IPSec and Hardware Acceleration#

We have full support for IPSec built into the ip-v4-hc and ip-v6-hc stack variants (don't forget to add the "ipsec" parameter option to the stack to enable ipsec when the stack starts).

There's a good reference page in the NetBSD man pages covering ipsec in general. There are some aspects that don't apply (you obviously don't have to worry about re-building the kernel :>) but the general usage is the same.

Examples#

The following examples illustrate basic ipsec setup

Setup an ipsec connection between two boxes manually#

Consider there are two boxes A and B, and want to establish ipsec between them. BoxA ==(ipsec) == BoxB

Step 1: On each box, create a script file (lets say it name is "script") having the following content:

```
#!/bin/ksh
# args: This script takes two arguments. The first one is the IP address of the box that is to run it on.
#         The second one is the IP address of the box that this box is to establish ipsec connection to.
Myself=$1
Remote=$2

# the following two lines are to clean the database. I put these here simply to demo the "hello world" level
# connection. You certainly don't need to do that ...
#
setkey -FP
setkey -F

# use setkey to input all of the SA content.
setkey -c << EOF

spdadd $Myself $Remote any -P out ipsec esp/transport/$Myself-$Remote/require;
spdadd $Remote $Myself any -P in ipsec esp/transport/$Remote-$Myself/require;

add  $Myself $Remote esp 1234 -m any -E 3des-cbc "KeyIsTwentyFourBytesLong";
add  $Remote $Myself esp 1234 -m any -E 3des-cbc "KeyIsTwentyFourBytesLong";
EOF
```

Step 2: On BoxA, run "./script `BoxA BoxB". Or directly give the IP address of each box if the name can not be resolved (i.e. "BoxA" =>"192.168.200.10"). Similarly, on BoxB, run "./script BoxB BoxA".

That's all there is. Now you can check the connection by pinging each other. You can get the ipsec status by using "setkey -PD".

Setup an ipsec connection by authentication using the pre-shared-key method#

Consider the simplest case where there are two boxes BoxA and BoxB. User A is on BoxA and User B is on ~Box B and the two users have a shared secret which is a string of "hello_world".

Step 1: On Box A, create a file psk.txt which has the related two lines in there,

```
usera@qnx.com   "Hello_world"
userb@qnx.com   "Hello_world"
```

This file will be used by ***racoon***, the IPSec IKE, to do the authentication and IPSec connection job. The psk.txt file must be owned by root and read/write only by root. To make sure this is the case, run "chmod 0600 psk.txt".

racoon needs a configuration file (e.g. racoon.conf). This file defines the way that *racoon* is to operate. In the remote session, specify that we are going to use pre-shared key method as authentication and let *racoon* know where to find the secret. For example:

```
......
path pre_shared_key "your_full_path_to_psk.txt" ; # let racoon know where your pre shared keys are
remote anonymous
{
    exchange_mode aggressive,main;
    doi ipsec_doi;
    situation identity_only;

    #my_identifier address;
    my_identifier user_fqdn "usera@qnx.com";
    peers_identifier user_fqdn "userb@qnx.com";

    nonce_size 16;
    lifetime time 1 hour;   # sec,min,hour
    initial_contact on;
    proposal_check obey;    # obey, strict or claim

    proposal {
        encryption_algorithm 3des;
        hash_algorithm sha1;
        authentication_method pre_shared_key ;
        dh_group 2 ;
    }
}
......
```

Step 2: Setup the policy using ***setkey***. You may use the following script to tell the stack that the IPSec between BoxA and BoxB requires key negotiation.

```
#!/bin/sh
# This is a simple configuration for testing racoon negociation.
#

Myself=$1
Remote=$2

setkey -FP
setkey -F
setkey -c << EOF
#
spdadd $Remote $Myself any -P in  ipsec esp/transport/$Remote-$Myself/require;
spdadd $Myself $Remote any -P out ipsec esp/transport/$Myself-$Remote/require;
#
EOF
```

Run on BoxA as "./script BoxA BoxB".

Step 3: Do Step 1 and Step 2 on BoxB. Needless to say, on BoxB you need to run as "./script BoxB BoxA" (and so on)

Step 4: On Both boxes, run "racoon -c *full_path_to_*racoon.conf". When you initiate traffic, say by trying to ping the peer box, *racoon* will do its job and establish the IPSec connection by creating SAs for both direction and then you can see the traffic passing back and forth which says the IPSec connection is established.

IPSec Tools#

We've taken IPSec tools (ipsec-tools.sourceforge.net) from the NetBSD source base and incorporate it into our source base. The build produces:

- libipsec (PF_KEY library routines)
- setkey (Security Policy Database and Security Association Database management tool)
- racoon (IKE Daemon)
- racoonctl (racoon commandl liine control tool)

OpenSSL Support#

The OpenSSL crypto and SSL libraries (from www.openssl.org) has been ported and the source and library are available for your applications to use.

Hardware Accelerated Crypto#

io-pkt has the (hardware-independent) infrastructure to load a (hardware-dependent) driver to take advantage of dedicated hardware which can perform cryptographic operations at high speed. This not only speeds up the crypto ops (such as used by IPSec) but also has the effect of reducing the CPU load. This interface is carefully crafted so that the stack does not block on the crypto op - rather, it continues on, and later on, using a callback, the driver returns the processed data to the stack. This is ideal for dma-driven crypto hardware.

Supported Hardware Crypto Engines#

One example of this is the MPCSEC crypto hardware core (present on some, but not all MPC 85XX and 83XX) which io-pkt had a driver for. This driver is called "devnp-mpcsec.so" and is found in the "sys/dev_qnx/mpcsec" source directory. It loads just like a network driver - to run it:

```
# io-pkt-v6-hc -d mpcsec -d mpc85xx
```

where devnp-mpc85xx.so is the driver for the tsec or etsec ethernet ports on the MPC 83xx and 85xx. For considerably more information on this subject, please see the following text file: "sys/dev_qnx/mpcsec/README" which goes into great detail about the layout and functioning of this driver, and some sample IPSec configurations.