Linux VPNs with OpenVPN

MICK BAUER

Connect safely to the mother ship with a Linux VPN.

The other day, I was accompanying local IT security legend and excellent friend Bill Wurster on his last official "wireless LAN walkabout" prior to his retirement (it was his last day!), looking for unauthorized wireless access points in our company's downtown offices. As we strolled and scanned, we chatted, and besides recalling old times, battles won and lost, and one very colorful former manager, naturally we talked about wireless security.

Bill, who has dedicated much of the past decade to wireless network security work in various capacities, reiterated something all of us in the security game have been saying for years: one of the very best things you can do to defend yourself when using someone else's WLAN is to use a Virtual Private Network (VPN) connection to connect back to some other, more trustworthy, network.

You might think the main value of a VPN connection is to encrypt sensitive communications, and that's certainly very important. But if you make your VPN connection your default route and use your trusted network's DNS servers, Web proxies and other "infrastructure" systems to communicate back out to the Internet (for example, for Web surfing), you won't have to worry about DNS spoofing, Web-session hijacking or other entire categories of localized attacks you might otherwise be subject to on an untrusted LAN.

For this reason, our employer *requires* us to use our corporate VLAN software any time we connect our corporate laptops to any WLAN "hotspot" or even to our personal/home WLANs. So, Bill asked me, why don't you write about making VLAN connections with Linux laptops?

This isn't the first good idea Bill's given me (nor, I hope, is it the last). So this month, I begin a series on Linux VPNs, with special attention to OpenVPN. I'd like to dedicate this series to Bill Wurster, whose skill, creativity, enthusiasm and integrity have been such an inspiration not only to me but also to a couple generations of coworkers. Your warmth and wisdom will be sorely missed, Bill—here's wishing you a long, happy and fun retirement!

VPN Basics

Rather to my surprise, the overview of VPN technologies in general, and Linux VPN choices in specific, that I did in 2005 is still pretty current (see Resources for a link to this earlier article). If you find the overview I'm about to give to be too brief, I refer you to that piece. Here, though, is a brief introduction.

To create a "Virtual Private Network" is to extend some private network—for example, your home Local Area Network (LAN) or your employer's Wide Area Network (WAN)—by connecting it to other networks or systems that aren't physically connected to it, using some sort of "virtual" (non-dedicated, non-persistent) network connection over network bandwidth not controlled or managed by you. In other words, a VPN uses a public network (most commonly the Internet) to connect private networks together.

Because by definition a public network is one over which you have no real control, a VPN must allow for two things: unreliability and lack of security. The former quality is mainly handled by low-level error-correcting features of your VPN software. Security, however, is tricky.

You must select a VPN product or platform that uses good security technologies in the first place (the world is filled with insecure VPN technologies), and you must furthermore *enable* those security features and resist the temptation to weaken them in order to improve VPN performance (which is ultimately futile anyhow, as Internet bandwidth is generally slower and less reliable than other long-range network technologies, such as dedicated circuits).

There are three categories of security we care about in this context:

1. Authentication: is the computer or network device trying to connect to the trusted network an expected, authorized VPN endpoint? Conversely, am I, the VPN client, really connecting to my trusted network or has my connection request been redirected to some impostor site?

2. Data integrity: is my connection truly usable only by me and my trusted network, or is it possible for some outsider to inject extraneous traffic into it or to tamper with legitimate traffic?

3. Privacy: is it possible for an attacker to read data

contained in my VPN traffic?

You may not need all three types of security. For example, if you're using a VPN connection to transfer large quantities of public or otherwise nonsensitive information, and are only using a VPN in the first place to tunnel some not-normally-IP-routable protocol, you might consider a "null-encryption" VPN tunnel. But even in that case, you should ask yourself, what would happen if an attacker inserted or altered data in these transactions? What would happen if an attacker initiated a bogus connection altogether?

Luckily, some VPN protocols, such as IPsec, allow you to "mix and match" between features that address different security controls. You can, for example, use strong authentication and cryptographic error/integrity-checking of all data without actually encrypting the tunnel. In most situations, however, the smart thing to do is leverage good authentication, integrity and privacy (encryption) controls. The remainder of this series assumes you need all three of these.

