MICK BAUER

Linux VPNs with OpenVPN, Part V

Tips for success in using OpenVPN for secure remote access.

In my four previous columns, I showed, in painstaking detail, how to set up OpenVPN to allow remote users to create secure remote-access connections—Virtual Private Network (VPN) tunnels—over the Internet back to your personal or corporate network. By now, you should understand how VPN technologies in general, and TLS/SSL-based VPNs in specific, work and how to create working server and client configurations for OpenVPN.

This month, I wrap up the series, with some miscellaneous but important notes about the previous columns' client-server scenario, including instructions on enabling IP forwarding, some tips on using a Web proxy and enforcing DNS use through the tunnel, and on "hiding" all VPN clients' IP addresses behind that of your OpenVPN server.

Listing 1. Server's server.ovpn File

```
port 1194
proto udp
dev tun

ca 2.0/keys/ca.crt
cert 2.0/keys/server.crt
key 2.0/keys/server.key  # This file should be kept secret
dh 2.0/keys/dh1024.pem
tls-auth 2.0/keys/ta.key 0

server 10.31.33.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway def1 bypass-dhcp"

keepalive 10 120

cipher BF-CBC        # Blowfish (default)
comp-lzo
max-clients 2

user nobody
group nogroup
persist-key
persist-tun

status openvpn-status.log
verb 3
mute 20
```

Review

Throughout this series, I've been implementing the OpenVPN server configuration shown in Listing 1, which causes OpenVPN to run in server mode. In my example scenario, I've got only one remote user connecting to this OpenVPN server, but if you have more, you should edit the `max-clients` parameter accordingly. Remember, because I've also set fairly liberal tunnel timeouts in order to minimize the odds that a tunnel will go down due to network problems, you should add 1 or 2 to the actual number of maximum concurrent client connections you expect.

The other setting in Listing 1 that I need to review is `push "redirect-gateway def1 bypass-dhcp"`, which pushes the OpenVPN's local default gateway setting to all clients. This has the effect of causing VPN clients to route all their Internet traffic through the VPN tunnel, which (as I discuss shortly) has important security benefits.

The client configuration file that corresponds to Listing 1 is shown in Listing 2. This file works equally well on Linux and Windows client systems. Remember that the parameter `remote` specifies the IP address or hostname of your OpenVPN server and the port on which it's accepting connections.

Remember also that the files ca.crt, minion.crt, minion.key and ta.key specified by the parameters `ca`, `cert`, `key` and `tls-auth`, respectively, need to be generated beforehand and placed alongside the configuration file itself in /etc/openvpn. The certificate and key specified by `ca` and `cert` should be unique for each client system!

Again, the purpose of the server configuration in Listing 1 and the client configuration in Listing 2 is to allow a remote user to connect from over the Internet back to the "home" network on which the OpenVPN server resides. (This may or may not be your residence. By home network, I mean "trusted corporate or personal network", as opposed to the remote network from which you're trying to connect.) Last month, however, I forgot to mention a critical step that you must perform on your OpenVPN server if you want remote clients to be

Listing 2. Client's client.ovpn File

```
client
proto udp
dev tun

remote 1.2.3.4 1194

nobind

ca ca.crt
cert minion.crt
key minion.key

ns-cert-type server
tls-auth ta.key 1

cipher BF-CBC
comp-lzo

user nobody
group nogroup
persist-key
persist-tun

mute-replay-warnings

verb 3
mute 20
```

able to communicate with anything besides the server itself: enabling IP forwarding.

Enabling IP Forwarding

By default, almost any Linux system is configured not to allow network packets entering one network interface to be forwarded to and sent out of a different network interface. This is a Linux security feature. It helps reduce the likelihood of your Linux system linking different networks together in undesirable or unintended ways.

But, generally you do want an OpenVPN server to link different networks. The exceptions to this are if:

1. All services and resources your remote users need are housed on the OpenVPN server itself.

2. It's possible to run proxy applications on the OpenVPN server that can proxy connections to services not hosted on it.

