CIA Lab Assignment:
Domain Name System (DNS)

Infrastructure

For this lab and DNSSEC that will follow, half of you will use BIND and half will use Unbound+NSD.

If you have an even student number, you should use BIND, otherwise Unbound+NSD.
You will use your students PC for installing DNS servers.

We have an *sne21.ru* domain and the subdomain with name *stX.sne21ru* (X - is your student number) that will be delegated for your nameserver.

Task 1 - Downloading and Installing a Caching Name Server

The sources for the latest version:

- BIND - https://www.isc.org/bind/
- Unbound - https://nlnetlabs.nl/projects/unbound/about/
  NSD will be installed later.

1.1. - Validating the Download

The https://www.isc.org website provides signature files in addition to the BIND tarball. These can be used to check if you have downloaded the version they intended to distribute. The Unbound website uses a different mechanism, in particular, a modification detection code, better known as a cryptographic hash.

➢ Why is it wise to verify your download?

➢ Download the BIND tarball (also if you are doing the Unbound+NSD part) and check its validity using one of the signatures.

➢ Which mechanism is the best one to use (signatures or hashes)? Why?

1.2 - Documentation & Compiling

Apart from the source code, the distributions contain documentation about the servers and DNS.

- For BIND, the README file contains instructions on installation, and in the doc/ directory you can find the Administrator Reference Manual (ARM) and other relevant documents.
- For Unbound, the doc/ directory contains all information, including a README.

Most things about DNS are described and standardized in so-called “Request For Comments (RFCs)”, created and published by the Internet Engineering Task Force (IETF). A good DNS RFC to start with is RFC 1034: Domain Names - Concepts and Facilities.

Compiling and installing BIND, Unbound and NSD servers is simple and consists of the usual sequence of commands on most systems: *./configure; make and make install*
First, make sure your installation does not contain a previous version of the servers, as that can really mess things up (*show how to check*).
Next, configure, compile and then install the servers in the directory */usr/local/*.
Make sure each server will look for its configuration files in

- BIND - */usr/local/etc/bind*
- Unbound - */usr/local/etc/unbound*

Let the server write its state information, such as the *named.pid* or *unbound.pid* file, in */var/run.*
Information about which configuration options to use to achieve the result you can find in the README files.

➢ What is the difference between */etc*, */usr/etc*, */usr/local/etc*

Task 2 - Configuring Caching Name Server

Compiling and installing a server is relatively simple, but configuring a DNS server is not trivial. To keep things simple, we will start with BIND and Unbound running as a caching-only name server. This type of name server does not control any zone data.

➢ Why are caching-only name servers still useful?

The main configuration file you will have to create:

- BIND - *named.conf* and should be stored in */usr/local/etc/bind*
- Unbound - */usr/local/etc/unbound/unbound.conf*

It contains general options for the name server as well as references to other configuration files. Both BIND and Unbound provide sample files.

Root Servers

Our server needs a file containing references to the DNS root servers, the root hint file. The root hint file contains a list of root servers that our server uses to retrieve a more recent list of root servers. This file can be downloaded from *ftp://ftp.rs.internic.net/domain*.

Resolving localhost

The BIND server also needs a file in the working directory called *named.local* to resolve the loopback address 127.0.0.1 to the name localhost.
For Unbound, this is done automatically (see the local-zone feature).
This *named.local* file is in the standardized zone file format that you will need further on in the assignment for both scenarios. So take some time to study the example sample. The zone file format is defined in RFC 1033 which explains the <*origin*>, <*person*> and <*server*> fields.

Access control

During the configuration, think of scenarios when a caching nameserver should be used, and who can query your server.

- ➢ Why access control is important for recursive server?

Now that you know all the elements of the main configuration, create a simple *named.conf* or *unbound.conf* for a caching-only name server. Show the configuration file in your report.

Testing

You can check the syntax of your configuration file by using the *named-checkconf* and *unbound-checkconf* programs, respectively.
The *named-checkconf* program returns only a result value on success, *unbound-checkconf* prints a line and returns a result value.

- ➢ Why do the programs return a result value?

