Exercise 10.01 – Securing DSE

In this exercise you will learn how to secure a single instance of DataStax Distribution of Apache Cassandra (DDAC) version 5.1.15. This is a streamlined version of Apache Cassandra with additional DataStax utilities. This exercise will require a basic installation and startup of DDAC and then a thorough run-through of secure preventative measures. Understanding these basic steps will assist in locking down additional nodes and systems.

This exercise will cover the following items:

1. Confirm all prerequisite applications are installed on the Linux platform.
2. Perform a standard installation of DDAC based on the DSE installation instructions:
    a. https://docs.datastax.com/en/ddac/doc/datastax_enterprise/install/installDDAC.html
3. Use the *firewalld* utility to disallow access to any port except for those required by DSE as dictated by the DSE Security Checklist.
4. Confirm completed work using the GUI utility "firewall-config" to check the CLI syntax.
5. Perform both Netstat and Nmap scans to confirm all unnecessary ports are closed.

Step 1: Checking All Prerequisites

Before installing Apache Cassandra, make certain you have a supported Linux installation. Remember that RHEL 7, CentOS 7, Ubuntu LTS 12, 14, 16 or 18 are some of the suggested platforms for installing DDAC.

Confirm a supported Java version is installed. DDAC requires Oracle Java SE 8 (JRE or JDK) version *1.8.0_151* at the minimum. A Linux version of OpenJDK will also work.

Confirm a version of `openjdk` is installed.

Use the `rpm` command to identify a version is installed. What is the command to query the RPM database and look for a valid version?

```
rpm -qa | grep openjdk
```

If `openjdk` is not installed, use the yum command to install the latest. What is the command for this?

```
yum search openjdk
install -y java-1.8.0-openjdk java-1.8.0-openjdk-headless
```

Having confirmed Java is installed, run a java query from the command line to identify the version. What command do you use?

```
java -version
```

Download and install the EPEL (Extra Packages for Enterprise Linux) distribution for CentOS or RHEL. A quick Google search should show you how to download and install the repo for EPEL. What commands should you use?

```
yum -y install epel-release
yum repolist
```

Confirm Python 2.7 is installed. What command will show the version of Python?

```
rpm -q python
```

If Python is not installed, what command or commands will install Python?

```
yum -y install python
```

Step 2: Download, Install, and Configure DDAC

Download the tarball for DDAC from the following page:
https://downloads.datastax.com/#ddac

Place the downloaded tarball in a secure location prior to extracting the files. I like to use */usr/local/src/* as my working directory for source files.

Extract the files:

```
cp ddac-5.1.15-bin.tar.gz /usr/local/src/
cd /usr/local/src/
tar -xzvf ddac-5.1.15-bin.tar.gz
ln -s ddac-5.1.15 ddac
```

Define the data and logging directory locations, use one of the following options:

Default Directory Locations - No action is required. When the directory locations are excluded or commented out in the *cassandra.yaml*, Cassandra uses the default locations:

- data_file_directories: *installation_location/data/data*
- commitlog_directory: *installation_location/data/commitlog*
- saved_caches_directory: *installation_location/data/saved_caches*

- hints_directory: *installation_location/data/hints*
- cdc_raw_directory: *installation_location/data/cdc_raw*

Recommended Directory Locations - Most production deployments store data and logs in */var/lib/cassandra*. To use the recommended location, perform the following commands:

Create and change owners for the */var/lib/cassandra* directory:

```
mkdir -p /var/lib/cassandra
chown -R $USER:$GROUP /var/lib/cassandra
```

Tip: Make certain that the account that runs *cassandra* has write access to directory for data and logs. The subdirectories are automatically created.

Go the directory containing the *cassandra.yaml* file:

```
cd installation_location/conf
```

Uncomment the following lines in the *cassandra.yaml* file:

- data_file_directories: */var/lib/cassandra/data*
- commitlog_directory: */var/lib/cassandra/commitlog*
- saved_caches_directory: */var/lib/cassandra/saved_caches*
- hints_directory: */var/lib/cassandra/data/hints*
- cdc_raw_directory: */var/lib/cassandra/cdc_raw*

Tip: When using a minimal YAML, add the options as shown above.

