Lesson-End Project

Securing Communication Network with Bridge Network

Project agenda: To establish a secure and isolated communication channel between Docker containers for secure data exchange and improved application security

Description: Your company is experiencing a transition toward modernized containerized applications using Docker. To address this, you are undertaking a project that aims to set up a secure communication network for applications within Docker containers. This project involves creating a custom Docker bridge network, deploying multiple containers within this network, and demonstrating secure interaction between these containers.

Tools required: Docker and Ubuntu OS

Prerequisites: None

Expected deliverables: Secure Docker bridge network enabling isolated communication between containers

Steps to be followed:

1. Create a Docker container
2. Deploy multiple containers
3. Establish communication between the containers in a custom network
4. Connect and verify the container network communication

Step 1: Create a Docker container

1.1 Create a project folder and navigate to it using the following command:

mkdir lep4

cd lep4

```
sakshiguptasimp@ip-172-31-90-22:~$ mkdir lep4
sakshiguptasimp@ip-172-31-90-22:~$ cd lep4
sakshiguptasimp@ip-172-31-90-22:~/lep4$
```

1.2 Check Docker network information using the following command:

docker info | grep Network

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker info | grep Network
  Network: bridge host ipvlan macvlan null overlay
sakshiguptasimp@ip-172-31-90-22:~/lep4$
```

1.3 Create a Docker container named server1 using the following command:

docker run -it --name server1 ubuntu /bin/bash

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker run -it --name server1 ubuntu /bin/bash
root@8afa9aea3313:/#
```

The server1 container is created and running with an interactive terminal.

1.4 Execute the following command to update package lists in **server1**:

apt update -y

```
root@8afa9aea3313:/# apt update -y
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1641 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1081 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [44.6 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2067 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2107 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [61.2 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1358 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1920 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [80.9 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [33.3 kB]
Fetched 30.7 MB in 2s (15.0 MB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
12 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@8afa9aea3313:/#
```

1.4 Install the ping utility using the following command:

apt install iputils-ping -y

```
root@8afa9aea3313:/# apt install iputils-ping -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2-bin libpam-cap
0 upgraded, 3 newly installed, 0 to remove and 12 not upgraded.
Need to get 76.8 kB of archives.
After this operation, 280 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcap2-bin amd64 1:2.44-1ubuntu0.22.04.1 [26.0 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 iputils-ping amd64 3:20211215-1 [42.9 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpam-cap amd64 1:2.44-1ubuntu0.22.04.1 [7928 B]
Fetched 76.8 kB in 0s (1088 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package libcap2-bin.
(Reading database ... 4393 files and directories currently installed.)
Preparing to unpack .../libcap2-bin_1%3a2.44-1ubuntu0.22.04.1_amd64.deb ...
Unpacking libcap2-bin (1:2.44-1ubuntu0.22.04.1) ...
Selecting previously unselected package iputils-ping.
Preparing to unpack .../iputils-ping_3%3a20211215-1_amd64.deb ...
```

1.5 Test the internet connectivity using the following command:

ping google.com

```
root@8afa9aea3313:/# ping google.com
PING google.com (142.251.111.138) 56(84) bytes of data.
64 bytes from bk-in-f138.1e100.net (142.251.111.138): icmp_seq=1 ttl=57 time=1.46 ms
64 bytes from bk-in-f138.1e100.net (142.251.111.138): icmp_seq=2 ttl=57 time=1.47 ms
64 bytes from bk-in-f138.1e100.net (142.251.111.138): icmp_seq=3 ttl=57 time=1.51 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.461/1.481/1.510/0.020 ms
root@8afa9aea3313:/#
```

Note: Press **ctrl + P** and **ctrl + Q** to come out of the container

Step 2: Deploy multiple containers

2.1 Create a Docker container named **server2** using the following command:

docker run -it --name server2 ubuntu /bin/bash

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker run -it --name server2 ubuntu /bin/bash
root@1f9cd2275751:/#
```

2.2 To note down the IP address of server2, execute the following command:

hostname -i

```
root@ba80fba29705:/# hostname -i
172.17.0.3
```

2.3 Connect your terminal to the **server1** container using the following command:

docker attach server1

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker attach server1
root@8afa9aea3313:/#
```

