

DNS Security Extensions (DNSSEC) Lab

Copyright © 2022 by Wenliang Du.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. If you remix, transform, or build upon the material, this copyright notice must be left intact, or reproduced in a way that is reasonable to the medium in which the work is being re-published.

1 Lab Overview

To protect DNS, the Domain Name System Security Extensions (DNSSEC) were developed. DNSSEC is a set of extension to DNS, aiming to provide authentication and integrity checking on DNS data. With DNSSEC, all answers from DNSSEC protected zones are digitally signed. By checking the digital signature, a DNS resolver is able to check if the information is authentic or not. With such a mechanism, the DNS cache poisoning attack will be defeated, because any fake data, whether from a spoofed response packet or from an authoritative nameserver, will be detected because they will fail the signature checking.

The purpose of this lab is to help students understand how DNSSEC works. Students are given a simplified DNS infrastructure, and their job is configure each of the nameservers, so they all support DNSSEC. This lab covers the following topics:

- DNS and how it works
- DNSSEC and how it works
- Key management, public key, digital signature

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

- Chapter 11 of the SEED Book, *Internet Security: A Hands-on Approach*, 3rd 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 much on the SEED VM. You can do this lab using other VMs, physical machines, or VMs on the cloud.

2 Lab Environment Setup

In the lab environment, we provide a mini DNS infrastructure, consisting of a root server, a top-level domain server for `edu`, a domain server for `example.edu`, and a local DNS server. Each of these nameservers is hosted inside a container. For the sake of simplicity, we put all the containers on the same LAN (see Figure 1). This is a simplified setup, because in the real world, these nameservers are not on the same LAN. However, the configuration of these nameservers are similar to that in the real world, regardless of whether they are on the same LAN or scattered across the Internet.

If students have finished the DNS Infrastructure Lab before this one, they can choose to use the setup from the DNS Infrastructure Lab, instead of using the simplified setup in this lab, because that setup uses an Internet emulator as the basis and it is more realistic.

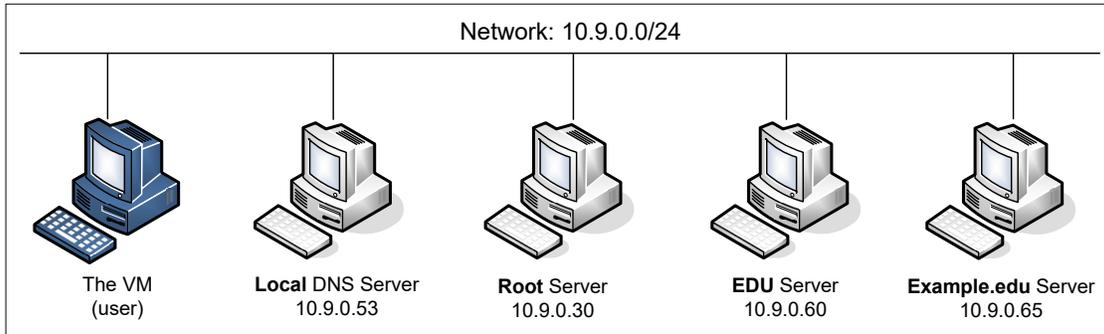


Figure 1: Lab environment setup

Software installation. We will use some tools that have not been installed in the SEED VM. Please use the following command to install the software:

```
$ sudo apt-get install bind9utils
```

The DNSSEC commands required for this lab are included in the package installed above. We only install them in the host VM, not inside the container. However, the results of the commands need to be copied into a corresponding container. We can use the "docker cp" command to copy files between the host and containers:

```
// Copy file xyz to container's /tmp folder
$ docker cp xyz <docker id>:/tmp

// Copy file /tmp/abc from container
$ docker cp <docker id>:/tmp/abc .
```

3 Task 1: Set Up the example.edu Domain

In this task, we will go to the container folder for the example.edu nameserver. This container hosts the example.edu domain. We will modify the files inside this folder, so the nameserver can support DNSSEC queries.

3.1 Task 1.a: Generate keys for example.edu server

First, we need to generate two pairs of keys, one is called Zone Signing Key (ZSK), and the other is called Key Signing Key (KSK). Each key consists of a public key and a private key. ZSK is used to sign the zone records, while the KSK is used to sign the ZSK.

