SCION Tutorial

First steps with SCION

Contact information

If you experience problems or have doubts during the course of the SIGCOMM2020 SCION tutorial, please contact us on the tutorial's Slack channel. After the tutorial is over, you can contact us by email.

Starting the VM

Open a terminal in your host OS and change the working directory to where the `Vagrantfile` from SCIONLab is located. You have to start up the VM, and for that you can execute:

```
vagrant up
```

Some messages will be displayed, indicating the progress of the VM creation. The command should exit with zero (you can check with `echo $?` right after `vagrant up` has finished).

Every time you want to log in to your VM you can go to the directory where the `Vagrantfile` is, and execute:

```
vagrant ssh
```

That will open a shell terminal inside your VM.

At any point you can check if the services are running with `sudo systemctl list-dependencies scionlab.target`. It should list at least 4 services: border router, control service, daemon, and dispatcher must be running.

Now that you have a terminal open in your VM, check connectivity to the infrastructure. We can perform an SCMP echo, which is similar to an ICMP echo but with SCION, by running:

```
scmp echo -remote 18-ffaa:0:1201,128.237.152.180
```

This sends SCMP echo packets to a computer with IP `128.237.152.180` in AS `18-ffaa:0:1201`. This computer is part of the SCIONLab infrastructure, and should reply to those echo packets.

The command should output the reply packets arriving shortly after, with some milliseconds latency. Of course, you can send SCMP echo packets to other computers in other ASes. Try with those of your colleagues.

Explore the SCION system and use SCION apps

USING THE WEB APPLICATION

The easiest way to explore the SCION system and the topology is by using the SCIONLab User AS web application. We are going to install it by logging into your VM (`vagrant ssh` as mentioned before), and then running:

```
sudo apt-get install -y scion-apps-webapp
sudo systemctl start scion-webapp.service
```

With that, the web application should be running inside the VM, listening for connections on port `8000`. Because the VM is configured to forward that port, you can already connect from your host machine at [http://localhost:8000]. In this page, the health status of your user AS is displayed. All the sections in the health check should be passing successfully, and displayed in green.

Go to the page *Apps* and select one of the apps (*bwtester*, *camerapp*, *sensorapp*):

- `bwtester` (link) Tests bandwidth for a given path. If you select this option, please don't overwhelm our routers by using much bandwidth, or running it for longer than some seconds.
- `camerapp` (link) Downloads image files from the server. These images can be periodically grabbed from a device, e.g. a camera.
- `sensorapp` (link) Downloads sensor information.

You should not need to change the source settings but you can select any of the ASes preconfigured in the list as destination or could enter the information of any of your colleagues' ASes.

You can either directly execute the different apps on the *Execute* tab or explore possible paths to your selected destination on the *Paths* tab. Note that you can explore paths to your colleagues' ASes even if they have not yet set up the selected services. You can have an overview of the SCIONLab AS infrastructure here.

The web application also allows you to look at the TRC of your ISD as well as your certificate and the SCION services running in your AS. Just copy the contents of the *payload* which are base64 encoded, and decode them online or with the command `base64 -d`. You will see a json object with fields like `"primary_ases"`, etc.

FROM A SHELL INSIDE YOUR VM

Inside your VM, you can explore several subdirectories:

- The directory `/var/log/scion/` allows you to access logs of the different SCION services. In particular the control service's log `cs*-1.log` can be interesting. Look for thing like path segments requests and responses, or beacon registrations (grep for `segReq`, `segment` or `beacon`).
- The directory `/etc/scion/gen/` contains configuration files for you AS such as the local topology.

