The Sysadmin's Daily Grind: OpenVPN

Secure Connections

Being able to work anywhere in the world just as if you were attached to the company's LAN is an appealing prospect, but far too dangerous without taking security measures. BY CHARLY KÜHNAST

The solution to this issue is well known: Use an encrypted point-to-point connection to build a tunnel through an insecure network. For Linux there are a few popular tools available to help you with this task, such as Cipe and the freeware IPSec implementation, FreeS/WAN, and of course OpenVPN [1] the tool we will be investigating in this issue.

Used: OpenSSL and TUN/TAP

The prerequisites for running OpenVPN are the OpenSSL library and the TUN/TAP driver. Any of the current distribution should contain both – if not, check out [2] and [3]. Just say the magic words, *./configure; make; make install*, to unpack the tarball, which is slightly over 200 KBytes.

To demonstrate the principle of OpenVPN let's open up an unencrypted tunnel between two computers called *left* and *right*. The IP addresses of the ethernet interfaces are 1.2.3.4 for *left* and 4.3.2.1 for *right*.

We can now assign private IP addresses to the endpoints of the tunnel on both computers (*/dev/tun0*). So let's assign 10.0.0.1 to the endpoint on *left* and 10.0.0.2 to the corresponding endpoint on *right*. Before we get started, we need to load the TUN driver.

```
modprobe tun
```

should do the trick. We will also need to enable IP forwarding using the following syntax:

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

We can now type the following command for *left*

```
openvpn --remote right --dev ➥
tun0 --ifconfig 10.0.0.1 ➥
10.0.0.2
```

and a similar command for *right*

```
openvpn --remote left --dev ➥
tun0 --ifconfig 10.0.0.2 ➥
10.0.0.1
```

The tunnel is up and running.

Encrypting the tunnel

For test purposes we can simply ping the IP addresses of the computers at the opposite ends of the tunnel, i.e. we *ping 10.0.0.2* on *left* and vice-versa. If this simple test does exactly what we expect, we can go on to encrypt the tunnel. The simplest way of doing this is to agree on a shared secret. To do so, we simply type

```
openvpn --genkey --secret key.
```

on one of the computers. This creates a *key* file containing random data that still needs to be copied securely to the second computer – we can use *scp* for this purpose.

Let's start up the tunnel using the following syntax for *left*:

```
openvpn --remote right --dev ➥
tun0 --ifconfig 10.0.0.1 ➥
10.0.0.2 --secret key
```

and the following syntax for *right*:

```
openvpn --remote left --dev ➥
tun0 --ifconfig 10.0.0.2 ➥
10.0.0.1 --secret key
```

All done! Of course, a shared secret is not all that secure, if you are in a tight corner. So I would not recommend using this method for official secrets. If you need to keep data secret, you might want to opt for a TLS based approach – OpenVPN offers you that possibility. ■

INFO

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

[2] OpenSSL: *http://www.openssl.org*

[3] TUN/TAP Drivers: *http://vtun.sourceforge.net/tun*

THE AUTHOR

Charly Kühnast is a Unix System Manager at a public datacenter in Moers, near Germany's famous River Rhine. His tasks include ensuring firewall security and availability and taking care of the DMZ (demilitarized zone). Although Charly started out on IBM mainframes, he has been working predominantly with Linux since 1995.