Assignment 3

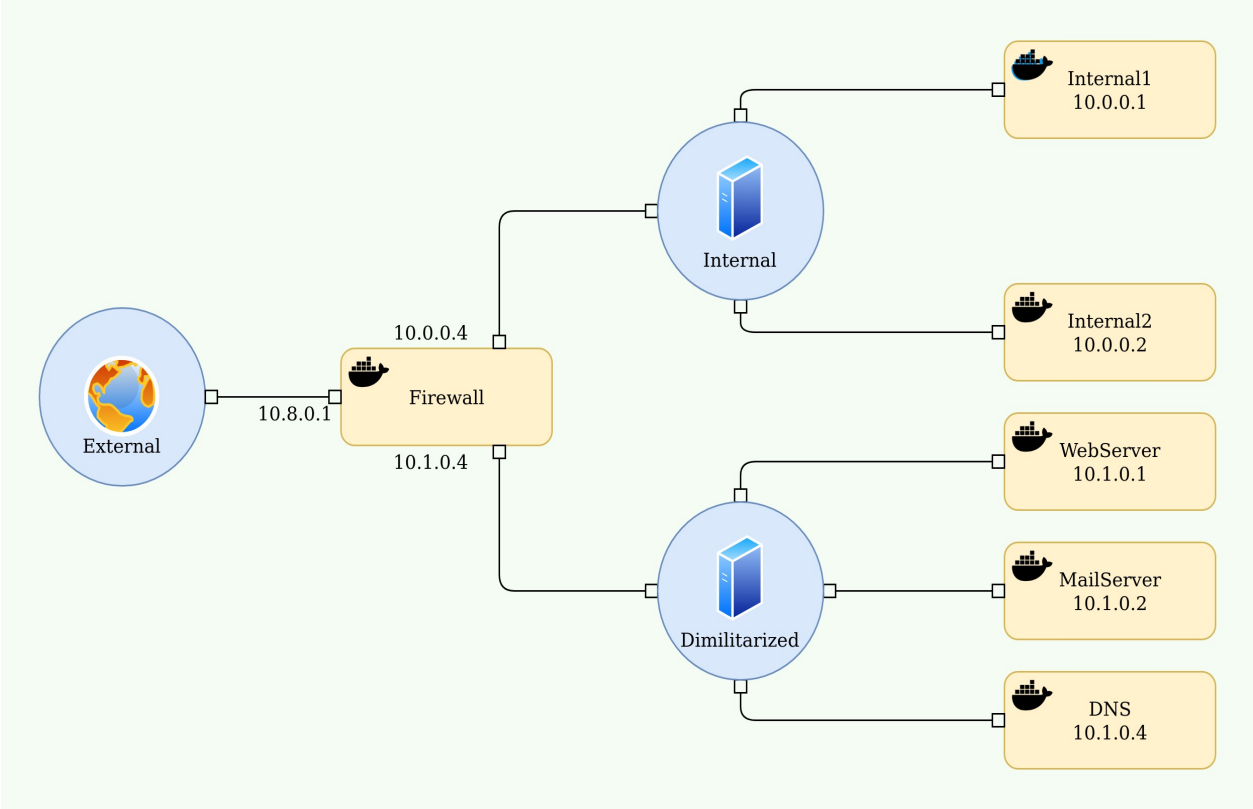
Group-14
B18048, B18190, B18115

Introduction

We created a network simulation as shown in the Image1, where we controlled all the data transfer to and from Internal and DMZ zone by using iptables in the gateway Container. We used docker containers to isolate the processes, apache to host the webpages, and postfix to create the mail server. See the youtube demo

Creating the Given Network Set-Up

- Create 2 internal networks named Internal and Dimilitarized.
  - `docker network create --internal --gateway=10.0.0.4 --subnet=10.0.0.0/16 Internal`
  - `docker network create --internal --gateway=10.1.0.4 --subnet=10.1.0.0/16 Dimilitarized`
  - the gateway tag creates a virtual network adapter to the host with ip = gateway ip.
- Create 2 dockers Connected to Internal network only, named internal1 and internal2.
- Create 3 dockers Connected to Dimilitarized network only, named webserver, mailserver and dns.
- While creating the dockers we will connect them to the respective networks as per the diagram.



- Also Set Static IPs for dockers/Network-Adapters as given below.

Docker/Network-Adapter	**Static-IP**
internal1 (Internal)	10.0.0.1
internal2 (Internal)	10.0.0.2
webserver (DMZ)	10.1.0.1
mailserver (DMZ)	10.1.0.2
dns (DMZ)	10.1.0.3
firewall (Internal)	10.0.0.4
firewall (Dimilitarized)	10.1.0.4
firewall (External)	10.8.0.1

Steps to test the networks a container can access

- get the container id from `docker ps`
- open the shell of the respective container using `docker exec -it <container_ID> /bin/bash`
- Test the network using ping.