Custom Location - To define custom data and logging directory locations:

Create the directories for data and logging directories. For example:

```
mkdir /var/lib/cassandra/data &&
mkdir /var/lib/cassandra/commitlog &&
mkdir /var/lib/cassandra/saved_caches &&
mkdir /var/lib/cassandra/data/hints &&
mkdir /var/lib/cassandra/cdc_raw
```

Tip: Ensure that the account that runs *cassandra* has write access to directory for data and logs.

Go the directory containing the *cassandra.yaml* file:

```
cd installation_location/conf
```

Edit the following lines in the *cassandra.yaml* file:

```
data_file_directories: /var/lib/cassandra/data
commitlog_directory: /var/lib/cassandra/commitlog
saved_caches_directory: /var/lib/cassandra/saved_caches
hints_directory: /var/lib/cassandra/data/hints
cdc_raw_directory: /var/lib/cassandra/cdc_raw
```

Optional: Single-node cluster installations only.

Start Cassandra from the installation directory:

```
bin/cassandra
```

From the installation directory, verify Cassandra is running:

```
$ bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens       Owns (effective)  Host ID
Rack
UN  127.0.0.1  103.53 KiB  256          100.0%            729f3d84-
e513-4ffa-98cc-f5a5f850aca3  rack1
```

Step 3: Customize Firewalld for Cassandra

After confirming Cassandra is up and running, start prepping the system to local down all unnecessary ports.

What is the command to show al open ports available using *firewalld*?

```
# firewall-cmd --zone=public --permanent --list-services
ssh http https
```

There should only be two or three ports available. On my test system, I see only the following:

```
ssh http https
```

Using the security checklist provided on the DataStax site (https://docs.datastax.com/en/security/6.7/security/secFirewallPorts.html) begin the process of locking down and creating custom ports, if needed.

What are the commands needed to added additional ports to the public zone, i.e., the public ports for DSE in firewalld? Given that the standard port used by DSE is 9042, what would the syntax be for the `firewalld-cmd`?

```
# firewalld-cmd --zone=public --permanent --add-port=9042/tcp
success
```

How would you check to verify this port was added?

```
# firewall-cmd --zone=public --permanent --list-ports
9042/tcp
```

Note: When adding unique ports, you must use `--add-port=` and not the `--add-service=` command.

Add any additional ports needed by the server as required by any of the other utilities that come with DDAC, i.e., Spark, Graph, DSEFS Inter-node communication, etc. Consult the following page for a complete list of required ports and services:

https://docs.datastax.com/en/security/6.7/security/secFirewallPorts.html

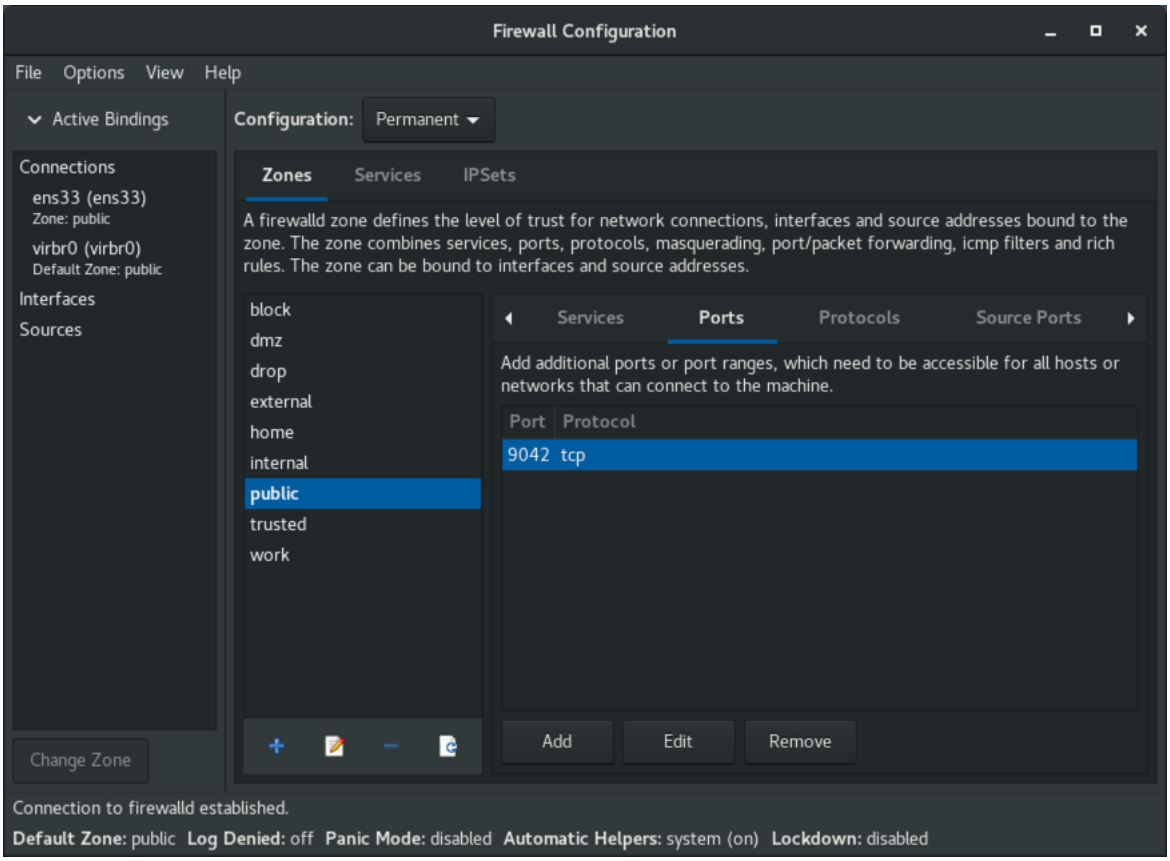
Step 4: Check Work Using the GUI firewall-config Utility

