Running OpenHIP under CORE

This is how to build the publicly-available open source version of OpenHIP, which differs from TNW OpenHIP.

This assumes a Ubuntu 16.04 install, see CORE Install Ubuntu 16.04.

External references: http://openhip.sourceforge.net/wiki/index.php/Configuration and http://openhip.sourceforge.net/wiki/index.php/Endbox

1.0 Prerequisites

OpenHIP depends on these libraries:

```
sudo apt-get install libxml2-dev libssl-dev libtool subversion
```

Subversion (optional) was added above for checking out the latest source from SourceForge.

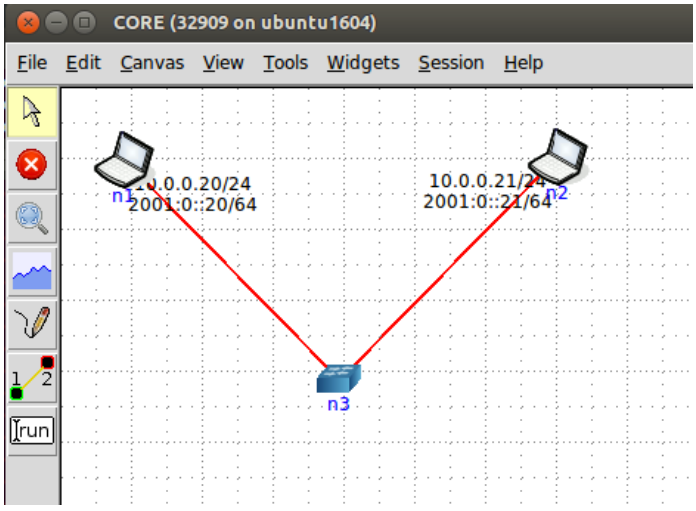
The following "modes" of HIP are decided at build-time, not run-time.

A basic understanding of the following may be helpful:

- /usr/local/etc/hip/hip.conf
  - HIP configuration options in XML format
  - can be same across all nodes, if interface names are the same
- /usr/local/etc/hip/my_host_identities.xml
  - contains the unique public/private key pair used as the HIP identity
  - unique per node
- /usr/local/etc/hip/known_host_identities.xml
  - contains peer HIPswitch public keys, HITs, and IP addresses
  - can be shared (same) across all nodes
  - for VPLS mode, also contains legacy node IP addresses, and "allowed peer" HIT pairs

2.0 Classic HIP - Transport Mode

ESP Transport mode is used to protect local application traffic. Apps use the LSI to communicate securely with a peer.



2.1 Build HIP in its default transport mode

```
svn checkout svn://svn.code.sf.net/p/openhip/code/hip/trunk openhip
cd openhip
# ignore warnings here
./bootstrap.sh
./configure
make -j4
# this installs /usr/local/sbin/hip,hitgen,hipstatus
sudo make install
```

2.2 Start a new CORE Session and build HIP configs

Using the CORE GUI, draw a simple network: 2 PCs connected to a switch.
Click start.

***CAUTION**: when you stop the CORE scenario, the /tmp/pycore.nnnn session dir is removed. Please copy all your configuration changes below before pressing the Stop button.*

Double-click n1, and set up node 1's identity using these commands:

```
mkdir $SESSION_DIR/n1.conf/usr.local.etc.hip
mkdir -p /usr/local/etc/hip
mount --bind $SESSION_DIR/n1.conf/usr.local.etc.hip /usr/local/etc/hip
cd /usr/local/etc/hip
hitgen -conf
# enter random text to seed the random number generator
hitgen
hitgen -publish
```

Double-click n2, and set up node 2's identity using these commands:

```
mkdir $SESSION_DIR/n2.conf/usr.local.usr.local.etc.hip
mount --bind $SESSION_DIR/n2.conf/usr.local.etc.hip /usr/local/etc/hip
cd /usr/local/etc/hip
hitgen -conf
# enter random text to seed the random number generator
hitgen
hitgen -publish
```

Build a "directory of peers" – a ***known_host_identities.xml*** file containing both peers (change your session number as needed; example here is *32909*).

```
# from the host machine
cat /tmp/pycore.32909/n1.conf/usr.local.etc.hip/n1_host_identities.pub.xml >
/tmp/pycore.32909/known_host_identities.xml
cat /tmp/pycore.32909/n2.conf/usr.local.etc.hip/n2_host_identities.pub.xml >>
/tmp/pycore.32909/known_host_identities.xml
```