Set-Up Internal Dockers

- Created the required `Dockerfile` in a new folder named internal.
- open terminal in the new folder if not already and run `docker build -t internal .` . this will create an image with name `internal`.
- the command `docker run internal` is used to start a new docker container with image `internal`.
- use the `--network` tag to configure which networks will the docker file be connected to.
- The `-d` tag is to run the container in a detached state from the terminal
- run `docker run --network Internal --ip 10.0.0.1 --dns 10.1.0.3 --name internal1 -d internal` (add the –rm tag while debugging).
- run `docker run --network Internal --ip 10.0.0.2 --dns 10.1.0.3 --name internal2 -d internal`
- test connections using ping

Internal > Dockerfile

```
FROM ubuntu:18.04

RUN apt-get update -y
RUN apt-get install -y iputils-ping

WORKDIR ~/app
COPY . .

CMD ["/bin/bash"]
```

Set-Up webserver docker

- In a new folder created a file named `Dockerfile`.
- Copied my webpage(HTMLs + resources) to the new folder.
- Created my domain and sub-domain configuration files (hellfire.conf) as shown in the figures below.
- In the Dockerfile added commands to
  - copy `hellfire.conf` to `/etc/apache2/sites-available/hellfire.conf`.
  - copy the domain folder `hellfire` to `/var/www/hellfire`.
  - disable the apache2 default page using `a2dissite 000-default.conf`.
  - enable the webpages whose conf were added `a2ensite hellfire.conf`.
- add the apache2 runer to CMD in the Dockerfile `CMD ["/usr/sbin/apache2ctl", "-DFOREGROUND"]`
- build the docker image `docker builld -t webserver .`
- start the container using `docker run --network Dimilitarized --ip 10.1.0.1 --dns 10.1.0.3 --name web_server webserver`
- test if the webpages are being hosted using `curl ashwin.hellfire.net --resolve 'ashwin.hellfire.net:80:10.1.0.1'`, here we have to specify how the curl needs to resolve the domain name as our DNS is not set up yet.

WebServer > hellfire.conf

```
# Domain : hellfire.net
<VirtualHost *:80>
    ServerAdmin ashwin@mail.hellfire.net
    ServerName hellfire.net
    ServerAlias www.hellfire.net
    DocumentRoot /var/www/hellfire
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>


# SubDomain : ashwin.hellfire.net
<VirtualHost *:80>
    ServerAdmin ashwin@mail.hellfire.net
    ServerName ashwin.hellfire.net
    ServerAlias www.ashwin.hellfire.net
    DocumentRoot /var/www/hellfire/ashwin
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Set-up DNS docker

- create the required `Dockerfile`.
- in the `Dockerfile` add commands to install bind9, bind9-utils, dnsutils and nano
- build the image using `docker build -t dns_base .`
- start a new container using `docker run -d --name dns_server dns_base`. Not attached the network as we need to work with the internet and right now the dmz network do not have internet access.
- now open terminal in the container using `docker exec -it dns_server /bin/bash`
- Set the dns as a caching Server
  - In the file /etc/default/bind9, we need to the OPTIONS variable to " -u bind -4 ". This willset Bind to ipv4 mode
  - Edit `/etc/bind/named.conf.options`.

- restart the bind9 service by running `/etc/init.d/bind9 restart`

- Set-Up Primary DNS Server
  - Edit `/etc/bind/named.conf.local`.
  - Add `/etc/bind/zones/for.hellfire.net`
  - Add `/etc/bind/zones/rev.hellfire.net`
  - restart the bind9 service by running `/etc/init.d/bind9 restart`
- disconnect the docker from the current network (bridge) `docker network disconnect bridge dns_server`
- connect the docker to `Dimilitaried` network `docker network connect --ip 10.1.0.3 --dns 10.1.0.3 Dimilitarized dns_server`
- test the connections using `curl hellfire.net`. if it dosn't work check if the dns server is specified in `/etc/resolv.conf`
- similarly specify dns to all other dockers.

DNS > for.hellfire.net

```
$TTL 86400
@   IN  SOA     hellfire.net. root.hellfire.net. (
    2011071001  ;Serial
    3600        ;Refresh
    1800        ;Retry
    604800      ;Expire
    86400       ;Minimum TTL
)
@       IN  NS          hellfire.net.
@       IN  A           10.1.0.1
ashwin  IN  A           10.1.0.1
saksham IN  A           10.1.0.1
keshav  IN  A           10.1.0.1
mail    IN  A           10.1.0.2
@       IN  MX          mail
```

DNS > rev.hellfire.net

```
$TTL 86400
@   IN  SOA     hellfire.net. root.hellfire.net. (
    2011071002  ;Serial
    3600        ;Refresh
    1800        ;Retry
    604800      ;Expire
    86400       ;Minimum TTL
)
@       IN  NS          hellfire.net.
@       IN  PTR         hellfire.net.
hellfire.net. IN A      10.1.0.1
ashwin  IN  A           10.1.0.1
saksham IN  A           10.1.0.1
keshav  IN  A           10.1.0.1
mail    IN  A           10.1.0.2
1       IN  PTR         hellfire.net.
1       IN  PTR         ashwin.hellfire.net.
1       IN  PTR         saksham.hellfire.net.
1       IN  PTR         keshav.hellfire.net.
2     IN  PTR           mail.hellfire.net.
```

DNS > named.conf.local