- The topology files are a set of `json` files located under `/etc/scion/gen/` that you can find with `find /etc/scion/gen -name topology.json`. They all are identical in your AS. If you open one, you will see something like:


```
{
  "Attributes": [],
  "BorderRouters": {
    "br17-ffaa_1_13-1": {
      "CtrlAddr": {
        "IPv4": {
          "Public": {
            "Addr": "127.0.0.1",
            "L4Port": 30045
          }
        }
      },
      "Interfaces": {
        "1": {
          "Bandwidth": 1000,
          "ISD_AS": "17-ffaa:0:1107",
          "LinkTo": "PARENT",
          "MTU": 1472,
          "Overlay": "UDP/IPv4",
          "PublicOverlay": {
            "Addr": "10.1.0.129",
            "OverlayPort": 50000
          },
          "RemoteOverlay": {
            "Addr": "10.1.0.1",
            "OverlayPort": 50039
          }
        }
      },
      "InternalAddrs": {
```

```
        "IPv4": {
          "PublicOverlay": {
            "Addr": "127.0.0.1",
            "OverlayPort": 31045
          }
        }
      }
    },
  },
  "ControlService": {
    "cs17-ffaa_1_13-1": {
      "Addrs": {
        "IPv4": {
          "Public": {
            "Addr": "127.0.0.1",
            "L4Port": 30254
          }
        }
      }
    },
  },
  "ISD_AS": "17-ffaa:1:13",
  "MTU": 1472,
  "Overlay": "UDP/IPv4"
}
```

This topology defines two services: one control service and one border router. The control service has the address where inside the AS it is listening for requests (such as path retrieval). The border router defines, among other things, the interfaces to and from the outside of the AS. The interface in the example is a connection to a provider, using an IPv4 overlay, listening at `10.1.0.129:50000` and connecting to `10.1.0.1:50039`. We will later on the tutorial work with the topology file when we add a new interface.

Aside from the directories, you can also run from a terminal several SCION related commands, like `showpaths` to show the available paths to a given AS, for example from your user AS to `18-ffaa:0:1201`:

```
showpaths -dstIA 18-ffaa:0:1201
```

Run as well `scmp echo` like mentioned before.

You can also run the different SCION apps directly from the shell. For that call either `scion-bwtestclient`, `scion-imagefetcher`, or `scion-sensorfetcher` with the option `-s ISD-AS,[Addr]:Port`. The bandwidth tester allows you to explore potential paths by using the *interactive* option `-i`.

There are already several servers just for testing:

- sensorserver at `17-ffaa:0:1102,[192.33.93.177]:42003`
- cameraserver at `17-ffaa:0:1102,[192.33.93.166]:42002`
- bwtestserver at list of bwtestservers

Be sure to visit the general tutorial about some existing SCION applications located here

Run your own SCION apps

Once you have setup one or more of following servers, you can post your IA and port to connect to on the Slack channel(https://sigcomm.slack.com/archives/C0186E0GY0G), so other people can try to connect to it and test it.

BANDWIDTH TESTER

Running the bandwidth test server is straight forward. On the SCION tutorials page go to its help page and follow the instructions there.

IMAGESERVER AND SENSORSERVER

You can also set up the apps *imageserver* and *sensorserver* following the instructions on the tutorials website. As you do not have sensors or a camera connected to your VM, you need to use dummy scripts that print some information that you can pipe into *sensorserver* or periodically create jpeg images for *imageserver*. Remember that you can transfer files between your host and your VM using the host's directory where the `Vagrantfile` is located, and that is mirrored inside the VM on `/vagrant`.

Please set up the `imageserver` on the default port (40002). We have a script constantly trying to fetch images from all ASes. You can fetch the combined image using `imagefetcher` from `17-ffaa:1:d9b,127.0.0.1:40002`.

Example of script copying an image:

```
#!/bin/bash

while true
do
  cp images/success.jpg `date +"%Y-%m-%d_%k:%M"`_success.jpg
  sleep 60
done
```

Example of script printing system info:

```
#!/bin/bash

while true
do
  echo "hostname: `hostname`, user: `whoami`, current time: `date`"
  sleep 1
done
```

Access a web server

WITH SCION

There is a simple web server that runs in a computer under the following address:

```
18-ffaa:0:1201,128.237.152.165:4443/
```

You can access the webpages manually with the `bat` application. Open a terminal inside your VM with `vagrant ssh` and follow the next steps:

```
sudo apt-get install scion-apps-bat
scion-bat 18-ffaa:0:1201,128.237.152.165:4443/
```

The last command runs `scion-bat`, which is a `curl` like application that works with the SCION protocol. More information here.

IP REGULAR HTTP SERVER

There is an IP regular http server running on a non publicly routeable address:

```
http://10.42.0.1:4443/
```

This server can be accessed using the SIG, as explained in the SIG task. It's listed here for completeness.