There are two common usage scenarios for VPNs: "site-to-site" and "remote-access". In a site-to-site VPN, two networks are connected by an encrypted "tunnel" whose endpoints are routers or servers acting as gateways for their respective networks. Typically, such a VPN tunnel is "nailed" (or "persistent")—once established, it's maintained as an always-available, transparent route between the two networks that end users aren't even aware of, in the same way as a WAN circuit, such as a T1 or Frame Relay connection.

In contrast, each tunnel in a remote-access VPN solution connects a single user's system to the trusted network. Typically, remote-access VPN tunnels are dynamically established and broken as needed. For example, when I work from home, I establish a VPN tunnel from my company's laptop to the corporate VPN concentrator. Once my tunnel's up, I can reach the same network resources as when I'm in the office; with respect to computing, from that point onward I can work as normal. Then at the end of the day, when it's time to shut down my machine, I first close my VPN tunnel.

For site-to-site VPNs, the endpoints are typically routers. All modern router platforms support VPN

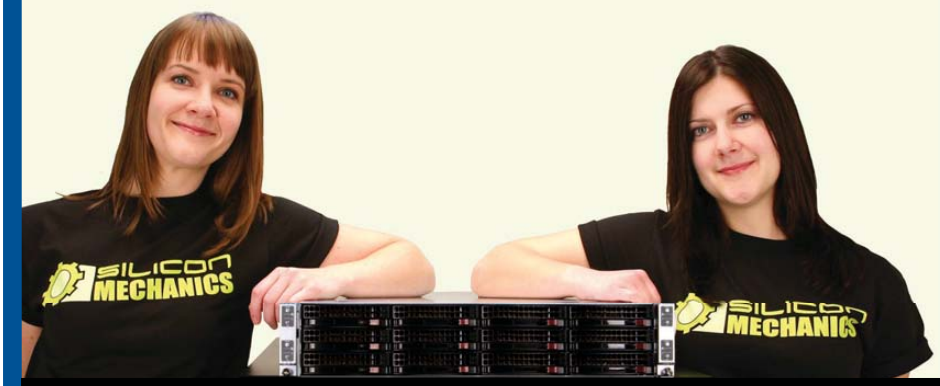
SILICON MECHANICS

visit us at www.siliconmechanics.com
or call us toll free at 866-352-1173

intel Xeon inside™

Powerful. Intelligent.

Meet Victoria (on the right). She is the Silicon Mechanics marketing expert responsible for the events and promotions that keep our customers informed about new products and technologies. She's pictured here with her twin sister Veronica, an industrial designer, to help us make a point about what makes twin servers so popular. Victoria and Veronica are twins, but they don't look exactly alike and they don't do the same job. Twin servers are two servers in a single 1U chassis: they can be configured differently, and they handle their own individual workloads.



With the introduction of the Rackform iServ R4410 from Silicon Mechanics, twin power has reached a whole new level: the twin². A twin² is a 2U 4-node system. It supports four swappable, full-featured nodes in a 2U chassis with redundant power. In each node you'll find 2 of the new Intel® Xeon® 5500 Series processors, 12 DDR3 DIMM slots, 3 hot-swap drives, and an integrated dual-port GigE adapter. Integrated InfiniBand is also available with the R4410-IB. Unmatched density and state-of-the-art processors make the R4410 a superior choice for high-performance computing, and Victoria is spreading the word with enthusiasm.

When you partner with Silicon Mechanics, you get more than the latest and greatest in density, performance, and energy efficiency—you get an expert like Victoria.

For more information about the Rackform iServ R4410 visit www.siliconmechanics.com/R4410

Expert included.

Silicon Mechanics and the Silicon Mechanics logo are registered trademarks of Silicon Mechanics, Inc. Intel, the Intel logo, Xeon, and Xeon Inside, are trademarks or registered trademarks of Intel Corporation in the US and other countries.

protocols, such as IPsec. Establishing and breaking VPN tunnels, however, can be computationally expensive—that is, resource-consuming.

For this reason, if you need to terminate a lot of site-to-site tunnels on a single endpoint (for example, a router in your data center connecting to numerous sales offices), or if you need to support many remote-access VPN clients, you'll generally need a dedicated VPN concentrator. This can take the form of a router with a crypto-accelerator circuit board or a device designed entirely for this purpose (which is likely to have crypto-accelerator hardware in the form of onboard ASICs).