2.4 Ping **server2** container's IP address using the following command:

ping 172.17.0.3

```
root@8afa9aea3313:/# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.105 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 172.17.0.3: icmp_seq=3 ttl=64 time=0.071 ms
^C
--- 172.17.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2051ms
rtt min/avg/max/mdev = 0.069/0.081/0.105/0.016 ms
root@8afa9aea3313:/#
```

Note: Press **ctrl + P** and **ctrl + Q** to come out of the container

Step 3: Establish communication between the containers in a custom network

3.1 Create a custom Docker network named **work** using the following command:

docker network create work

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker network create work
19149793a8b6bbb66a9fdea3dc2115185dd0ec6697edea5ec376728436270870
sakshiguptasimp@ip-172-31-90-22:~/lep4$
```

3.2 Verify the creation of the network using the following command:

docker network ls

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
15d4b69db9cb   bridge    bridge    local
bccf20b603ea   host      host      local
75466cd29e4e   none      null      local
19149793a8b6   work      bridge    local
sakshiguptasimp@ip-172-31-90-22:~/lep4$
```

3.3 Create **server3** container connected to the work network using the following command:

docker run -it --network work --name server3 ubuntu /bin/bash

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker run -it --network work --name server3 ubuntu /bin/bash
root@4f7b8ded3bc0:/#
```

3.4 To retrieve the IP address of **server3**, use the following command:
hostname -i

```
root@4f7b8ded3bc0:/# hostname -i
172.18.0.2
root@4f7b8ded3bc0:/#
```

3.5 Inside the **server3** container, update package lists using the following command:
apt update -y

```
root@4f7b8ded3bc0:/# apt update -y
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1081 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1641 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2067 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [44.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1357 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2161 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1966 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [61.2 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [33.3 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [80.9 kB]
Fetched 30.8 MB in 2s (15.6 MB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
12 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@4f7b8ded3bc0:/#
```

3.6 Install the ping utility inside the **server3** container using the following command:
apt install iputils-ping -y

```
root@4f7b8ded3bc0:/# apt install iputils-ping -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2-bin libpam-cap
0 upgraded, 3 newly installed, 0 to remove and 12 not upgraded.
Need to get 76.8 kB of archives.
After this operation, 280 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcap2-bin amd64 1:2.44-1ubuntu0.22.04.1 [26.0 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 iputils-ping amd64 3:20211215-1 [42.9 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpam-cap amd64 1:2.44-1ubuntu0.22.04.1 [7928 B]
Fetched 76.8 kB in 0s (1067 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package libcap2-bin.
(Reading database ... 4393 files and directories currently installed.)
Preparing to unpack .../libcap2-bin_1%3a2.44-1ubuntu0.22.04.1_amd64.deb ...
Unpacking libcap2-bin (1:2.44-1ubuntu0.22.04.1) ...
Selecting previously unselected package iputils-ping.
Preparing to unpack .../iputils-ping_3%3a20211215-1_amd64.deb ...
Unpacking iputils-ping (3:20211215-1) ...
Selecting previously unselected package libpam-cap:amd64.
Preparing to unpack .../libpam-cap_1%3a2.44-1ubuntu0.22.04.1_amd64.deb ...
Unpacking libpam-cap:amd64 (1:2.44-1ubuntu0.22.04.1) ...
```

Note: Press **ctrl + P** and **ctrl + Q** to come out of the container

Step 4: Connect and verify the container network communication

4.1 Connect the **server2** container to the work network using the following command:
docker network connect work server2

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker network connect work server2
sakshiguptasimp@ip-172-31-90-22:~/lep4$
```

4.2 Attach to server3 container using the following command:
docker attach server3

```
sakshiguptasimp@ip-172-31-90-22:~/lep4$ docker attach server3
root@4f7b8ded3bc0:/#
```

4.3 Verify communication with server2 by pinging its hostname using the following command:
ping server2

```
root@4f7b8ded3bc0:/# ping server2
PING server2 (172.18.0.3) 56(84) bytes of data.
64 bytes from server2.work (172.18.0.3): icmp_seq=1 ttl=64 time=0.111 ms
64 bytes from server2.work (172.18.0.3): icmp_seq=2 ttl=64 time=0.054 ms
64 bytes from server2.work (172.18.0.3): icmp_seq=3 ttl=64 time=0.057 ms
^C
--- server2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2016ms
rtt min/avg/max/mdev = 0.054/0.074/0.111/0.026 ms
root@4f7b8ded3bc0:/#
```

By following these steps, you have successfully established a secure and isolated communication channel between Docker containers for securing data exchange and improving application security.