SEED Lab: DNS In a Box

1 Lab Overview

DNS (Domain Name System) is the Internet's phone book; it translates hostnames to IP addresses (and vice versa). This translation is through DNS resolution, which happens behind the scene. The resolution process involves many nameservers, including root servers, TLD servers, and final domain servers. These nameservers form the entire DNS system, which is an essential infrastructure for the Internet.

To help students understand how these nameservers work together to form the infrastructure, we will create a miniature DNS system called *DNS in a Box*. As suggested by its name, the entire DNS system, which consists of multiple nameservers, runs inside a single machine. This is made possible by the container technology.

Even though this system is small, it has all the essential elements of a real DNS infrastructure. By building such a system, students will have a deeper understanding of how the DNS actually works. Although this lab is not a security lab, it is the basis for several SEED labs. This lab covers the following topics:

- DNS and how it works
- The DNS query process
- Root and TLD servers
- Docker container, docker compose

Readings and videos. Detailed coverage of the DNS protocol can be found in the following:

- Chapter 18 of the SEED Book, *Computer & Internet Security: A Hands-on Approach*, 2nd Edition, by Wenliang Du. See details at `https://www.handsonsecurity.net`.
- Section 7 of the SEED Lecture, *Internet Security: A Hands-on Approach*, by Wenliang Du. See details at `https://www.handsonsecurity.net/video.html`.

Lab environment. This lab has been tested on our pre-built Ubuntu 20.04 VM, which can be downloaded from the SEED website. Since we use containers to set up the lab environment, this lab does not depend too much on our SEED VM. You can do this lab using other VMs or physical machines.

2 Lab Setup

In this lab, we will build a simplified DNS infrastructure. We will start with a small one, and then gradually make it bigger. The first DNS system that we build consists of four nameservers, each one representing a specific role in the DNS infrastructure. In the real world, these nameservers are on different networks, but in this lab, for the sake of simplicity, we place them on the same network. Figure 1 depicts the system setup. We will use containers to run these nameservers.

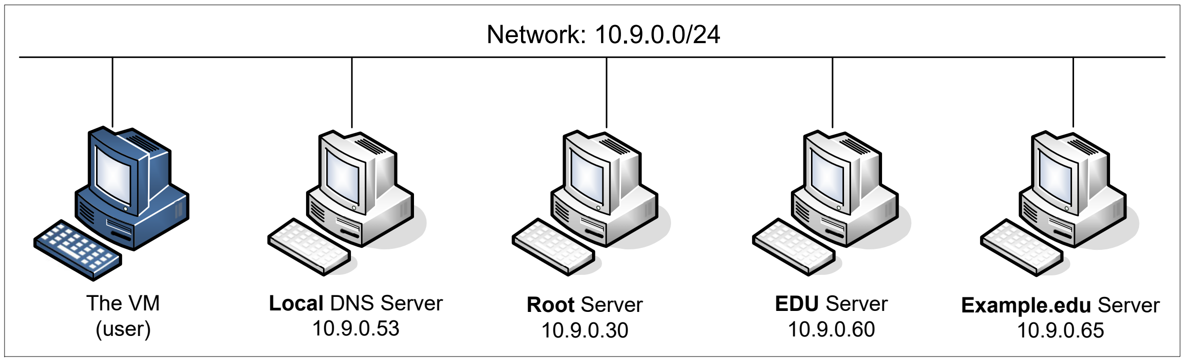

Figure 1: A simplified DNS infrastructure

2.1 Container Setup and Commands

Please download the `Labsetup.zip` file to your VM from the lab's website, unzip it, enter the `Labsetup` folder, and use the `docker-compose.yml` file to set up the lab environment. Detailed explanation of the content in this file and all the involved `Dockerfile` can be found from the user manual, which is linked to the website of this lab. If this is the first time you set up a SEED lab environment using containers, it is very important that you read the user manual.

In the following, we list some of the commonly used commands related to Docker and Compose. Since we are going to use these commands very frequently, we have created aliases for them in the `.bashrc` file (in our provided SEEDUbuntu 20.04 VM).

```
$ docker-compose build  # Build the container image
$ docker-compose up     # Start the container
$ docker-compose down   # Shut down the container

// Aliases for the Compose commands above
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

All the containers will be running in the background. To run commands on a container, we often need to get a shell on that container. We first need to use the "`docker ps`" command to find out the ID of the container, and then use "`docker exec`" to start a shell on that container. We have created aliases for them in the `.bashrc` file.