Set up an additional peering link with your second user AS

CONFIGURE A SECOND USER AS

Configure another user AS in SCIONLab. Use a different attachment point or even ISD than the one you have for your first AS. Download the tar file and extract the `Vagrantfile` like you did with your first AS. Now, since we are going to run this VM at the same time as we also run the first one, you have to edit the `Vagrantfile` and comment out with a `#` the following line:

```
config.vm.network "forwarded_port", guest: 8000, host: 8000, protocol: "tcp"
```

Add a new line right before `config.vm.provider "virtualbox" do |vb|` in the `Vagrantfile`. It will configure a new interface in the VM with that IP:

```
config.vm.network :private_network, ip: "10.0.0.20"
```

You might also want to edit the settings about the memory reserved to the VM with the line `vb.memory = "2048"`. Setting it to `1024` should also work okay, if your host machine starts to run out of memory.

After saving the file, proceed like with the first VM:

- `vagrant up`
- `vagrant ssh`
- Check connectivity to the infrastructure with scmp echo.
- Output services running with `sudo systemctl list-dependencies scionlab.target`

RECONFIGURE YOUR FIRST USER AS

Now go back to the directory where you have the `Vagrantfile` for your first VM and stop it by running:

```
vagrant halt
```

Edit the `Vagrantfile` to add a new interface, like you did with your second AS, but with a different IP. So, right before the line `config.vm.provider "virtualbox" do |vb|` add the following:

```
config.vm.network :private_network, ip: "10.0.0.10"
```

Save the file and run the VM again with `vagrant up`. Once it has booted, check again connectivity with scmp echo. If everything is still fine, you can ping the other machine with a normal IP ping command:

```
ping 10.0.0.20
```

This will send ICMP packets to the second VM, and it should reply normally.

Right now, the two VMs have a private network where they can communicate directly, using `10.0.0.10` and `10.0.0.20` as their IPs.

EDIT THE TOPOLOGY FILES

Go to your first VM with `vagrant ssh`. Inside `/etc/scion/gen` there will be an `ISD*` directory, with an `AS*` directory. Change directory to the `sciond` service by running:

```
cd /etc/scion/gen/ISD*/AS*/endhost
```

Now when you list the files, there will be one called `topology.json`. Open it with your favorite editor, e.g. `vim`:

```
sudo cp topology.json topology.json.bak  # create a backup copy of the file, convenient
sudo vim topology.json
```

- In the `Interfaces` section, copy-paste one of the interfaces
- Adjust the new interface appropriately:
  - Select a new unused interface ID e.g. `2`.
  - Set the `LinkTo` to `PEER`
  - Set the `PublicOverlay` to `10.0.0.10`, port `50001`
  - Set the `RemoteOverlay` to `10.0.0.20`, port `50001`
  - Set `ISD_AS` to your second user AS ISD and AS ID.
- Save the modified topology file
- Copy the modified topology file into all subdirectories of `/etc/scion/gen/ISD*/AS*/` (all other services need to know the new topology) `sudo cp topology.json ../br*/ ; sudo cp topology.json ../cs*/`.
- Restart SCION by calling `sudo systemctl restart scionlab.target`

The added section to the topology file should look like this (where `<ISD2>-ffaa:1:YYYY` is the IA of your second user AS); **remember to replace** `<ISD2>-ffaa:1:YYYY` with the appropriate ISD-AS ID of your second user AS):

```
"2": {
  "Bandwidth": 1000,
  "ISD_AS": "<ISD2>-ffaa:1:YYYY",
  "LinkTo": "PEER",
  "MTU": 1472,
  "Overlay": "UDP/IPv4",
  "PublicOverlay": {
    "Addr": "10.0.0.10",
    "OverlayPort": 50001
  },
  "RemoteOverlay": {
```

```
      "Addr": "10.0.0.20",
      "OverlayPort": 50001
    }
  }
```

As always, restart services with `sudo systemctl restart scionlab.target` and check connectivity to the infrastructure with scmp echo.