Having added some ports and services, it is a good practice to check your work before enabling it in production. Use the firewalld GUI utility to review all open ports and services.

From a root CLI prompt, launch the following command:

```
firewall-config
```

Make certain the `permanent` option is enabled under the *Configuration* menu. A GUI application such as the following should appear:



Note: You might be asking, why did we go to all the trouble to learn the `firewall-cmd` command line options when there is a GUI app? Good question… In most instances, servers do not have a GUI interface installed. This results in extra overhead and additional system requirements. Also, GUI applications are only as effective as the person who wrote them. The true power to manage an application lies in the command line. When working on DSE or Linux servers, access will most likely be CLI only.

Tip: Consider using the GUI *only* if required for checking your work. Otherwise, do everything using the CLI.

Step 5: Scanning Ports Using Netstat and Nmap

Manually test the ports to confirm no additional ports are open and listening. A useful tool for monitoring a local system is the `netstat` command. Run the following command and identify any and all listening ports:

```
netstat -an | grep LISTEN | less
```

Can you identify all the ports running? Is Cassandra running? Is the correct port specified?

A more intrusive tool is Nmap. It was actively scan and alert on all ports up and running and will attempt to fingerprint what applications are running on the open ports.

Nmap is extremely useful and can be configured to scan and resolve a host of issues. Execute the following command to perform a fast scan against the local server and query only the 100 most popular ports:

```
$ nmap -F 127.0.0.1

Starting Nmap 6.40 ( http://nmap.org ) at 2019-06-28 13:02 MDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0021s latency).
Not shown: 96 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
111/tcp open  rpcbind
631/tcp open  ipp

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
```

Notice it picked up additional open ports. However, it did not detect the default DSE port of 9042. What command might be used to scan for a specific port?

```
$ nmap -p 9042 127.0.0.1

Starting Nmap 6.40 ( http://nmap.org ) at 2019-06-28 13:06 MDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00017s latency).
PORT     STATE SERVICE
9042/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

Note that this scan did pick up the listening port, but only because we knew it was running. To scan all 55,536 available ports, running the following command:

```
$ nmap -p- 127.0.0.1

Starting Nmap 6.40 ( http://nmap.org ) at 2019-06-28 13:08 MDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00047s latency).
Not shown: 65524 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
111/tcp   open  rpcbind
631/tcp   open  ipp
7000/tcp  open  afs3-fileserver
7199/tcp  open  unknown
9042/tcp  open  unknown
36932/tcp open  unknown
```

```
41768/tcp open  unknown
45788/tcp open  unknown
59400/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 2.53 seconds
```

This is more like it. Ports greater than 1024 are now showing up and the default Cassandra port is available.

Go ahead and remove TCP port 9042 from the list of available and listening ports.

```
# firewall-cmd --zone=public --permanent --remove-port=9042/tcp
Success
```

Run the netstat and nmap commands again. Does this port show up on either of the two? If so, why? Or why not?