```
$ dockps        // Alias for: docker ps --format "{{.ID}}  {{.Names}}"
$ docksh <id>   // Alias for: docker exec -it <id> /bin/bash

// The following example shows how to get a shell inside hostC
$ dockps
b1004832e275  hostA-10.9.0.5
0af4ea7a3e2e  hostB-10.9.0.6
9652715c8e0a  hostC-10.9.0.7

$ docksh 96
root@9652715c8e0a:/#
```

```
// Note: If a docker command requires a container ID, you do not need to
//       type the entire ID string. Typing the first few characters will
//       be sufficient, as long as they are unique among all the containers.
```

If you encounter problems when setting up the lab environment, please read the "Common Problems" section of the manual for potential solutions.

2.2 The Docker Compose File

Our DNS system includes the four nameservers. The purpose of these four nameservers are summarized in the following:

- Local DNS server: conduct the DNS resolution process for other computers.
- Root server: a nameserver serving the root zone.
- `edu` server: nameserver serving the `edu` zone, a Top Level Domain (TLD).
- `example.edu` server: nameserver serving the `example.edu` zone.

We use a container for each of the nameservers above. How these containers are built and run is described in the `docker-compose.yml` file, which is included in the lab setup file. Inside this file, we create a network `10.9.0.0/24` and four containers: each entry in the `services` section represents a container. We also assign IP addresses to those containers. See the following snippet:

```
services:
  seed_base_router:                      ✰
    build: ./base_image
    image: seed-base-image-bind
    container_name: seed-base-bind
    command: " echo 'exiting ...' "

  example_edu_server:
    build:
        context: ./nameserver
        args:
            BIND_CONF_DIR: edu.example
    image: example-edu-server
    container_name: example-edu-10.9.0.65
    tty: true
    networks:
      seed-net:
        ipv4_address: 10.9.0.65

  edu_server:        ... (omitted, similar to example_edu_server) ...
        ipv4_address: 10.9.0.60
  root_server:       ... (omitted, similar to example_edu_server) ...
        ipv4_address: 10.9.0.30
  local_dns_server:  ... (omitted, similar to example_edu_server) ...
        ipv4_address: 10.9.0.53
```

Line ✰ specifies a base image, which basically builds an image based on the `handsonsecurity/seed-server:bind` image we placed on Docker Hub (see `Dockerfile` in the `base_image` folder). Although all the nameservers can directly build their containers using this base image from the Docker Hub, to reduce the number of access to the Docker Hub (the company puts a limit on how many accesses one can make during a six-hour time window), we first build a local base image using the one from Docker Hub, and then use the local base image to build the rest of the container images in this lab.

2.3 Container Images

Except for the local DNS server, all the nameservers use the same folder (`nameserver`) to build Docker images. The `Dockerfile` is shown in the following:

```
FROM seed-base-image-bind                             ❶

ARG BIND_CONF_DIR

# Copy the BIND confirguration files
COPY named.conf named.conf.options  /etc/bind/    ❷
COPY ${BIND_CONF_DIR}     /etc/bind/              ❸

# Start the nameserver
CMD service named start  && tail -f /dev/null
```

Line ❶ indicates this is local image (the base image). Line ❷ copies the BIND 9's configuration files to the `/etc/bind` folder inside the container. These two files are the same for all the nameservers, but each nameserver's does have its own zone files. That is why we use the `BIND_CONF_DIR` argument (Line ❸) to indicate which configuration folder should be used (there is a folder for each nameserver). The value of `BIND_CONF_DIR` is set in the Compose file.

Local DNS server. The container image for the local DNS server is quite similar to that in the nameservers. We will explain the difference later, as building the image is one of the tasks.

3 Task l: Building the nameserver for `example.edu`

In this task, we will build a nameserver for a domain called `example.edu`. The container files are put inside the `nameserver` folder. The configuration files for this particular domain is placed inside the `edu.example` sub-folder.

Step 1. Adding the zone entry . To host a server for the `example.edu` domain, we need to add a zone entry to the BIND 9's configuration file `named.conf`, so it knows what zones it is going to host. For convenience, we will add the entry to the `named.conf.seedlabs` file, which is included in our modified `named.conf` file.

This entry indicates that the current nameserver is the master server for this domain, and the zone file is specified in the `file` entry.

```
zone "example.edu" {
        type master;
        file "/etc/bind/example.edu.db";
};
```

Step 2. Creating the zone file . When building the docker image, the zone file `example.edu.db` will be copied into the `/etc/bind` folder of the container. To get students started, we have provided some information in this zone file, but students need to make all then necessary changes.

