

OpenBSD as a VPN Solution

Alex Withers

One of the most pertinent topics in the networking industry today is Virtual Private Networks, or VPNs. In short, VPNs arose out of a need for a cheaper method to connect networks and hosts while still maintaining security and privacy. Many businesses found that in order to connect their branches they had to invest in expensive equipment and leased lines. The same went for providing connections to remote clients; however, in this case the cost went into dedicated modem lines. While methods like these provided for a relatively secure solution, they also tended to be expensive on the implementation and maintenance side. The Internet is an obvious solution to the cost issue by providing one giant medium for connection at a reasonable price (at least when compared with the other solutions).

Even though the Internet can be used to get around the cost issue, we still have to deal with security. It is here that VPNs play a part by protecting the data transmitted on the Internet. VPNs essentially form an artificial network on an already existing public network. The technology can authenticate, encrypt data, and provide a secure tunnel between two or more points. As a result, VPNs are being used more often for a cost-effective and secure method to communicate.

Many VPN solutions are emerging on the market today, and many of these are offered by big name companies. The technology is often proprietary and very expensive, incurring costs on the software and hardware level. However, since VPNs are more of a concept than an actual technology, we are allowed much working room to implement it. Software such as SSH can be used to create VPNs and there are free Open Source solutions for the free UNIX variants. While these solutions may work fine, they can also be somewhat complex in the implementation process.

Another solution involves the OpenBSD operating system. In the case of VPNs, you can take advantage of some of the features OpenBSD has to offer natively. The OpenBSD solution uses a relatively new technology called IPSec. IPSec was designed to secure the existing IP technology. It is independent of any application since it works at a lower level. This is rather convenient, because all data sent is encrypted without having to configure any applications.

Implementing the VPN

Even within OpenBSD, there are several methods available to deploy IPSec. In this article, I will use the **isakmpd** daemon. This daemon allows for a secure method to exchange keys between two hosts; however, the configuration can be more sophisticated than “manual keying”. All of the following examples were developed using OpenBSD 2.6, and they may need to be modified for earlier or later versions of OpenBSD.

Enabling IPSec in OpenBSD

Two variables must be set in order to enable IPSec. They are not set by default, but the following two commands will do the trick:

```
sysctl -w net.inet.esp.enable=1
sysctl -w net.inet.ah.enable=1
```

The **sysctl** command is used to change the state of the kernel, and in this case we are enabling AH and ESP. (See the article by Michael Lucas in this issue for more information about **sysctl**.) You may be wondering what AH and ESP are and what they have to do with IPSec. AH authenticates, but does not encrypt, the data. ESP will authenticate *and* encrypt the data sent through the network. Both of these are protocols of IPSec and, in the case of **isakmpd**, should both be enabled. If you wish to have this done each time the system boots, then you need to edit the **/etc/sysctl.conf** file. To do so, simply add the following two lines:

```
net.inet.esp.enable=1
net.inet.ah.enable=1
```

To disable it during boot time, simply change the 1's to 0's or comment it out with the # symbol. The VPN **man** page strongly recommends that you have inbound packet verification enabled. To do this, issue the following command:

```
sysctl -w net.inet.ip.ipsec-ac1=1
```

If the host will be a gateway, then you must make sure IP forwarding is enabled. This is yet another **sysctl** command:

```
sysctl -w net.inet.ip.forward=1
```

Add these to the **sysctl.conf** like the first two tags in order for this to take place during boot time. Once everything has been enabled, IPSec is ready to be configured.

The isakmpd Daemon

OpenBSD 2.6 should come with the **isakmpd** binaries without having to take any further steps after installation. When dealing with **isakmpd**, there are two main files that will occupy our attention:

```
/etc/isakmpd/isakmpd.conf
/etc/isakmpd/isakmpd.policy.
```

The first of these is the configuration file, which determines the behavior for **isakmpd**. The latter file is the policy file, which tells how to authenticate connections. These two files are very important because they are the keys to creating the VPN.

Configuring isakmpd

Step One: isakmpd.policy

Our scenario for discussion involves three networks: 10.0.1.0, 10.0.2.0, and 10.0.3.0. Each of these networks has a main router: 10.0.1.1, 10.0.2.1, and 10.0.3.1. These networks will communicate using the “outside” network: 192.168.1.0. In the real world, this “outside” network may very well be the Internet. In this example, each router will be connected to the “outside” network with the following IP addresses: 192.168.1.1, 192.168.1.2, and 192.168.1.3. Each of these routers is running OpenBSD 2.6, and they will be the boxes configured for this example of a simple VPN.