Manually edit /tmp/pycore.32909/known_host_identities.xml

- remove the middle *known_host_identities* tags and *xml* tag to make a valid XML file
- add `<addr>10.0.0.20</addr>` under the n1 host identity
- add `<addr>10.0.0.21</addr>` under the n2 host identity

Deploy the new file:

```
sudo cp /tmp/pycore.32909/known_host_identities.xml /tmp/pycore.32909/n1.conf/usr.local.etc.hip/
sudo cp /tmp/pycore.32909/known_host_identities.xml /tmp/pycore.32909/n2.conf/usr.local.etc.hip/
```

2.3 Run HIP and perform Base Exchange (BEX)

Run HIP on node 1; in a n1 terminal, type:

```
hip -v
```

Run HIP on node 2; in a n2 terminal, type:

```
hip -v
```

Start a ping from node 1 to 2 using HIP. Open a new n1 terminal and find the LSI of the peer using:

```
grep LSI /usr/local/etc/hip/known_host_identities.xml
```

it is the 2nd entry; ping the peer LSI to establish a HIP association.

```
ping 1.140.171.41
```

Tip: use '`brctl show`' on the host; you can run Wireshark on the bridge to inspect the HIP exchange, encrypted, and other protocol traffic.

2.4 Test HIP IP address mobility

You can test mobility - from n1 first do:

```
# remove IPv6 address to prevent software bug
ip addr del 2001::20/64 dev eth0
# keep ping running as before
ping 1.140.171.41
```

From n2 terminal, use:

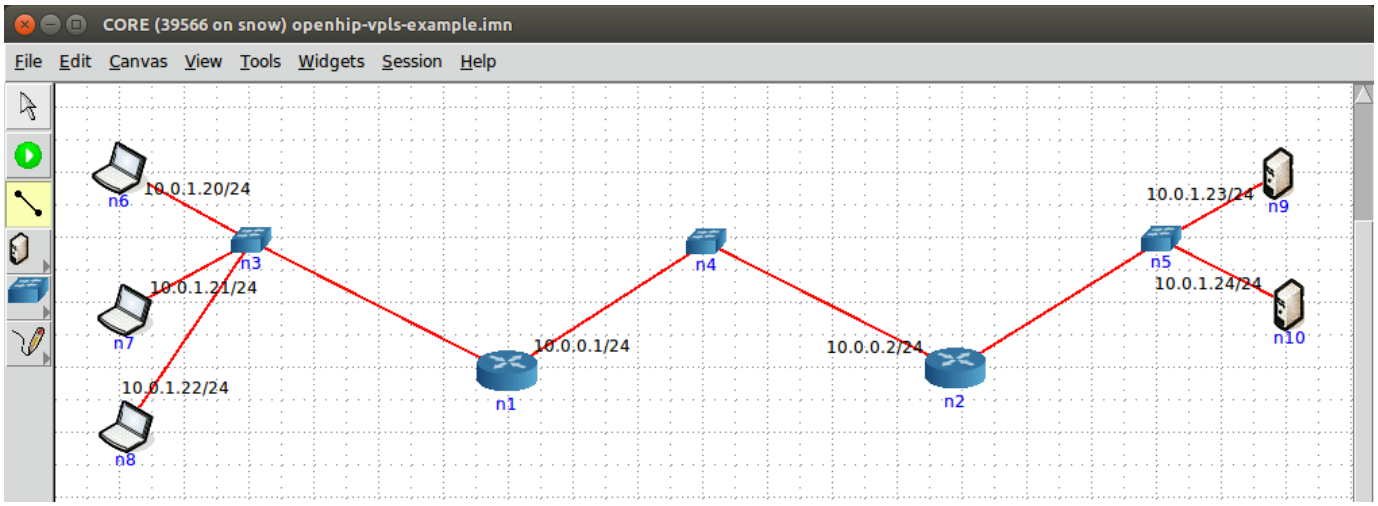
```
ip addr del 2001::21/64 dev eth0
ifconfig eth0 10.0.0.30/24
```

You can repeat the mobility demo using SSH by enabling the SSH service in CORE.

3.0 HIP VPLS - Tunnel Mode - Files

ESP Tunnel mode is used to protect traffic of connected devices. This is almost the same as TNW's usage of HIP, without the Conductor orchestration of policy, and various other networking features that we offer.

This uses static XML files to configure peer policy.