```
$TTL 3D
@       IN      SOA   ns.example.edu. admin.example.edu. (
                2008111001
                8H
                2H
                4W
                1D)

; Records for this nameserver (you need to make changes)
@               IN   NS    ns.example.edu.
ns.example.edu  IN   A     1.2.3.4


; IP addresses for the hostnames in the example.du domain
@       IN      A     1.2.3.5
www     IN      A     1.2.3.5
xyz     IN      A     1.2.3.6
*       IN      A     1.2.3.7
```

Step 3. Testing. Using `docker-compose` commands to build and start all the containers. Once the containers are running, run the following command from your VM (i.e., outside of the container). We use the `@10.9.0.65` option to send our DNS query directly to texttt10.9.0.65, which is the IP address of the `example.edu` nameserver in our setup. If everything is done correctly, you can get the IP address specified in your zone file. Please report your observations.

```
$ dig @10.9.0.65 www.example.edu
...
;; ANSWER SECTION:
www.example.edu.    259200   IN  A   1.2.3.5
...
```

4 Task 2: Building the edu TLD server

In this task, we will build a nameserver for a TLD domain, the `edu` domain. We will modify the files in the `nameserver/edu` folder. First, we need to add the following zone entry to the `named.conf.seedlabs` file. This entry indicates that the current nameserver is the master server for this domain, and the zone file is specified in the `file` entry.

```
zone "edu" {
        type master;
        file "/etc/bind/edu.db";
};
```

Creating the zone file. Next, we need to add records to the zone file. We have already created the zone file `edu.db`, but it is incomplete. Students should make all the necessary changes.

All the nameservers within the `edu` domain must register their nameservers with this TLD server; otherwise, nobody can find them. As an example, we have added two records for the `syr.edu` domain: an `NS` record and an `A` record. The `NS` record specifies the nameserver for the `syr.edu` domain, while the `A` record specifies the IP address of the nameserver. The data used in the records are real data, as it simulates the fact the `syr.edu` domain has registered itself with our `edu` TLD server.

```
; Real records for syr.edu
syr.edu.                IN      NS      ns1.syr.edu.
ns1.syr.edu.            IN      A       128.230.12.8
```

Lab Task. Students need to do the following tasks: (1) register your `example.edu` nameserver with this TLD server; (2) choose three real domain names, and register them with this TLD server. You can find the nameserver of a domain using "`dig <name> NS`" (e.g., "`dig syr.edu NS`").

Using `docker-compose` commands, students can build and start all the containers. Once the containers are running, run the following `dig` commands from your VM to send DNS queries to the `edu` nameserver (its IP address is `10.9.0.60` in our setup). Please report your observations.

```
$ dig @10.9.0.60 www.example.edu
$ dig @10.9.0.60 www.syr.edu
$ dig @10.9.0.60 <the other three domains you have chosen>
$ dig @10.9.0.60 <a domain not included in your zone file>
```

It should be noted, the `edu` TLD server is configured not to conduct the recursive query, so it will only tell you the nameserver for the domain name in the query; it will not resolve the query for you. In the last command, you should try some domains (ended with `edu`) that are not included in your zone file.

5 Task 4: Building root server

In this task, we will build a nameserver for the root zone. All TLD nameservers need to register with the root nameserver, so they can be found in the DNS query process. In our container, the root zone is defined in `bind.conf.seedlabs`, which loads the records from the zone file `root.db`.

For every TLD zone that we would like to include in our miniature DNS system, we need to add at least two records in the zone file, including an `NS` record and an `A` record. In the following example, we have added the nameserver information for the `com` and `net` zones (the data in the records are real data). These two zones are managed by the same company, so they share the same set of nameservers (we have only included one nameserver).

```
com.                 IN   NS   a.gtld-servers.net.
net.                 IN   NS   a.gtld-servers.net.
a.gtld-servers.net.  IN   A    192.5.6.30
```

Lab task. Students need to add records to the zone file `root.db` to satisfy the following requirements:

- Register your `edu` nameserver with this root server.
- Choose two more real TLD names, and register them with this TLD server. One of the TLDs should be a country-code TLD (ccTLD), which is the root for a country. You can find the nameserver of a TLD domain using "`dig <tld> NS`" (e.g., "`dig com NS`"). You can also find all the TLD information from `https://www.internic.net/domain/root.zone`.

Using `docker-compose` commands, students can build and start all the containers. Once the containers are running, run the `dig` command from your VM to send DNS queries to the root nameserver (using "`dig @10.9.0.30`"). Please use the testing to demonstrate that your DNS setup is working as expected.

6 Task 5: Building the Local DNS Server

When a computer needs to resolve the IP address from a hostname (or vice versa), it sends a request to its helper, which is called local DNS server (it does not need to be local). This local DNS server will conduct the entire DNS resolution process, and then send the result back to the computer. In this task, we will set up this local DNS server.

The root hint file. If the local DNS server cannot find the answer from its cache, it will go through an iterative process to get the answer from outside nameservers. The process starts from the root server. Therefore, the local DNS server needs to know the IP address of the root servers.

In `/etc/bind/named.conf.default-zones`, which is included in `BIND9`'s configuration file `/etc/bind/named.conf`, there is an entry for the root zone. This entry specifies a hint file for the root zone, and that is how the local DNS server knows the IP address of the root server.