Unfortunately, the default installation of OpenBSD does not come with the **isakmpd.policy** or **isakmpd.conf** files. However, there are example files available if you have the official OpenBSD CD-ROM or can get them through CVS. Under the **/src/sbin/isakmpd/samples** directory, there are two files that can be useful: **policy** and **VPN-east.conf**. Simply copy the file **policy** to **/etc/isakmpd/** and **isakmpd.policy** and **VPN-east.conf** to **/etc/isakmpd/isakmpd.conf**. They will be the files dealt with in this article. In this example, they will exist on all three machines.

First, edit the **isakmpd.policy** file to specify who can connect using IPSec. If we wanted to allow anyone to connect, then **isakmpd.policy** would look like this:

```
KeyNote-Version: 2
Comment: A policy which accepts anyone.>
Authorizer: "POLICY"
```

This would probably be useful only for testing purposes, as it is not very secure. For something more secure, we could use the following:

```
KeyNote-Version: 2
Comment: A policy which accepts anyone \
that uses "yoursecret" as a passphrase.
Authorizer: "POLICY"
Licensees: "passphrase:yoursecret"
Conditions: app_domain == "IPsec policy" &&
```

```
esp_present == "yes" &&
esp_enc_alg != "null" -> "true";
```

Note that a lot more can be done with this file, and you should refer to the **isakmpd.policy man** pages for more information. We could use digital certificates or other such things for authentication. I will keep this example simple and use a passphrase.

Step Two: isakmpd.conf

The next step is to modify the **isakmpd.conf** file. Since we have three networks between which we would like to establish a VPN, we need to tell each router about the other two routers. But before we do this, each **isakmpd.conf** needs to have the following:

```
[General]
Retransmits=      3
Exchange-max-time= 60
Listen-on=        192.168.1.x
```

The first variable, **Retransmits**, tells **isakmpd** how many times a message should be retransmitted before it gives up. The second variable, **Exchange-max-time**, gives the number of seconds available for an exchange to set up. The third variable, **Listen-on**, gives the IP address of the external interface and, in this example, the **x** should be replaced with 1,2, and 3 for each machine, respectively. There are many more parameters available under the General section (see the **isakmpd.conf(5) man** page). These values should be tweaked, since every network and situation is different.

When establishing an IPSec connection between two nodes, there are two phases. The first phase establishes the secure connection, and the second phase does the rest of the work by establishing the tunnel in a secure connection. Therefore, the next section in the configuration file specifies the daemon from which it can accept connections:

```
[Phase 1]:
192.168.1.2=      host2
192.168.1.3=      host3
```

The above would have been what host1 (192.168.1.1) needs in its configuration file. Thus, in this scenario, every host would need the other two hosts listed. Now, we add a section for the second phase:

```
[Phase 2]:
Connections=      host1-host2
Connections=      host1-host3
```

This section has also been taken from what would have been the configuration for host1, and describes the possible connections. Note that the right-hand side values are quite meaningless, since they are nothing but strings that signify the name of the next sections. It is after the phase 2 section that these sections are defined. Continuing with host1, we add the next section to the configuration file:

```
[host2]
Phase=            1
Transport=        udp
Address=          192.168.1.2
Configuration=    MainMode
Authentication=   yoursecret

[host3]
Phase=            1
Transport=        udp
Address=          192.168.1.3
Configuration=    MainMode
Authentication=   yoursecret
```

Since we are dealing with host1, we listed the two other nodes. The **Transport** flag tells **isakmpd** which protocol to use. The **Address** flag gives the IP of the remote node. The **Configuration** flag points to another section, and the **Authentication** flag is the passphrase matched with the policy found in **isakmpd.policy**.

Note that this setup is similar to the example configuration file provided by OpenBSD. Also note that in regard to the **Transport** flag, some firewalls won't allow UDP packets, so some change would be necessary. Next, we define the sections referred to in the phase 2 section:

```
[host1-host2]
Phase=            2
ISAKMP-peer=      host2
Configuration=    QuickMode
Local-ID=         Net1
Remote-ID=        Net2

[host1-host3]
Phase=            2
ISAKMP-peer=      host3
Configuration=    QuickMode
Local-ID=         Net1
Remote-ID=        Net3
```

Remember that we are still dealing with host1's configuration. The **ISAKMP-peer** flag (host2) refers to the remote node. The **Configuration** flag refers to a section further below in the configuration file. The **Local-ID** and **Remote-ID** flag refers to the sections that describe the networks we are dealing with. We would define those sections in host1's configuration file as follows:

```
[Net1]
ID-type=          IPV4_ADDR_SUBNET
Network=          10.0.1.0
Netmask=          255.255.255.0

[Net2]
ID-type=          IPV4_ADDR_SUBNET
Network=          10.0.2.0
Netmask=          255.255.255.0

[Net3]
ID-type=          IPV4_ADDR_SUBNET
Network=          10.0.3.0
Netmask=          255.255.255.0
```

The **ID-type** describes the node and might be **IPV4_ADDR** instead. The **Network** and **Netmask** flag give the appropriate numbers to describe the private network.

