MINI-FAQ: OpenBSD 2.4 IPSEC VPN Configuration

Maintainer: Steve McQuade <smcquade@codetalker.com>

v1.07 - March 2, 1999

After trying to configure an OpenBSD 2.4 IPSEC based VPN based on the samples and documentation provided with the OpenBSD 2.4 release, I decided that a mini-faq would be a good idea. Hopefully, others will too. **Thanks to:** Steve McQuade, Kjell Wooding, Matt Zimmerman, Niels Provos, Angelos D. Keromytis, Niklas Hallqvist and many others...

1.0 - IPSec and VPN Background

2.0 - IPSec VPN Configuration

3.0 - Changelog

1.0 - IPSec and VPN Background

1.1 - Glossary of Terms

- **Security Association (SA)** - Identified by a unique triple of IP Address, SPI (numeric ID) and security protocol (e.g. ESP, AH). Specifies the parameters for communication with the specified host
- **Security Parameter Index (SPI)** - Used to uniquely identify which SA should be applied to a packet
- **Transport SA** - SA used when encapsulating transport layer (e.g. TCP, UDP) datagrams
- **Tunnel SA** - SA used when encapsulating network layer (in this case, IP) datagrams

1.2 - What is a VPN?

From the vpn(8) man page:

> A virtual private network is used to securely connect two or more subnets over the internet. For each subnet there is a security gateway which is linked via a cryptographically secured tunnel to the security gateway of the other subnet. ipsec(4) is used to provide the necessary network-layer cryptographic services. This document describes the configuration process for setting up a VPN.

1.3 - In a nutshell, how do I create an IPSec tunnel between two OpenBSD boxes?

Briefly:

- Create a Security Association (SA) for each side of the connection
- Create a shared secret and initialization vector (or use a key management daemon such as *photurisd* or *isakmpd*)
- Create the appropriate IPSec flows between your OpenBSD boxen.

1.4 - What do *photurisd* and *isakmpd* do for me?

These are both key management daemons. They eliminate the need to exchange a shared secret before communications can begin. Sadly, they don't yet work in every situation. See 1.5 for details.

1.5 - How can I set up Photurisd to work with private networks?

Private networks cannot be used in a VPN with the current version of Photuris. This is due

to a limitation that the remote firewall address (the "-dst" argument) lies inside the remote subnet range. Basically this means that if the remote internal network is private (i.e. non-routable, or nat'ed in some other way), then one of the private addresses would then need to be exposed to the Internet.

Since Photuris is only being used for key-exchange, it is not absolutely necessary in order to implement a VPN using private networks. You can still use symmetric shared secret keys for authentication and encryption.

And finally, from the most recent (post 2.4) vpn(8) man page:

> **BUGS**
> When using photurisd(8) in VPN mode, both of the security gateways IP addresses must fall within their protected netranges. In situations where the gateway IP is outside the desired netrange, such as with private networks (RFC 1597), manual keying must be used. This should be fixed in the next release.

1.6 - Blowfish and Cast have variable key lengths. Does a larger key length equal higher security, shorter key length lower security ?

Taken from the OpenBsd-Misc mailing list:

> Yes, to some extend; the point with variable key-length ciphers is that in the future (when Pentium17 comes out, which can crack DES in less time than you need to ping 127.0.0.1) you won't have to design a new algorithm.
>
> With today's technology, 128 bits are considered sufficient for the next 10 years. The difference between 256 and 300 bits is just theoretical :-)
>
> -*Angelos D. Keromytis* <angelos@dsl.cis.upenn.edu>

1.7 - Where can I get more information on IPSec and IPSec VPNs?

OpenBSD Man Pages:

- vpn(8)
- ipsec(4)
- ipsecadm(1)
- photurisd(8)
- isakmpd(8) [-*current only*]

RFCs and Other Standards:

- IPSec Working Group Archive
- RFC2411 - IP Security Document Roadmap
- RFC2401 - Security Architecture for the Internet Protocol
- RFC2406 - Encapsulating Security Payload (ESP)
- Draft - Photuris: Session-Key Management Protocol
- And my personal favorite.

