Using the docker-compose based Moonshot demo

On this page you will find instructions on how to get and use the Moonshot docker-compose demo, which lets you see and test all of the components of Moonshot.

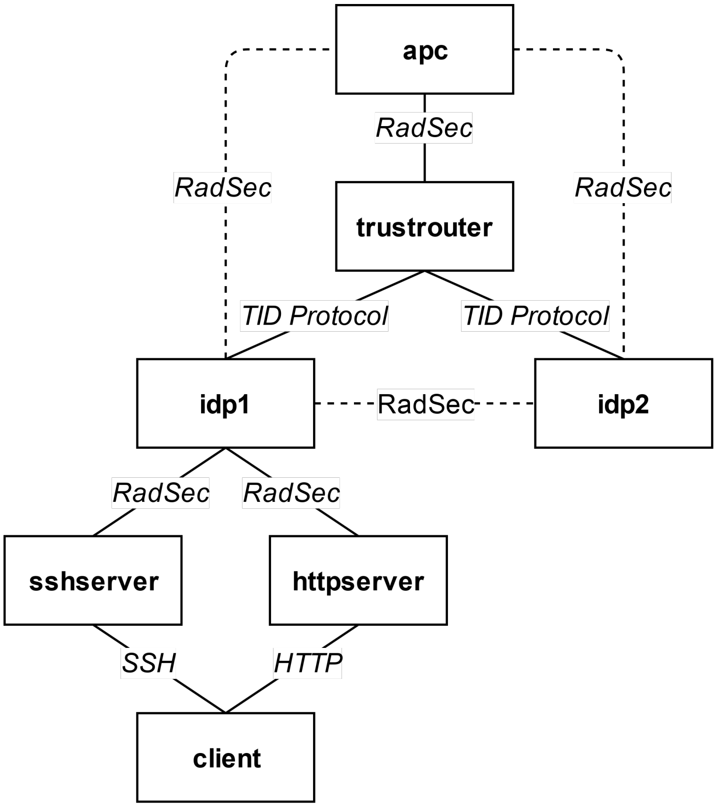
Contents

1. 1. Introduction

The docker-compose demo runs locally a small Moonshot infrastructure, including:

- One APC (`apc`)
- A Trust Router server (`trustrouter`)
- Two IDPs (`idp1` and `idp2`)
- An SSH server (`sshserver`)
- An HTTP server (`httpserver`)
- A Client (`client`)

The names in brackets correspond to the names of the docker-compose services. The following figure depicts how these containers are connected and what protocols are used for each connection:



Dashed lines represent RadSec connections established as a consequence of having used the `TID Protocol`.

2. 2. Prepare the host system

The only requirement to run this demo is having a Docker and docker-compose installation. Both are available for GNU/Linux, Windows and MacOS. See instructions on how to install it for your chosen platform (https://www.docker.com/).

If you do not want to install docker in your system or you do not have a Linux host but want to try the Moonshot GTK UI, you can use the Debian 10 Live DVD as a host, using the following steps:

1. Download the .ISO image from here.
2. Run the image, either on a physical machine or a VM.

   > Make sure the machine has at least 6GB of RAM, since the live system uses memory as storage.

3. Install docker and docker compose.

   ```
   sudo apt-get update
   sudo apt-get install docker docker-compose
   ```

4. Start docker

   ```
   sudo systemctl start docker
   ```

5. Assign Live user to the docker group and make sure group becomes active.

   ```
   sudo adduser user docker
   newgrp docker
   ```

3. 3. Get the demo files

Get the demo files from https://github.com/janetuk/moonshot_docker. Use the `Clone or download` button and follow instructions. If you choose downloading the ZIP file, please uncompress it to a known location.

4. 4. Start the services

Enter into the demo folder you got via `git` or from the ZIP file. The provided `docker-compose` file will build and start each individual service conforming the Moonshot infrastructure. To do so, simply execute the following command:

Build and start

```
docker-compose up -d --force-recreate --build
```

That will build the images, recreate the containers, and run them in the background.

The expected output is something similar to:

```
Creating moonshot_docker_idp2_1        ... done
Creating moonshot_docker_trustrouter_1 ... done
Creating moonshot_docker_client_1      ... done
Creating moonshot_docker_idp1_1        ... done
Creating moonshot_docker_sshserver_1   ... done
Creating moonshot_docker_httpserver_1  ... done
Creating moonshot_docker_apc_1         ... done
```

5. 5. Performing Moonshot authentications

Once all the services are up, you can log into the Client container and start performing Moonshot authentications.