When it's clear that the first AS works fine, adapt the topology files of the second AS. Remember to write the correct IA of the first AS, and to use the correct IPs for the public and remote overlays. The section should look like this (where `<ISD1>-ffaa:1:XXXX` is the IA of your first AS; **remember to replace** `<ISD1>-ffaa:1:XXXX` with the appropriate ISD-AS ID of your first user AS):

```
"2": {
  "Bandwidth": 1000,
  "ISD_AS": "<ISD1>-ffaa:1:XXXX",
  "LinkTo": "PEER",
  "MTU": 1472,
  "Overlay": "UDP/IPv4",
  "PublicOverlay": {
    "Addr": "10.0.0.20",
    "OverlayPort": 50001
  },
  "RemoteOverlay": {
    "Addr": "10.0.0.10",
    "OverlayPort": 50001
  }
}
```

Like before, copy the `topology.json` file to all subdirectories of `/etc/scion/gen/ISD*/AS*/`. Then restart services with `sudo systemctl restart scionlab.target`.

CHECK PEERING LINK

After waiting some seconds for the beacons to be propagated, you should now be able to use this new path in any application. Let's first find out if the peering path is visible by running:

```
showpaths -dstIA <ISD2>-ffaa:1:YYYY
```

This will print many lines with some of the possible paths from the two ASes. As an example, one of the first lines that you should see would look like:

```
[ 0] Hops: [19-ffaa:1:XXXX 2>2 17-ffaa:1:YYYY] MTU: 1472, NextHop: 127.0.0.1:31045
```

When run from `19-ffaa:1:XXXX` looking for `17-ffaa:1:YYYY`.

Many of the applications accept a `-i` switch (interactive) that allows you to select the path manually. Use it with `scmp echo` and observe that the latency is much lower than going upstream and down again.

NETCAT

At any point you can install the scion netcat application with:

```
sudo apt-get install scion-apps-netcat
```

The application supports a subset of the regular BSD netcat command. Run netcat in listening mode in a machine of one AS like:

```
scion-netcat -u -l 40002
```

And netcat connecting with:

```
scion-netcat -u <ISD1>-ffaa:1:XXXX,[127.0.0.1]:40002
```

Where `<ISD1>-ffaa:1:XXXX` is the IA of your other user AS. You can type in any of the running processes, and it will echo it to the other side.

Set up a SCION-IP gateway (SIG)

The SCION-IP gateway (SIG) is a mechanism that tunnels IP packets through the SCION network. It can be configured to send packets to specified destination prefixes to another SIG in a specified SCION AS, where they are decapsulated and forwarded. We have set up a SIG in AS `18-ffaa:0:1201` and a standard webserver listening at `10.42.0.1:4443` (a private IP address from RFC 1918 address space that is not globally routable). The goal of this exercise is for you to set up a SIG that encapsulates IP packets sent to the prefix `10.42.0.0/24` in SCION packets and sends them to the SIG in AS `18-ffaa:0:1201`, such that you can query the webserver with the standard (non-SCION-aware) `curl` tool.

The following steps are a slightly modified version of the general SIG tutorial.

INSTALL AND CONFIGURE SIG

To install `sig`, run:

```
sudo apt install scion-sig
```

See Installation for details.

The `scion-sig` package includes configuration file templates. First, we copy and fill-in the `sig.toml` template:

```
# Set some variables that will be used below:
sigID=sig1
sigIP=127.0.0.1  # IP for the SCION address on which this SIG will bind
ISD=$(ls /etc/scion/gen/ | grep ISD | awk -F 'ISD' '{ print $2 }')
AS=$(ls /etc/scion/gen/ISD${ISD}/ | grep AS | awk -F 'AS' '{ print $2 }')

# Create a configuration directory for the SIG
sudo mkdir /etc/scion/gen/ISD${ISD}/AS${AS}/sig${ISD}-${AS}-1/

# Expand the placeholders in the sig.toml template and install it:
sed -e "s/\${ISD}/${ISD}/g;
        s/\${AS}/${AS}/g;
        s/\${IA}/${ISD}-${AS}/g;
        s/\${IAd}/${ISD}-${AS//_/:}/g;
        s/\${sigID}/${sigID}/g;
        s/\${sigIP}/${sigIP}/g;" < /usr/share/doc/scion-ip-gateway/templates/sig.config \
        | sudo tee \
         --output-error=exit /etc/scion/gen/ISD${ISD}/AS${AS}/sig${ISD}-${AS}-1/sig.toml
```

Each SIG requires traffic rules, in the form of a `json` configuration file. This configuration specifies IP prefixes that can be forwarded to a SIG in a remote AS. Create the traffic rules for the SIG at `/etc/scion/gen/ISD${ISD}/AS${AS}/sig${ISD}-${AS}-1/${sigID}.json`:

```
{
    "ASes": {
        "18-ffaa:0:1201": {
            "Nets": [
                "10.42.0.0/24"
            ]
        }
    },
    "ConfigVersion": 9001
}
```

Here, we tell the SIG to send all IP packets addressed to `10.42.0.0/24` to the AS `18-ffaa:0:1201`.