3.1 Build HIP in Virtual Private LAN Service (VPLS) Mode

```
svn checkout svn://svn.code.sf.net/p/openhip/code/hip/trunk openhip
cd openhip
# ignore warnings here
./bootstrap.sh --enable-vpls
./configure --enable-vpls=files
# fix bugs
sed -i '182s|return(-1)|//return(-1)|' src/util/cfg-fileship_cfg_files.cpp
sed -i '188s|return(-1)|//return(-1)|' src/util/cfg-fileship_cfg_files.cpp
make
# this installs /usr/local/sbin/hip,hitgen,hipstatus
# and /usr/local/lib/libhipcfg*
sudo make install
```

```
# manually install these
mkdir -p /usr/local/etc/hip
sudo cp src/util/scripts/get_mylsi.sh /usr/local/etc/hip/
sudo cp src/util/scripts/bridge*.sh /usr/local/etc/hip/
```

3.2 Start a new CORE Session and build HIP configs

Build a CORE VPLS scenario. With VPLS, "legacy node" traffic is protected using IPsec tunnel mode.

- Draw two routers and three switches.
- Connect each router to one of the switches (the addresses 10.0.0.1 and 10.0.0.2 are assigned.)
    - The router icon will become the HIP node.
    - This first link will become the *eth0* interface on each router. This will be important in the HIP config later.
- Connect a second interface on n1 to another switch, and n2 to the remaining switch.
    - This second link will become the routers' *eth1* interface.
- Draw several PC nodes for acting as legacy nodes. Link them to the second switches (link to routers' *eth1*). See diagram.
- Fix up the IP addressing and routing.
    - Choose *Tools* > *IP Addresses* and under *Remove IPv6:* click the *All* button. This removes all IPv6 addresses. Click *OK*.
    - Remove the *eth1* IP address from router nodes n1 and n2, e.g. *10.0.1.1* and *10.0.2.1*. HIP will act as a Layer-2 forwarder and does not require an address on these interfaces.
    - Readdress all legacies so they are in the same subnet. For example, double-click each legacy in 10.0.2.x/24 and change the address to be 10.0.1.x/24.
- Right-click each router and choose *Services*. Disable everything but *IPForward*. Click *Apply*.

Start the session to begin HIP configuration.

3.3 Build HIP VPLS configs

Double-click n1, and set up node 1's identity using these commands:

```
mkdir $SESSION_DIR/n1.conf/usr.local.etc.hip
mkdir -p /usr/local/etc/hip
mount --bind $SESSION_DIR/n1.conf/usr.local.etc.hip /usr/local/etc/hip
cd /usr/local/etc/hip
hitgen -conf
# enter random text to seed the random number generator
hitgen
hitgen -publish
```

Double-click n2, and set up node 2's identity using these commands:

```
mkdir $SESSION_DIR/n2.conf/usr.local.etc.hip
mount --bind $SESSION_DIR/n2.conf/usr.local.etc.hip /usr/local/etc/hip
cd /usr/local/etc/hip
# enter random text to seed the random number generator
hitgen
hitgen -publish
```

Modify the default generated hip.conf of n1, and use the same file on n2:

- edit **/tmp/pycore.32909/n1.conf/usr.local.etc.hip/hip.conf**
- change this tag: `<disable_udp>yes</disable_udp>`
- add these tags:
  `<master_interface>eth0</master_interface>`
  `<use_local_known_identities>yes</use_local_known_identities>`
  `<cfg_library>/usr/local/lib/libhipcfgfiles.so</cfg_library>`
- on the host, copy the hip.conf from n1 to n2:
  sudo cp /tmp/pycore.32909/n1.conf/usr.local.etc.hip/hip.conf /tmp/pycore.32909/n2.conf/usr.local.etc.hip/hip.conf

Build a "directory of peers" – a ***known_host_identities.xml*** file containing both peers (change your session number as needed; example here is*3 2909*).

```
# from the host machine
cat /tmp/pycore.32909/n1.conf/usr.local.etc.hip/n1_host_identities.pub.xml >
/tmp/pycore.32909/known_host_identities.xml
cat /tmp/pycore.32909/n2.conf/usr.local.etc.hip/n2_host_identities.pub.xml >>
/tmp/pycore.32909/known_host_identities.xml
```

Manually edit /tmp/pycore.32909/known_host_identities.xml

- remove the middle *known_host_identities* tags and *xml* tag to make a valid XML file
- add `<addr>10.0.0.1</addr>` under the n1 host identity
- add `<addr>10.0.0.2</addr>` under the n2 host identity
- add the legacy node IP addresses to each identity.
    - for n1, add 10.0.1.20, 10.0.1.21, 10.0.1.22
    - for n2, add 10.0.1.23, 10.0.1.24
    - add a new tag within each host_identity block for the above IPs, using this format:
      `<legacyNodesIp>10.0.1.20</legacyNodesIp>`
    - copy and paste the two HITs into a new <peer_allowed> section following the <host_identity> tags:
      (***NOTE***: update the HITs below to match your n1, n2 HITs.)