A number of tunneling protocols are used for Internet VPNs. IPsec is an open standard that adds security headers to the IPv4 standard (technically it's a back-port of IPv6's security features to IPv4), and it allows you either to authenticate and integrity-check an IPv4 stream "in place" (without creating a tunnel per se) or to encapsulate entire packets within the payloads of a new IPv4 stream. These are called Authentication Header (AH) mode and Encapsulating Security Payload (ESP) mode, respectively.

Microsoft's Point-to-Point Tunneling Protocol (PPTP) is another popular VPN protocol. Unlike IPsec, which can be used only to protect IP traffic, PPTP can encapsulate non-IP protocols, such as Microsoft NETBIOS. PPTP has a long history of security vulnerabilities.

Two other protocols are worth mentioning here. SSL-VPN is less a protocol than a category of products. It involves encapsulating application traffic within standard HTTPS traffic, and different vendors achieve this in different (proprietary) ways. SSL-VPN, which usually is used in remote-access solutions,

2009: a Bad Year for SSL/TLS?

OpenVPN depends on OpenSSL, a free implementation of the SSL and TLS protocols, for its cryptographic functions. But SSL/TLS has had a bad year with respect to security vulnerabilities. First, back in February 2009, Moxie Marlinspike and others began demonstrating man-in-the-middle attacks that could be used to intercept SSL/TLS-encrypted Web sessions by "stripping" SSL/TLS encryption from HTTPS sessions.

These are largely localized attacks—in practice, the attacker usually needs to be on the same LAN as (or not very far upstream of) the victim—and they depend on end users not noticing that their HTTPS sessions have reverted to HTTP. The NULL-prefix man-in-the-middle attack that Marlinspike and Dan Kaminsky subsequently (separately) demonstrated that summer was more worrisome. It exploited problems in X.509 and in Firefox that made it possible for an attacker essentially to proxy an HTTPS session, breaking the encryption in the middle, in a way that allows the attacker to eavesdrop on (and meddle with) HTTPS traffic in a way that is much harder for end users to notice or detect.

But, that wasn't all for 2009 (which isn't even finished yet, as I write this). In November, security researchers uncovered problems with how the SSL/TLS protocol handles session state. These problems at least theoretically allow an attacker not only to eavesdrop on but also inject data into SSL/TLS-encrypted data streams. Although the effects of this attack appeared similar to those of the NULL-prefix attack, the latter involved client/browser-side X.509 certificate-handling functions that were browser/platform-specific and didn't involve any server-side code.

In contrast, the November revelation involved actual flaws in the SSL/TLS protocol itself, whether implemented in Web browsers, Web servers or *anything else* using SSL/TLS. Accordingly, application or platform-specific patches couldn't help. The SSL/TLS specifications themselves, and *all implementations of it* (mainly in the form of libraries such as OpenSSL), had to be changed.

That's the bad news. OpenVPN depends on protocols that have been under intense fire lately. The good news is, because e-commerce, on-line banking and scores of other critical Internet applications do as well, at the time of this writing, the IETF has responded very rapidly to make the necessary revisions to the SSL/TLS protocol specifications, and major vendors and other SSL/TLS implementers appear to be poised to update their SSL/TLS libraries accordingly. Hopefully, by the time you read this, that particular issue will have been resolved.

Obviously, by even publishing this article, I'm betting on the continued viability of SSL/TLS and, therefore, of OpenVPN. But, I'd be out of character if I didn't speak frankly of these problems! You can find links to more information on these SSL/TLS issues in the Resources section.

typically allows clients to use an ordinary Web browser to connect to an SSL-VPN gateway device. Once authenticated, the user is presented with a Web page consisting of links to specific services on specific servers on the home network.

How, you might ask, is that different from connecting to a "reverse" Web proxy that's been configured to authenticate all external users? For Web traffic, most SSL-VPN products do, in fact, behave like standard Web proxies. The magic, however, comes into play with non-HTTP-based applications, such as Outlook/Exchange, Terminal Services, Secure Shell and so forth.

For non-HTTP-based applications, the SSL-VPN gateway either must interact with a dedicated client application (a "thick" client) or it must push some sort of applet to the user's Web browser. Some SSL-VPN products support both browser-only access and thick-client access. Others support only one or the other.

Thick-client-only SSL-VPN, unlike browser-accessible, can be used to encapsulate an entire network stream, not just individual applications' traffic. In common parlance though, the term SSL-VPN usually connotes browser clients.