```
zone "." {
   type hint;
   file "/usr/share/dns/root.hints";
};
```

The root hint file specifies the nameservers for the root zone (there are 13 nameservers for the root zone). The IP address (v4 and v6) for these nameservers are provided in this file. The following example shows a snippet of the file.

```
.                         3600000      NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.       3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.       3600000      AAAA  2001:503:ba3e::2:30
...
```

Inside our `local_dns_server` folder in the lab setup file, we have created an empty file called `root.hints`. This file will be copied into the `/usr/share/dns/` folder when we build the container image for the Local DNS server. You need to add records to this file, so your local DNS server can use your root server, instead of using the ones in the real world.

Testing. Using `docker-compose` commands to build and start all the containers. Once the containers are running, run the following "`dig @10.9.0.53`" commands from your VM to send DNS queries to the local DNS nameserver (its IP address is `10.9.0.53` in our setup). Please report your observations.

```
$ dig @10.9.0.53 www.example.edu
$ dig @10.9.0.53 www.example.com
$ dig @10.9.0.53 www.example.net
$ dig @10.9.0.53 www.syr.edu
$ dig @10.9.0.53 www.xyz.<TLD registered with your root server>
$ dig @10.9.0.53 www.xyz.<TLD not registered with your root server>
```

7 Task 6: Configure our VM to use this local DNS server

So far, we need to use `@<ip>` in our `dig` command to indicate what DNS server the `dig` command should talk to. While this is not an issue for `dig`, it is a problem for other software that depends on DNS. We need to tell the operating system that the DNS server container built in this task is the system's local DNS server.

This is achieved by changing the resolver configuration file (`/etc/resolv.conf`) of the user machine, so the container's IP address is added as the first `nameserver` entry in the file, i.e., this server will be used as the primary DNS server. Unfortunately, our provided VM uses the Dynamic Host Configuration Protocol (DHCP) to obtain network configuration parameters, such as IP address, local DNS server, etc. DHCP clients will overwrite `/etc/resolv.conf` with the information provided by the DHCP server.

One way to get our information into `/etc/resolv.conf` without worrying about the DHCP is to add the following entry to the `/etc/resolvconf/resolv.conf.d/head` file:

```
Add the following entry to /etc/resolvconf/resolv.conf.d/head
  nameserver 10.9.0.53

Run the following command for the change to take effect
$ sudo resolvconf -u
```

The content of the head file will be prepended to the dynamically generated resolver configuration file. Normally, this is just a comment line (the comment in `/etc/resolv.conf` comes from this head file).

After you finish configuring the user machine, repeat the testing commands in Task 5, but this time, do not include the `@<ip>` in the command. Pay attention to the highlighted IP address to check whether the response does come from your container. If the setup is unsuccessful, the IP will be different.

```
$ dig abc.example.edu

...
;; ANSWER SECTION:
abc.example.edu.  259200   IN A  1.2.3.7

;; Query time: 3 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
...
```

★★ Note (very very important) ★★. After finishing the lab, please do remember to remove the added entry from `/etc/resolvconf/resolv.conf.d/head`; otherwise, it will cause lot of problems for your future labs. If in the future, you have trouble connecting to the Internet, but your network is fine, the chances are that you may have forgotten to remove this entry, and it messes up the DNS query process. This happens to me many times, as I often forgot to remove the entry.

8 Task 7: Making It More Realistic

In this task, we will make our miniature DNS system a little bit more realistic by adding the following nameservers or records to our system.

- Adding 2 more root servers. The real-world DNS system has 13 different IP address for the root server. We will create 3 in this lab.
- Pick two of the TLDs registered with your root server, build a second-level domain for each of them, use your last name as the domain name. For example, if your last name is `Smith`, you need to build the nameservers for the `smith.<tld1>` and `smith.<tld2>` domains, You need to host both domains on the same nameserver.

9 Submission

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.

Cleanup. This is another reminder regarding the removing of the entry added to `/etc/resolvconf/resolv.conf.d/head`.