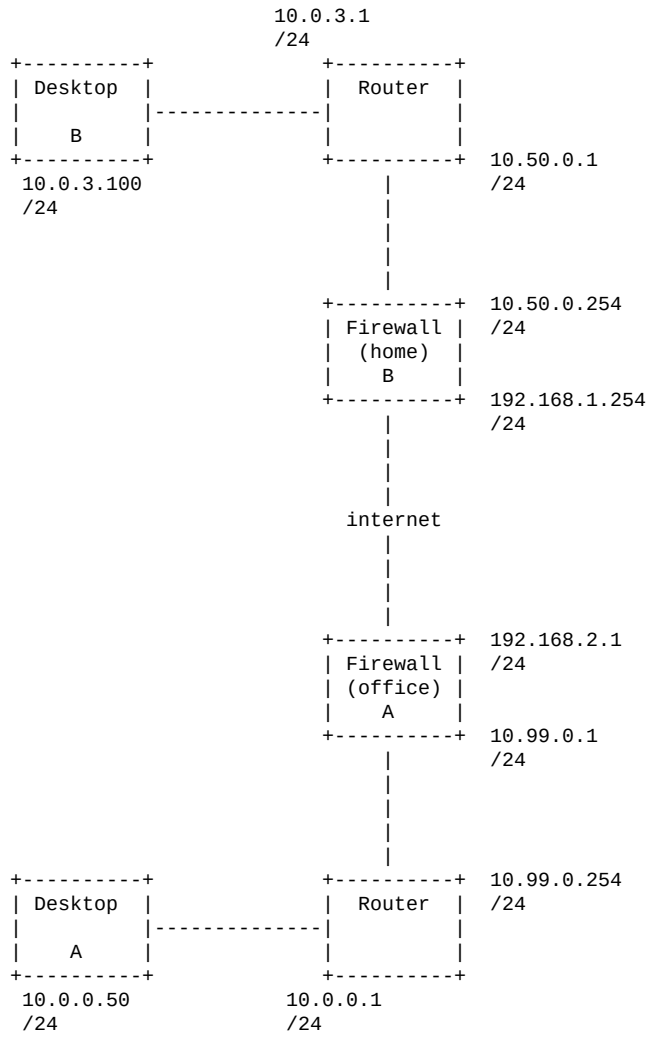
Other References:

- OpenBSD Journal article on IPSec
- How to use Photuris with IPSec

2.0 - IPSec and VPN Configuration

2.1 - Sample Network Diagram for Examples in this FAQ

This diagram is implicitly referred to throughout the document...



How do I create shared secret keys?

The secret symmetric key used for encryption and authentication can be any hexadecimal string of your choice. There are no special requirements or conditions, except that the key should be as random as possible. Depending on the cipher chosen, the key length will change. The following table provides common ciphers and key lengths:

Cipher Type	Key Length
DES	64-bits, 8 bytes

Note: For CBC based ciphers, the initialization vector (IV) must be an 8 byte hexadecimal offset. If none is specified, ipsecadm will derive one for you.

The secret symmetric key cipher and key is specified using *ipsecadm*. You must specify a valid length key for a corresponding cipher.

For example,

```
DES
key=D69403E2673E611D
iv =4CBD3FAD6FD1788E

3DES
key=596A96CC7BF9108CD896F33C44AEDC8AA8ACF0B8C74ACD62
iv =CD28C327C7FD0943

BLF
key=99754106633F94D350DB34D548D6091A
iv =1AB93C2692A0A046
```

2.3 - How do I create authentication keys?

The authentication keys that you will use can be any hexadecimal string of your choice. There are no special requirements or conditions, except that the key should be as random as possible. Depending on the authentication method chosen, the key length will vary. The following table provides common authentication schemes and key lengths:

The authentication key is used when setting up the IPSEC routes using *ipsecadm*. You must specify a corresponding authentication scheme using the -key modifier.

For example,

```
-key sha1 -authkey c9fff55b501206a6607fb45c392c5e1568db2aaf
-key md5 -authkey 926dd13f324733014851dcfdb50407de
```

How do I set up the Security Associations?

You will need to define two Security Associations (SA's) on each end of the VPN. The format of these commands is as follows:

format of these commands is as follows:

```
ipsecadm new esp -spi SPI_OUT -src MY_EXTERNAL_IP
         -dst PEER_EXTERNAL_IP
         -tunnel MY_EXTERNAL_IP PEER_EXTERNAL_IP
         -enc blf -auth sha1 -iv INITIALIZATION_VECTOR
         -key ENCRYPTION_KEY -authkey AUTHENTICATION_KEY

ipsecadm new esp -spi SPI_IN -src PEER_EXTERNAL_IP
         -dst MY_EXTERNAL_IP
         -tunnel PEER_EXTERNAL_IP MY_EXTERNAL_IP
         -enc blf -auth sha1 -iv INITIALIZATION_VECTOR
         -key ENCRYPTION_KEY -authkey AUTHENTICATION_KEY
```

The SPI_OUT SPI will be used for communication from A->B, and SPI_IN will be used for communication from B->A.

For simplicity, we'll use use the same SA configuration on both ends of the VPN (lines have been split for easy copy/pasting)

```
/sbin/ipsecadm new esp -src 198.168.2.1 -dst 198.168.1.254  \
  -tunnel 198.168.2.1 198.168.1.254 \
  -spi 1000 -enc 3des -auth sha1 -iv CD28C327C7FD0943 \
  -key 596A96CC7BF9108CD896F33C44AEDC8AA8ACF0B8C74ACD62 \
  -authkey c9fff55b501206a6607fb45c392c5e1568db2aaf


/sbin/ipsecadm new esp -src 198.168.1.254 -dst 198.168.2.1  \
  -tunnel 198.168.1.254 198.168.2.1 \
  -spi 1001 -enc 3des -auth sha1 -iv CD28C327C7FD0943 \
  -key 596A96CC7BF9108CD896F33C44AEDC8AA8ACF0B8C74ACD62 \
  -authkey c9fff55b501206a6607fb45c392c5e1568db2aaf
```

How do I set up the IPSEC routes using the SPIs?