Now we need to define the MainMode and QuickMode. These describe the encryption methods for the first and second phases, respectively.

```
[MainMode]
DOI=              IPSEC
EXCHANGE_TYPE=    ID_PROT
Transforms=       BLF-SHA-M1024

[QuickMode]
DOI=              IPSEC
EXCHANGE_TYPE=    QUICK_MODE
SUITES=           QM-ESP-3DES-SHA-PFS-SUITE
```

The **Transform** tag in MainMode tells **isakmpd** to use the blowfish algorithm for encryption and the SHA hashing algorithm. There are many to choose from, and

the example configuration in the **isakmpd.conf** man page lists some. The **SUITES** tag in the QuickMode section points to predefined sections, but it basically tells **isakmpd** to use the ESP protocol, Triple DES for encryption, and the SHA hashing algorithm. These tags, **Transform** and **SUITES**, point to predefined sections, which can be found in the OpenBSD's sample configuration file or the **isakmpd.conf** man page, but they aren't necessary for the configuration file.

Preparing for and Running isakmpd

If the machine is a gateway to a network, then it's a good idea to make it a firewall. All incoming packets, except those from the VPN, should be blocked. To do this, open a plain text file for the firewall rules:

```
block in log on <network-interface> \
  from any to any
block out log on <network-interface> \
  from any to any

block in log on enc0 from any to any
block in log on enc1 from any to any

pass in proto esp from 192.168.2.1/32 \
  to 192.168.1.1/32
pass in proto esp from 192.168.3.1/32 \
  to 192.168.1.1/32
pass out proto esp from 192.168.1.1/32 \
  to 192.168.2.1/32
pass out proto esp from 192.168.1.1/32 \
  to 192.168.3.1/32

pass in on enc0 from 10.0.2.0/24 to 10.0.1.0/24
pass in on enc1 from 10.0.3.0/24 to 10.0.1.0/24
```

This set of firewall rules would be used on host1. The rules are quite easy to read, and it's not hard to understand what is going on here. Note that **<network-interface>** should be changed to the external interface of the machine (the interface connected to the external network). Also, enc0 and enc1 are the IPSec interfaces from which host1 receives the packets.

The above rules will block any incoming traffic, but we need to make sure a connection can be established. Since **isakmpd** works on port 500 using the UDP protocol, incoming packets need to be allowed in:

```
pass in proto udp from 192.168.2.1/32 \
  to 192.168.1.1/32 port = 500
pass in proto udp from 192.168.3.1/32 \
  to 192.168.1.1/32 port = 500
```

Because I am still using host1 as an example, I am allowing UDP traffic on port 500 from 192.168.2.1 and 192.168.3.1. Now that you have compiled these rules (and any others you might need) in a text file, you must pipe it to OpenBSD's firewall tool, **ipf**:

```
ipf -f textfile
```

If you want this done every time during boot, then add **ipfilter=YES** to the **/etc/rc.conf** file. Now, **ipf** will read the **/etc/ipf.rules** during boot time.

The next step is to run the daemon. To run the daemon in the foreground, simply use the **-d** flag as follows:

```
isakmpd -d
```

Using the **-DA=99** option will set all debugging classes to level 99. This will provide quite a bit of information. If **isakmpd** is not started during boot time automatically, then edit **/etc/rc.local**. You can now test your setup by sending traffic between the two nodes and using **tcpdump** to see whether the traffic contains ESP or AH packets.

Conclusion

The examples I used here were for a simple VPN, which worked well enough for my employer, but **isakmpd** is extremely flexible. There are many more options to use in **isakmpd** for authentication, such as X509 digital certificates. **isakmpd** is also known to work with many popular VPN clients, which are listed in the OpenBSD FAQ. I strongly suggest going through all the man pages for **isakmpd**, **ipsec**, **vpn**, etc. I also suggest reading the OpenBSD FAQ. These were my primary resources, and they contain many more examples, especially with digital certification.

Since OpenBSD tightly integrates cryptography, the solutions for a VPN are myriad and easy to deploy. OpenBSD is also very secure and rock stable, and with support of hardware encryption just around the corner, OpenBSD can be very attractive in many situations. n

About the Author

Alex Withers (awithers@gonzaga.edu) is currently a student at Gonzaga University in Spokane, WA. When not sitting through the riveting lectures, Alex either works to administer networks or enjoys running.