And, that brings us to the subject of the remainder of this month's column and the exclusive focus of the next few columns: other SSL-based VPNs. As I just implied, it's possible to encrypt an entire network stream into an SSL session, for the same reason it's possible to stream audio, video, remote desktop sessions and all the other things we use our browsers for nowadays.

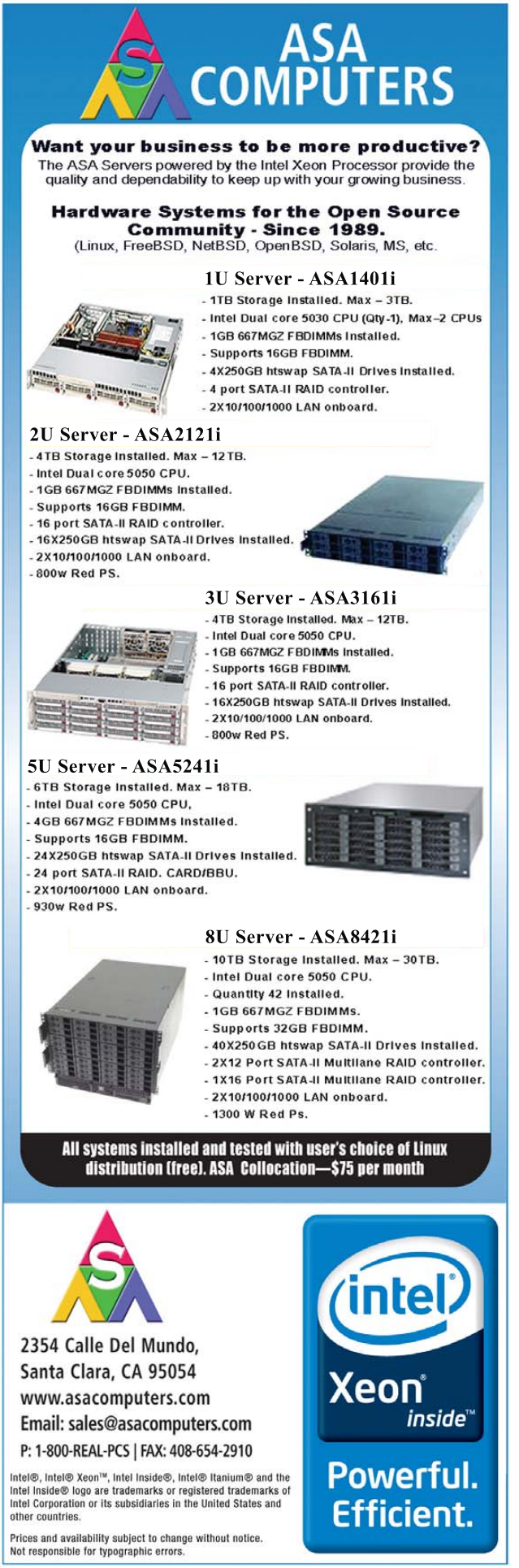
OpenVPN is a free, open-source VPN solution that achieves this very thing, using its own dæmon and client software. Like PPTP, it can tunnel not only IP traffic, but also lower-level, non-IP-based protocols, such as NETBIOS. Like IPsec, it uses well-scrutinized, well-trusted implementations of standard, open cryptographic algorithms and protocols. I explain how, in more detail, shortly. But for overview purposes, suffice it to say that OpenVPN represents a class of encapsulating SSL/TLS-based VPN tools and is one of the better examples thereof.

Some Linux VPN Choices

Nowadays, a number of good VPN solutions exist for Linux. Some commercial products, of course, release Linux versions of their proprietary VPN client software (so many more than when I began this column in 2000!).

In the IPsec space, there are Openswan, which spun off of the FreeS/WAN project shortly before the latter ended; Strongswan, another FreeS/WAN spin-off; and NETKEY (descended from BSD's KAME), which is an official part of the Linux 2.6 kernel and is controlled by userspace tools provided by the ipsec-tools package. All of these represent IPsec implementations for the Linux kernel. Because IPsec is an extension of the IPv4 protocol, *any* IPsec implementation on any operating system must be integrated into its kernel.

vpnc is an open-source Linux client for connecting to Cisco VPN servers (in the form of Cisco routers, Cisco ASA firewalls and so forth). It also works with Juniper/Netscreen VPN servers.