The IPSEC routes use the SPIs configured within the SA to determine where to send the IPSEC traffic, and what encryption and authentication schemes to use. The *flow* command creates a flow that determines which packets are routed via which SA. You can use the *netstat -rn* command to view existing flows.

On the office firewall (firewall A):

```
# Set up the IPSEC routes on Firewall A
/sbin/ipsecadm flow -dst 192.168.1.254 -spi 1000 \
  -addr 192.168.2.1 255.255.255.255 192.168.1.254 255.255.255.255 -local

/sbin/ipsecadm flow -dst 192.168.1.254 -spi 1000 \
  -addr 10.0.0.0 255.255.255.0 10.0.3.0 255.255.255.0

/sbin/ipsecadm flow -dst 192.168.1.254 -spi 1000 \
  -addr 192.168.2.1 255.255.255.255 10.0.3.0 255.255.255.0 -local

/sbin/ipsecadm flow -dst 192.168.1.254 -spi 1000 \
  -addr 10.0.0.0 255.255.255.0 192.168.1.254 255.255.255.255
```

On the home firewall (firewall B), the IPSEC routes will be exactly opposite. Make sure that you change the SPI on the remote firewall to the second SPI created by the SA. Packets will be sent / received on other SPI. Firewall A will send packets to B on SPI 1000. Firewall B will send packets to A on SPI 1001:

```
# Set up the IPSEC routes on Firewall B
/sbin/ipsecadm flow -dst 192.168.2.1 -spi 1001 \
```

```
  -addr 192.168.1.254 255.255.255.255 192.168.2.1 255.255.255.255 -local

/sbin/ipsecadm flow -dst 192.168.2.1 -spi 1001 \
  -addr 10.0.3.0 255.255.255.0 10.0.0.0 255.255.255.0

/sbin/ipsecadm flow -dst 192.168.2.1 -spi 1001 \
  -addr 192.168.1.254 255.255.255.255 10.0.3.0 255.255.255.0 -local

/sbin/ipsecadm flow -dst 192.168.2.1 -spi 1001 \
  -addr 10.0.3.0 255.255.255.0 192.168.2.1 255.255.255.255
```

Note that the '-local' flag is only used for routes referencing the local external IP. The '-addr' argument specifies the source and destination addresses that will match this route. The '-dst' argument (for flows) is always the outside IP of the remote firewall.

The OpenBSD distribution supplies a sample script to automate the ipsecadm commands. It can be found in /usr/share/ipsec/rc.vpn (as of release 2.4).

On firewall A, the relevant output from *netstat -rn* will look something like this.

2.6 - How do I set up my IPF rules?

The following ruleset is configured on the office firewall (firewall A). Firewall B will be similar.

```
# xl0 is external interface
# xl1 is internal interface

# Default Deny and Log Everything
block in log all
block out log all

# Passing in encrypted traffic from security gateways
pass in on xl0 proto sipp-esp from 192.168.1.254 to 192.168.2.1
pass out on xl0 proto sipp-esp from 192.168.2.1 to 192.168.1.254

# Allow packets to pass from the internal (local) side of the VPN
# to the internal (remote) side of the VPN. This traffic will get
# encapsulated within the VPN tunnel on enc0 before going
# out the physical interface.

pass in quick on xl1 from 10.0.0.0/24 to 10.0.3.0/24
pass out quick on xl1 from 10.0.3.0/24 to 10.0.0.0/24

# If packets are on the encrypted interface, enc0, they have been
# authenticated / decrypted. Pass them.

pass in quick on enc0
pass out quick on enc0
```

2.7 - Is there a quick way to flush IPSec routes?

To delete all current IPSec routes, do a:

```
# route flush -encap
```

3.0 - Changelog

1.07 - [99-03-02] Reorganization of content. Added New References & Q2.7
1.06 - [99-02-04] Fixed typos and incorrect links (thanks to Jean-Charles Grégoire)
1.05 - [99-01-28] Modified Q5,6. Inserted new Q6 and reordered.
1.04 - [99-01-28] Added Changelog (Irony. Love it)
1.03 - [99-01-27] Incorporated Matt's HOWTO information

1.02 - [99-01-26] Added Q1,2,3,4,10
1.01 - [99-01-20] Initial Release

--
http://www.codetalker.com/greenbox/docs/vpn-24-minifaq.html
Please send any comments, questions, or suggestions to smcquade@codetalker.com