5.1. 5.1. Getting into the Client container

To get an interactive bash session on the Client container, use the following command:

Get into the container

```
docker-compose exec client bash
```

That will give you a bash session as the root user.

5.1.1. 5.1.1. Providing access to host's display (only for Linux hosts)

If your host is running an Xorg desktop session, you can allow docker containers to access the display and run graphical applications by executing the following command on your host.

Allow local X connections

```
# run this on the host
xhost +local:
```

The, from within the Client container, set the appropriate value to the DISPLAY variable:

Set DISPLAY env var

```
# run this on the client container
export DISPLAY=:0
```

If you do not do this, Moonshot will use the TEXT UI instead.

5.2. 5.2. Testing SSH

1. From within the Client container, try to authenticate to the SSH server container by issuing the following command:

   Launch SSH connection

   ```
   ssh moonshot@sshserver
   ```

2. The Moonshot UI will pop up, asking you to select an identity. You can use the `Import` button on the right to import the `/config/user_credentials.xml` file. That will import the alice@test1.org and alice@test2.org identities.
3. Select one of them and click on the `Send` button. Note that using alice@test1.org will perform a local Moonshot authentication, that is, the Trust Router infrastructure will not be involved since the SSH server is directly connected to the IDP1. However, using alice@test2.org will imply a Moonshot authentication that will also involve the use of the Trust Router server and the APC, as IDP1 will need to ask for details about IDP2.
4. After having authenticated, you should see yourself logged into the `sshserver` as the `moonshot` user. You can verify both by using the `hostnam` and `whoami` system tools.

   SSH connection succeeded

   ```
   The programs included with the Debian GNU/Linux system are free software;
   the exact distribution terms for each program are described in the
   individual files in /usr/share/doc/*/copyright.

   Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
   permitted by applicable law.
   Last login: Tue Oct 29 12:02:21 2019 from 192.168.240.2
   $ whoami
   moonshot
   $ hostname
   sshserver
   ```

5.3. 5.3. Testing HTTP

1. From within the Client container, try to authenticate to the SSH server container by issuing the following command:

   Launch HTTP connection

   ```
   curl --negotiate -u ":" http://httpserver/protected/hello.cgi
   ```

2. The Moonshot UI will pop up, asking you to select an identity. Select one of them and click on the Send button.
3. After having authenticated, you should see something like this, including the username and attributes of the authenticated user.

