, "I am not a cryptographer"), they appear to have some clue as to what they're doing.

I leave it to you to check out the tinc Web site, read Gutmann's page (which stops well short of being a serious research report), do a few Google searches for the aftermath of Gutmann's statements and decide for yourself whether tinc looks like just the thing you've been looking for or more like an unjustifiable risk given the availability and quality of OpenS/WAN and OpenVPN.

SSL-VPN

Finally, a word about a popular new approach supported in many commercial VPN products, SSL-VPN. SSL-VPN works in practically the same way as Stunnel and SSH port forwarding. It tunnels network transactions on a per-service, per-server basis rather than at the circuit level. Unlike those other approaches, however, SSL-VPN products present end users with a centralized Web interface in which all available servers/services hosted by the VPN server are listed as hyperlinks. When the user clicks on a link, typically a Java applet is downloaded that serves as the application client software.

The SSL-VPN server products I've seen are all proprietary, but because the client side is usually cross-platform, in Java, Linux systems can act as SSL-VPN clients.

Conclusions

FreeS/WAN and OpenS/WAN (preferably the latter) and IPSec are probably the most secure and powerful VPN tools in the Linux toolbox. OpenVPN appears to be a simpler, albeit less-scrutinized, alternative. OpenSSH and Stunnel provide handy point solutions when encapsulating more than a few specific applications is overkill. Still other Linux VPN tools are available, but some are provably dangerous, and on the others the jury is still out. Which VPN tool is the best fit for you? Obviously, I can't tell you that without knowing your particular needs and resources. But, I hope this little overview has at least given you a useful starting point.

Resources for this article: www.linuxjournal.com/article/7923 [3].

Mick Bauer, CISSP, is *Linux Journal*'s security editor and an IS security consultant in Minneapolis, Minnesota. He's the author of *Building Secure Servers With Linux* (O'Reilly & Associates, 2002).

Links

[1] http://www.linuxjournal.com//articles/lj/0130/7881/7881f1.png
[2] http://www.linuxjournal.com//articles/lj/0130/7881/7881f2.png
[3] http://www.linuxjournal.com//article/7923

Source URL: http://www.linuxjournal.com/article/7881