      ```
      <peer_allowed>
        <hit1>2001:17:c44d:4ced:167f:3245:a294:ebd1</hit1>
        <hit2>2001:11:722b:4d58:549c:6ca7:34fb:f4d4</hit2>
      </peer_allowed>
      ```

Deploy the new file:

```
sudo cp /tmp/pycore.32909/known_host_identities.xml /tmp/pycore.32909/n1.conf/usr.local.etc.hip/
sudo cp /tmp/pycore.32909/known_host_identities.xml /tmp/pycore.32909/n2.conf/usr.local.etc.hip/
```

Also deploy the scripts from the host's /usr/local/etc/hip/ dir:

```
sudo cp /usr/local/etc/hip/*.sh /tmp/pycore.32909/n[1,2].conf/usr.local.etc.hip/
```

Example fully-worked **known_host_identities.xml** file for VPLS mode:

```
<?xml version="1.0" encoding="UTF-8"?>
<known_host_identities>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
    <name>n1-1024</name>
    <HIT>2001:17:c44d:4ced:167f:3245:a294:ebd1</HIT>
    <LSI>1.148.235.209</LSI>
    <addr>10.0.0.1</addr>
    <legacyNodesIp>10.0.1.20</legacyNodesIp>
    <legacyNodesIp>10.0.1.21</legacyNodesIp>
    <legacyNodesIp>10.0.1.22</legacyNodesIp>
  </host_identity>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
    <name>n2-1024</name>
    <HIT>2001:11:722b:4d58:549c:6ca7:34fb:f4d4</HIT>
    <LSI>1.251.244.212</LSI>
    <addr>10.0.0.2</addr>
    <legacyNodesIp>10.0.1.23</legacyNodesIp>
    <legacyNodesIp>10.0.1.24</legacyNodesIp>
  </host_identity>
  <peer_allowed>
    <hit1>2001:17:c44d:4ced:167f:3245:a294:ebd1</hit1>
    <hit2>2001:11:722b:4d58:549c:6ca7:34fb:f4d4</hit2>
  </peer_allowed>
</known_host_identities>
```

Example fully-worked **hip.conf** file:

```
<?xml version="1.0" encoding="UTF-8"?>
<hip_configuration>
<master_interface>eth0</master_interface>
<cfg_library>/usr/local/lib/libhipcfgfiles.so</cfg_library>
<use_local_known_identities>yes</use_local_known_identities>
<cookie_difficulty>10</cookie_difficulty>
<packet_timeout>10</packet_timeout>
<max_retries>5</max_retries>
<sa_lifetime>900</sa_lifetime>
<send_hi_name>yes</send_hi_name>
<dh_group>3</dh_group>
<dh_lifetime>900</dh_lifetime>
<r1_lifetime>300</r1_lifetime>
<failure_timeout>50</failure_timeout>
<msl>5</msl>
<ual>600</ual>
<min_reg_lifetime>96</min_reg_lifetime>
```

```
<max_reg_lifetime>255</max_reg_lifetime>
<hip_sa>
<transforms>
<id>1</id>
<id>2</id>
<id>3</id>
<id>4</id>
<id>5</id>
<id>6</id>
</transforms>
</hip_sa>
<esp_sa>
<transforms>
<id>1</id>
<id>2</id>
<id>3</id>
<id>4</id>
<id>5</id>
<id>6</id>
</transforms>
</esp_sa>
<disable_dns_lookups>no</disable_dns_lookups>
<save_known_identities>no</save_known_identities>
<disable_notify>no</disable_notify>
<disable_dns_thread>yes</disable_dns_thread>
<enable_broadcast>no</enable_broadcast>
<disable_udp>yes</disable_udp>
</hip_configuration>
```

3.4 Run HIP and perform Base Exchange (BEX)

From node n6 (legacy node in the upper-left):

```
ping 10.0.1.23
```

Within each HIP node, you can run '`tcpdump -nli hipbr`' to see legacy device traffic.

To see the BEX / encrypted traffic, right-click on a HIP node and choose *tcpdump* for *eth0*.

4.0 CORE scenario files

These files contain all of the required HIP configuration. You just need to build/install HIP according to the instructions above.

- openhip-bex-example.imn
- openhip-vpls-example.imn