In the first case, once remote users have connected to the OpenVPN server successfully, they can connect to other services hosted on that server by targeting the server's real/local IP address rather than its Internet-facing address. For example, the client configuration in Listing 2 is targeting a server address of 1.2.3.4, which is Internet-routable. Suppose that this is actually a router or firewall address that is translated to your OpenVPN server's address 10.0.0.4.

To `ssh` to the OpenVPN server after you've established a tunnel to it, you'd target 10.0.0.4, not 1.2.3.4. The same would apply to Samba, NFS, HTTP/S or any other service running on the OpenVPN server.

In the second case, to reach other resources on the remote network, you would configure the applications running on your client system to use the OpenVPN server's real/internal address (10.0.0.4) as its proxy address. The best example of this is Squid. If all the resources you wanted to reach on your remote network involve Web services, you could run Squid on the OpenVPN server and configure your client's Web browser to use 10.0.0.4 as its proxy address (although this will work only when the tunnel is up).

In either of the above scenarios, you *don't* need IP forwarding enabled on the OpenVPN server, because all direct communication between VPN clients and your home network terminates on the OpenVPN server. If, however, your clients need to reach other things on the home network or beyond, *without* using the OpenVPN server as a proxy, you do need to enable IP forwarding.

This is very simple. To turn on IP forwarding without having to reboot, simply execute this command:

```
bash-$ sudo sysctl -w net.ipv4.ip_forward=1
```

To make this change persistent across reboots, uncomment the following line in /etc/sysctl.conf (you'll need to `su` to root or use sudo to edit this file):

```
net.ipv4.ip_forward=1
```

Web Proxies and VPN Clients

In talking about the value of using VPN software when using untrusted networks like WLAN hot spots, I've described the benefits of using your home network's Web proxy rather than surfing the Web directly through the untrusted network. From a policy-enforcement standpoint, this allows you to enforce whatever URL or content filtering with which your home network's proxy is configured; from an endpoint-security standpoint, it makes phishing and man-in-the-middle attacks harder.

On the downside, it also results in a somewhat slower Web browsing experience, because each user's Web traffic must traverse a longer, slower path than without the VPN tunnel in place. Also, making remote users use your corporate Web proxy without also configuring them to use your corporate DNS servers may fail to prevent man-in-the-middle attacks (in which DNS redirection is a common technique), giving a false sense of security.

I return to the DNS problem shortly, but how do you use Web proxies with OpenVPN? It's quite

simple. On the Web proxy itself, you simply need to make sure there's an Access Control List (ACL) allowing client connections from tunnel IPs. This is a moot question if your Squid server is running on a different box from the OpenVPN server, *and* the OpenVPN server is using Network Address Translation (NAT) to "hide" all tunnel-originated packets behind its own IP address (I discuss NAT shortly).

If, however, you are running the Web proxy on the OpenVPN server itself, you need an ACL. For Squid, you need to add something like this to /etc/squid/squid.conf:

```
acl openvpn_tunnels src 10.31.33.0/24
http_access allow openvpn_tunnels
```

The `acl` line defines an object named `openvpn_tunnels`, representing transactions whose source IP addresses fall between 10.31.33.1 and 10.31.33.254. The `http_access` line allows transactions initiating from this IP range. As with any other change you make to this file, you need to restart Squid for this ACL to take effect (`sudo /etc/init.d/squid restart`).

Your clients will, of course, need to be configured to use your Web proxy, but they target the same IP address regardless of whether they're connecting from afar via OpenVPN or connecting directly to your LAN. That is, if you're already having your users proxy all their Web traffic, no change to their Web browser settings should be necessary for them to use the same proxy through OpenVPN.

Enforcing DNS

If you're requiring all remote users to route all their Internet traffic through the VPN tunnel, it isn't enough to force them to use the remote network's default gateway. You also need to force them to use the remote network's DNS servers. Otherwise, a man-in-the-middle attack that involves DNS spoofing on the client side of the tunnel will succeed. Once a remote user's browser has been handed a phishing site's IP address for a given URL, it doesn't matter whether it connects to that IP directly or through the VPN tunnel (unless, perhaps, the phishing site's IP address is on a blacklist enforced by your corporate Web proxy or firewall).