Although I don't recommend either due to PPTP's security design flaws, PPTP-linux and Poptop provide client and server applications, respectively, for Microsoft's PPTP protocol. Think it's just me? Both PPTP-linux's and Poptop's maintainers recommend that you not use PPTP unless you have no choice! (See Resources for links to the PPTP-linux and Poptop home pages.)

And, of course, there's OpenVPN, which provides both client and server support for SSL/TLS-based VPN tunnels, for both site-to-site and remote-access use.

Introduction to OpenVPN

All the non-PPTP Linux VPN tools I just mentioned are secure and stable. I focus on OpenVPN for the rest of this series, however, for two reasons. First, I've never covered OpenVPN here in any depth, but its growing popularity and reputation for security and stability are such that the time is ripe for me to cover it now.

Second, OpenVPN is much simpler than IPsec. IPsec, especially IPsec on Linux in either the client or server context, can be very complicated and confusing. In contrast, OpenVPN is easier to understand, get working and maintain.

Among the reasons OpenVPN is simpler is that it doesn't operate at the kernel level, other than using the kernel's tun and tap devices (which are compiled in the default kernels of most mainstream Linux distributions). OpenVPN itself, whether run as a VPN server or client, is strictly a userspace program.

In fact, OpenVPN is composed of exactly *one* userspace program, openvpn, that can be used either as a server dæmon for VPN clients to connect to or as a client process that connects to some other OpenVPN server. Like stunnel, another tool that uses SSL/TLS to encapsulate application traffic, the openssl dæmon uses OpenSSL, which nowadays is installed by default on most Linux systems, for its cryptographic functions.

OpenVPN, by the way, is not strictly a Linux tool. Versions also are available for Windows, Solaris, FreeBSD, NetBSD, OpenBSD and Mac OS X.

Getting OpenVPN

OpenVPN is already a standard part of many Linux distributions. Ubuntu, Debian, SUSE and Fedora, for example, each has its own "openvpn" package. To install OpenVPN on your distribution of choice, chances are all you'll need to do is run your distribution's package manager.

If your distribution lacks its own OpenVPN package, however, you can download the latest source code package from **www.openvpn.net**. This package includes instructions for compiling and installing OpenVPN from source code.

Conclusion

Now that you've got some idea of the uses of VPN, different protocols that can be used to build VPN tunnels, different Linux tools available in this space and some of the merits of OpenVPN, we're ready to roll up our sleeves and get OpenVPN running in both server and client configurations, in either "bridging" or "routing" mode.

But, that will have to wait until next month—I'm out of space for now. I hope I've whetted your appetite. Until next time, be safe!■

Mick Bauer (darth.elmo@wiremonkeys.org) is Network Security Architect for one of the US's largest banks. He is the author of the O'Reilly book *Linux Server Security*, 2nd edition (formerly called *Building Secure Servers With Linux*), an occasional presenter at information security conferences and composer of the "Network Engineering Polka".

Resources

Mick Bauer's Paranoid Penguin, January 2005, "Linux VPN Technologies": **www.linuxjournal.com/article/7881**

Wikipedia's Entry for IPsec: **en.wikipedia.org/wiki/IPsec**

Home Page for Openswan, an IPsec Implementation for Linux Kernels: **en.wikipedia.org/wiki/IPsec**

Home Page for Strongswan, Another Linux IPsec Implementation: **www.strongswan.org**

Home Page for pptp-linux (not recommended): **pptpclient.sourceforge.net**

Poptop, the PPTP Server for Linux (not recommended): **poptop.sourceforge.net/dox**

Tools and Papers Related to Moxie Marlinspike's SSL Attacks (and Others): **www.thoughtcrime.org/software.html**

"Major SSL Flaw Find Prompts Protocol Update", by Kelly Jackson Higgins, DarkReading: **www.darkreading.com/security/vulnerabilities/showArticle.jhtml?articleID=221600523**

Official OpenVPN Home Page: **www.openvpn.net**

Ubuntu Community OpenVPN Page: **https://help.ubuntu.com/community/OpenVPN**

Charlie Hosner's "SSL VPNs and OpenVPN: A lot of lies and a shred of truth": **www.linux.com/archive/feature/48330**