   HTTP connection succeeded

   ```
   HTTP + Moonshot authentication as user 'moonshot' with the following attributes (in JSON format):
   {"name":"alice@test2.org","attributes":{
   "urn:ietf:params:gss:radius-attribute 79":{"authenticated":true,"complete":true,"values":[{"raw":"
   AwgABA==","display":null}]},
   "urn:ietf:params:gss:radius-attribute 80":{"authenticated":true,"complete":true,"values":[{"raw":"
   0DWDD6ZzPyHqEpzvuz48gw==","display":null}]},
   "urn:ietf:params:gss:radius-attribute 1":{"authenticated":true,"complete":true,"values":[{"raw":"
   YWxpY2VAdGVzdDIub3Jn","display":"alice@test2.org"}]},
   "urn:ietf:params:gss:federated-saml-assertion":{"authenticated":true,"complete":true,"values":[{"raw":"
   PG5zMDpBc3NlcnRpb24geG1sbnM6bnMwPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6YXNzZXJ0aW9uIiBJRD0iYzNhYjExMjQtN
   2EzNS00YTI1LWE0ZjctYjA5ZmQ0ODFiYzQ2IiBJc3N1ZUluc3RhbnQ9IjIwMTktMTAtMjlUMTI6MTQ6MTVaIiBWZXJzaW9uPSIyLjAiPj
   xuczA6SXNzdWVyPmh0dHBzOi8vdGVzdGlkcC5vcmcvPC9uczA6SXNzdWVyPjxuczA6U3ViamVjdD48bnMwOk5hbWVJRCBGb3JtYXQ9InV
   ybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpuYW1laWQtZm9ybWF0OnRyYW5zaWVudCI+NDM4MWVjMjItMjI4MS00NmViLTk1MGMtYWIy
   ODg2NGIxNTcwPC9uczA6TmFtZUlEPjxuczA6U3ViamVjdENvbmZpcm1hdGlvbiBNZXRob2Q9InVybjppZXRmOnBhcmFtczphYmZhYjpjb
   Tp1c2VyIi8+PC9uczA6U3ViamVjdD48bnMwOkNvbmRpdGlvbnMgTm90QmVmb3JlPSIyMDE5LTEwLTI5VDEyOjE0OjE1WiIgTm90T25Pck
   FmdGVyPSIyMDE5LTEwLTI5VDEzOjE0OjE1WiIvPjxuczA6QXR0cmlidXRlU3RhdGVtZW50IHhtbG5zOm5zMD0idXJuOm9hc2lzOm5hbWV
   zOnRjOlNBTUw6Mi4wOmFzc2VydGlvbiIgeG1sbnM6eHM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hIiB4bWxuczp4c2k9
   Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hLWluc3RhbmNlIj48bnMwOkF0dHJpYnV0ZSBOYW1lPSJ1cm46b2lkOjEuMy42L
   jEuNC4xLjU5MjMuMS4xLjEuNyIgTmFtZUZvcm1hdD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmF0dHJuYW1lLWZvcm1hdDp1cm
   kiPjxuczA6QXR0cmlidXRlVmFsdWUgeHNpOnR5cGU9InhzOnN0cmluZyI+bW9vbnNob3Q8L25zMDpBdHRyaWJ1dGVWYWx1ZT48L25zMDp
   BdHRyaWJ1dGU+PC9uczA6QXR0cmlidXRlU3RhdGVtZW50PjwvbnMwOkFzc2VydGlvbj4=","display":"<ns0:Assertion xmlns:
   ns0=\"urn:oasis:names:tc:SAML:2.0:assertion\" ID=\"c3ab1124-7a35-4a25-a4f7-b09fd481bc46\" IssueInstant=\"
   2019-10-29T12:14:15Z\" Version=\"2.0\"><ns0:Issuer>https://testidp.org/</ns0:Issuer><ns0:Subject><ns0:
   NameID Format=\"urn:oasis:names:tc:SAML:2.0:nameid-format:transient\">4381ec22-2281-46eb-950c-
   ab28864b1570</ns0:NameID><ns0:SubjectConfirmation Method=\"urn:ietf:params:abfab:cm:user\"/></ns0:
   Subject><ns0:Conditions NotBefore=\"2019-10-29T12:14:15Z\" NotOnOrAfter=\"2019-10-29T13:14:15Z\"/><ns0:
   AttributeStatement xmlns:ns0=\"urn:oasis:names:tc:SAML:2.0:assertion\" xmlns:xs=\"http://www.w3.org/2001
   /XMLSchema\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"><ns0:Attribute Name=\"urn:oid:
   1.3.6.1.4.1.5923.1.1.1.7\" NameFormat=\"urn:oasis:names:tc:SAML:2.0:attrname-format:uri\"><ns0:
   AttributeValue xsi:type=\"xs:string\">moonshot</ns0:AttributeValue></ns0:Attribute></ns0:
   AttributeStatement></ns0:Assertion>"}]},
   "urn:ietf:params:gss:federated-saml-attribute urn:oasis:names:tc:SAML:2.0:attrname-format:uri urn:oid:
   1.3.6.1.4.1.5923.1.1.1.7":{"authenticated":true,"complete":true,"values":[{"raw":"bW9vbnNob3Q=","
   display":"moonshot"}]},
   "urn:ietf:params:gss:federated-saml-nameid urn:oasis:names:tc:SAML:2.0:nameid-format:transient":
   {"authenticated":true,"complete":true,"values":[{"raw":"
   NDM4MWVjMjItMjI4MS00NmViLTk1MGMtYWIyODg2NGIxNTcw","display":"4381ec22-2281-46eb-950c-ab28864b1570"}]},
   "local-login-user":{"authenticated":true,"complete":true,"values":[{"raw":"bW9vbnNob3Q=","display":"
   moonshot"}]}}}
   ```

6. 6. Checking the logs

You can check the log output of the different services by using the `docker-compose logs` functionality. For instance, to check the Trust Router server logs, use:

Check IDP1 logs

```
docker-compose logs idp1
```