One way to see the result value is as follows:

```
if named-checkconf; then echo t; else echo f; fi
```

Task 3 - Running Caching Name Server

You can now start the DNS server by hand using *named -g -d2*, or *unbound -d -vv*, respectively (it will start the daemon with debug level 2 - read the manual page for more information). But it is better to use a "name server control" tool to start and stop the server. For BIND this tool is called *rndc*, and for Unbound, it is called *unbound-control*. You'll have to adapt the configuration to enable it. Also, it would be better to configure the name server to write debug information to a log file. This is described in Chapter 3 of the BIND ARM, or the unbound-control manual page, respectively.

➢ Show the changes you made to your configuration to allow remote control

➢ What other commands/functions does rndc/unbound-control provide? What

➢ What is the difference between stop -> start and reload?

To use your own server name you will need to adapt *resolv.conf*. Take into account that on recent distributions this file is automatically generated.

➢ What do you need to put in *resolv.conf* (and/or other files) to use your own name server?

Now use the tools and scripts provided with your distribution to test your name server (*Hint: dig, drill - study these tools to understand how to correctly query your server or f.e. google's or cloudflare's dns).*

Show that your queries are successfully resolved and cached by also inspecting the server's log file *(Hint: configure log verbosity level).*

Task 4 - Authoritative Name Server

If you are using Unbound+NSD, you should now install and test the NSD server before going to the next step. You can get it from https://nlnetlabs.nl/projects/nsd/about/. Follow the same steps as above for Unbound to configure, make and install it, also with the */usr/local* prefix.

Note: in order to run Unbound and NSD together you will need to decide how they should share the machine's network interfaces and ports. Again, think of use cases for authoritative and recursive name servers and make a decision.

Now that you have checked that your configuration works correctly, you can set it up to serve your own subdomain - *stX.sne21.ru*
But before, please answer:

➢ What is a private DNS zone? Is *stX.sne21.ru* private?

➢ What information was needed by TAs so they can implement the delegation?

The zone file format is standardized, so it is the same for both BIND and NSD.
The subdomain *stX.sne21.ru* is delegated to your experimentation server - *ns.stX.sne21.ru*.
Create a forward mapping zone file for your domain. It must contain the following resource records:

➢ 2 MX records. Make sure that mail for your domain is delivered to your own public IP. We will use the MX records later on.

➢ 4 A or AAAA records. Use your imagination.

➢ 2 CNAME records.

Then:

➢ Show the resulting zone file.

➢ Add a reference to your zone file to named.conf or nsd.conf, respectively.

➢ Restart or reload the name server and test your configuration using the provided tools *(Hint: dig, drill)*.

BONUS - Task 5 - Delegating Your Own Zone (2 points)

Work together with one of your fellow students. Each of you will create a subdomain of your own domain and delegate the authority for that subdomain to your partner. So in the end there should be two additional zones:

- `st<X>.st<Y>.sne21.ru` and
- `st<Y>.st<X>.sne21.ru`

where `X` - the number of your machine, `Y` - the number of your partner's machine.

How did you set up the subdomains and their delegation:

➢ How did you update `st<X>.sne21.ru` zone file to delegate subdomain to your partner?

➢ Show content of `st<X>.st<Y>.sne21.ru` zone file that you have delegation for.

➢ What named.conf/nsd.conf options did you add or change?

➢ Show the results of the tests that you performed.

BONUS - Task 6 - Zone Transfers (2 points)

Now that you have delegated a subdomain to your partner, you will set up your own server as a secondary for that domain. The primary and secondary servers must always contain the exact same zone data.

➢ How did you set up the secondary nameserver? Show the changes to the configuration files that you made.

➢ What happens if the primary name server for the subdomain fails?

The primary and secondary servers must always contain the exact same zone data. If the data on your primary server is updated, the secondary must also be updated. This is done via zone transfers. Set up the primary server for the domain that was delegated to you and the secondary name server on your partner's machine to allow zone transfers.

➢ Show the changes you had to make to your configuration

➢ Use a DNS tool (for instance *dig*) to initiate a zone transfer from your primary name server to your secondary.

➢ Show the output of the DNS tool and describe the steps in the transfer process.

➢ What information did the secondary server receive?