If your remote clients all run Windows, it's easy to enforce server-side DNS settings. Simply add the following line to your OpenVPN server's OpenVPN configuration file:

```
push "dhcp-option DNS 10.0.0.100"
push "dhcp-option DNS 10.0.0.120"
```

Of course, you should replace 10.0.0.100 and 10.0.0.120 with the addresses of the DNS servers you want your clients to use.

Unfortunately, this won't work for non-Windows clients. For Linux and other UNIX clients, you'll need to edit those client systems' /etc/resolv.conf files either manually or dynamically. The server-side configuration parameter `foreign_option_<I>n<I>` lets you pass data to tunnel-initiation scripts (`--up` scripts); for example, the line `foreign_option_1='dhcp-option DNS 10.0.0.100'` sends the line `dhcp-option DNS 10.0.0.100` to any defined "up" scripts, which can then act on that data.

The details of how all this works are out of the scope of this article. Suffice it to say that the OpenVPN man page describes how "up" scripts work, and the link to the update-resolv-conf script in the Resources for this article provides a script you can alter to rewrite /etc/resolv.conf to give precedence to your "home" network's DNS servers.

NAT and iptables on the OpenVPN Server

There's one more critical step necessary to allow remote users to route packets to the Internet through their VPN tunnels. You need to set up Network Address Translation (NAT) so that traffic entering your "home" network from VPN tunnels appears to originate from the OpenVPN server.

This is because the networks from which remote clients connect will have either different network IP addresses than your "home" network, in which case the odds are your "home" network infrastructure won't have a route to the remote clients, or they'll have the same network IP addresses, in which case it's quite possible that different hosts on opposite ends of the VPN tunnels will have the *same* host IP addresses!

Note that this problem plays out differently on "bridging" (Layer 2) VPN tunnels than on "routing" (Layer 3) VPN tunnels. Because all my examples so far have involved a routing VPN scenario, what I'm about to say regarding NAT applies to routed VPN tunnels.

So, the way to sidestep the problem of foreign IP addresses on remote clients' packets completely is simply to rewrite all packets entering the OpenVPN server's local network with the OpenVPN server's local IP address. To do so, add just one firewall rule, like this:

```
bash-$ sudo iptables -t nat -A POSTROUTING
➥-s 10.31.33.0/24 -o eth0 -j MASQUERADE
```

Note that as with any other time you execute the command `iptables`, this is not a persistent change. To make this rule persistent across reboots, you need to add an equivalent line to whatever configuration file or script controls firewalling on your OpenVPN server.

The OpenVPN man page has an entire section on firewalls (called "FIREWALLS") that contains lots of good information about managing iptables firewall rules on your OpenVPN server. Remember, any

VPN server is a security device. It's a good idea to run not just a single NAT rule, but a detailed set of filtering rules that restrict how people can connect to the server and to what systems your VPN clients may connect.

(Speaking of iptables, it's been a long time since I covered Linux's powerful firewall capabilities in this column. Look for an in-depth article on writing your own Linux firewall rules in a future column.)

Conclusion

This article and my previous four columns covered Virtual Private Network principles and architectures; described a few VPN technologies available for Linux and how SSL/TLS solutions differ from IPsec; covered OpenVPN server configuration, including how to generate and manage digital certificates; and described client configuration and usage; all for a simple remote-access usage scenario.

With all of that plus the practical use details I covered this month, you should be well on your way to a secure remote-access VPN solution using OpenVPN. If you decide to use OpenVPN instead or additionally to build network-to-network VPNs or to do a "bridging" OpenVPN solution, the OpenVPN man page, HOWTO and FAQ should make more sense to you now than they would have before reading these articles—all of which means, you no longer have any excuse to surf the Web through insecure wireless hot spots without protection!■

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

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

OpenVPN FAQ: **openvpn.net/index.php/open-source/faq.html**

OpenVPN HOWTO: **www.openvpn.net/index.php/open-source/documentation/howto.html**

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

The update-resolv-conf Script by Thomas Hood and Chris Hanson: **www.subvs.co.uk/openvpn_resolvconf**