```
zone "hellfire.net" {
    type master;
    file "/etc/bind/zones/for.hellfire.net";
    allow-transfer {none;};
};

zone "0.1.10.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/rev.hellfire.net";
    allow-transfer {none;};
};
```

Set-Up mailserver docker

- Created the required `Dockerfile` in a new folder named MailServer.
- cd into the new folder.
- run `docker build -t mailserver .`
- install `postfix mailutils`
- configure postfix from `etc/postfix/00-cinfig.yaml`
- restart postfix
- run `mail root@hellfire.net`, add cc, subject and content of the mail
- run `mail --user=root` to see those mails.

Set-Up firewall

- since Docker uses my system's kernel and iptables are part of my kernel I have to set up the ip-tables in my system as well.
- run `ifconfig` in your pc and note ip with network adpter names.
    - br-4c4f7259bfb4 : 10.0.0.4
    - br-df4d2c05b54c : 10.1.0.4
    - wlp3s0 : 192.168.0.100
    - docker0 : 172.17.0.1
- run `sudo iptables --list` to see the already present rules and understand them.
- all the changes that are to be done in the DOCKER-USER Chain. it run before other docker rules. (in general)
- save the current docker config using `iptables-save > /path/to/backup/file`
- Clear the iptables using `iptables -F`
- Configure iptables to ACCEPT only incomming and DROP all outgoing/input packets by default.

    - `iptables -P FORWARD DROP`
    - `iptables -P INPUT DROP`

    - `iptables -P OUTPUT ACCEPT`

    - enabling internet in the computer : `iptables -A INPUT -d 192.168.0.0/8 -j ACCEPT`

    - test connections via `ping 8.8.8.8` from internal1, web_server and host pc.

    - Enable internet on Internal zone by adding Network Address Translation rules in iptables

        - `iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o wlp3s0 -j MASQUERADE`
        - `iptables -A FORWARD -s 10.0.0.0/24 -o enp0s9 -j ACCEPT`
        - `iptables -A FORWARD -m state --state ESTABLISHED -i wlp3s0 -d 10.0.0.0/24 -j ACCEPT`
        - `iptables -A FORWARD -m state --state RELATED -i wlp3s0 -d 10.0.0.0/24 -j ACCEPT`

    - try ping google.com from some internal VM . it should work.

    - Create Custom Chain to Enable internet in dmz Zone

        - `iptables -t nat -A POSTROUTING -s 10.1.0.0/24 -o wlp3s0 -j MASQUERADE`
        - `iptables -N chain-dmz-internet`
        - `iptables -A chain-dmz-internet -s 10.1.0.0/24 -o wlp3s0 -j ACCEPT`
        - `iptables -A chain-dmz-internet -m state --state ESTABLISHED -i wlp3s0 -j ACCEPT`
        - `iptables -A chain-dmz-internet -m state --state RELATED -i wlp3s0 -j ACCEPT`

    - Now run `iptables -A FORWARD -j chain-dmz_internet` to enable internet and `iptables -D FORWARD -j chain-dmz_internet` to disable internet in DMZ Zone.

    - Enable Port Forwarding for web, mail and DNS Servers such that they are accessible from the outside
        - `iptables -t nat -A PREROUTING -i wlp3s0 -p tcp --dport 80 -j DNAT --to-destination 10.1.0.1`
        - `iptables -t nat -A POSTROUTING -o wlp3s0 -p tcp --dport 80 -j SNAT --to-source 10.1.0.1`
        - `iptables -A FORWARD -p tcp --dport 80 -s 10.1.0.1 -j ACCEPT`

        - `iptables -A FORWARD -p tcp --dport 80 -d 10.1.0.1 -j ACCEPT`

- iptables -t nat -A PREROUTING -i wlp3s0 -p tcp --dport 25 -j DNAT --to-destination 10.1.0.2

- iptables -t nat -A POSTROUTING -o wlp3s0 -p tcp --dport 25 -j SNAT --to-source 10.1.0.2
- iptables -A FORWARD -p tcp --dport 25 -s 10.1.0.2 -j ACCEPT

- iptables -A FORWARD -p tcp --dport 25 -d 10.1.0.2 -j ACCEPT

- iptables -t nat -A PREROUTING -i wlp3s0 -p tcp --dport 53 -j DNAT --to-destination 10.1.0.3

- iptables -t nat -A POSTROUTING -o wlp3s0 -p tcp --dport 53 -j SNAT --to-source 10.1.0.3
- iptables -A FORWARD -p tcp --dport 53 -s 10.1.0.3 -j ACCEPT

- iptables -A FORWARD -p tcp --dport 53 -d 10.1.0.3 -j ACCEPT

- iptables -t nat -A PREROUTING -i wlp3s0 -p udp --dport 53 -j DNAT --to-destination 10.1.0.3

- iptables -t nat -A POSTROUTING -o wlp3s0 -p udp --dport 53 -j SNAT --to-source 10.1.0.3
- iptables -A FORWARD -p udp --dport 53 -s 10.1.0.3 -j ACCEPT
- iptables -A FORWARD -p udp --dport 53 -d 10.1.0.3 -j ACCEPT