Finally, the topology files need to be updated to include a SIG entry. This entry is required so that the border routers can resolve the SIG service address. Insert the following snippet to the topology file of every border router in the AS (after replacing `${ISD}` and `${AS}` with the appropriate values):

```
"SIG": {
  "sig${ISD}-${AS}-1": {
    "Addrs": {
      "IPv4": {
        "Public": {
          "Addr": "127.0.0.1",
          "L4Port": 31056
        }
      }
    }
  }
},
```

For these changes in the topology files to take effect, the services need to be restarted:

```
sudo systemctl restart scionlab.target
```

RUNNING THE SIG AND CONFIGURING ROUTING

Start the SIG process:

```
sudo -u scion sig -config=/etc/scion/gen/ISD${ISD}/AS${AS}/sig${ISD}-${AS}-1/sig.toml &
```

Now we need to ensure that packets sent to an address in `10.42.0.0/24` actually pass through the SIG and that replies are also sent through the SIG. We have pre-configured the SIG in `18-ffaa:0:1201` with a mapping between the addresses in `172.16.0.0/12` and ASes between `17-ffaa:1:c00` and `20-ffaa:1:fff`. For a particular ISD-AS, we have configured a `/24` prefix, where the last 12 bits are obtained as follows:

- The 2 most-significant bits are determined by the ISD (0x0 for ISD17, …, 0x3 for ISD20)
- The remaining 10 bits are given by the 10 least-significant bits of the AS

For example, the prefix `172.17.162.0/24` corresponds to the AS `17-ffaa:1:da2`. It is a fun exercise to calculate the IP prefix for your AS by hand (or with a custom script), but you can also simply use the following one-liner:

```
a=${AS: -3: -2} b=${AS: -2} c=`echo "($ISD-17) * 4 + $((16#$a)) + 4" \
| bc` d=$((16#$b)) && echo "172.$c.$d.0/24"
```

You can set up a simple routing configuration, where only applications on SIG host can make use of the link:

```
# Assign an address in your IP prefix
sudo ip address add 172.XX.XXX.1 dev ${sigID}
# Setup route to 10.42.0.0/24 in sigA.json
sudo ip route add 10.42.0.0/24 dev ${sigID}
```

> **HINT**
> These address and route settings will only live as long as the sig tunnel device. As soon as the `sig` process terminates, this will be gone.

Now you should be able to `ping` the remote host

```
ping 10.42.0.1
```

> **WARNING**
> The MTU set on the SIG's tun device is unreliable or just plain wrong. Set a conservative value of, e.g., 1200 bytes:
>
> ```
> sudo ip link set mtu 1200 dev ${sigID}
> ```

You should now be able to fetch our website using `curl`:

```
curl http://10.42.0.1:4443
```

Partner exercises

Select a partner for the remaining exercises. (If you don't find a partner, you can use your two configured SCIONLab ASes.) You can choose any of the following exercises in any order.

Connect to the SCION apps of your partner

If you or your partner have set up any of the server apps in your AS, you can access your partner's apps in the same way as you have accessed the ones on the core ASes.

Bonus exercise: Set up a SIG connection to your partner's AS

You should already have set up a SIG in your AS and should be able to extend the configuration to also tunnel traffic between your AS and your partner's. Have a look at the general SIG tutorial for further explanations.

Bonus exercise: Set up an additional peering link to your partner's AS

> **Warning** this exercise requires some expertise with your network.

Similarly to the peering exercise above, you can configure a new peering interface to your partner. You will need to know their IA, their public IP and port. For this both you and your partner must have public IP addresses (or the ability to "punch a hole" in the firewall of your network or NAT setup).

- Open the topology file and add a new interface with the correct IA, public IP and port.
- Save the modified topology file and copy it into all subdirectories of `/etc/scion/gen/ISD*/AS*/`
- Restart SCION by calling `sudo systemctl restart scionlab.target`