This two-key scheme is quite common in key management. The one that actually signs the record (i.e., ZSK) can change frequently to achieve better security. If only one key is used, when this key changes, the data stored in the key management system needs to be updated. In DNSSEC, the parent zone stores the information about this key, so the parent zone file needs to be updated.

To avoid this overhead, the second key is introduced. This key is called Key Signing Key, i.e., it only signs the ZSK record in the zone file, while the other records are still signed by ZSK. The KSK's information is provided to the parent zone, but KSK does not change very frequently. When ZSK changes, all we need

to do is to use the KSK to generate a new signature for the ZSK.

Both ZSK and KSK are public key pairs, but since their roles and lifetime are different, typically the KSK key is set to be stronger than the ZSK key, i.e., the key size is longer. In the following, the first command generates a ZSK key using the RSASHA256 algorithm with the key size 1024. The second command generates a KSK key using the same algorithm, but doubling the key size. If your OS cannot find the command, you forgot to install the required software (see the lab setup section).

```
dnssec-keygen -a RSASHA256 -b 1024 example.edu
dnssec-keygen -a RSASHA256 -b 2048 -f KSK example.edu
```

The second command generates a KSK. The `-f` option put 257 in the key file, indicating this is the KSK. Without this option, a default value 256 is put in the key file, indicating this is the ZSK.

3.2 Task 1.b: Sign the `example.edu` domain's zone file

We will sign the zone file using the following `dnssec-signzone` command.

```
dnssec-signzone -e 20501231000000 -S -o example.edu example.edu.db
```

With the `-S` option, the command will look for the zone signing key and key signing key from the specified folder using the zone name (we did not specify the key folder using the `-K` option, so the default is the current folder). If you have multiple pairs of keys in the specified folder, the command will generate signatures using each key, so you would get multiple signatures for each record.

The `-e` option specifies the expiration time for the signatures. By default, the expiration time is only one month. Therefore, if you do not use the `-e` option, when you check your setup a month later, it will not work any more. For students, one month should be sufficient, but for instructors who might reuse their answers in the future, setting a long expiration time can avoid unnecessary trouble. The format of the expiration time is `YYYYMMDDHHMMSS`. In the example, we set the expiration time to year 2050.

Once the zone file is signed, a new zone file ended with `.signed` will be generated. Take a look at the file and describe what new entries are added to the original zone file. Please explain their purposes.

This new zone file will be used by our nameserver, so we will modify the `named.conf.seedlabs` file to tell the nameserver to use this file as its zone file. See the following example:

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

3.3 Task 1.c: Testing

Using the `docker-compose` command, we can build and start all the containers. We then use the following `dig` command to conduct the testing. The command allows us to directly ask a server (`@server`) to get a specific `type` of DNS record for a specific domain or host name. By using the `+dnssec` flag, we set the DNSSEC OK bit (DO) in the OPT record in the additional section of the query, requesting the server to send back the related DNSSEC records, such as the signature.

```
General format:
$ dig @server name type +dnssec
```

```
Example:
$ dig @10.9.0.65 example.edu DNSKEY +dnssec
```

Run the following commands to get different types of records from the `example.edu` nameserver. Provide an explanation for each of the records in the response. In our setup, `10.9.0.65` is the IP address assigned to the `example.edu` nameserver.

```
$ dig @10.9.0.65 example.edu DNSKEY +dnssec
$ dig @10.9.0.65 example.edu NS +dnssec
$ dig @10.9.0.65 www.example.edu A +dnssec
```

4 Task 2: Set Up the edu Server

In this task, we will set up the `edu` server. The instructions are similar to those in Task 1.

4.1 Task 2.a: Find and add the DS record

After signing the zone file of `example.edu`, a DS record is generated for the Key Signing Key. By default, it is placed inside a file called `dsset-<xyz>`, where `<xyz>` is the zone name. In our case, `dsset-example.edu.` is the file name.

DS (Delegation Signer) record holds the name of a delegated zone. It references a DNSKEY record in the sub-delegated zone. The DS record should be placed in the parent zone along with the delegating NS records. The following gives an example of the DS record:

```
example.edu.    IN DS 10246      8  2    563D... (omitted) ...1D59D1
                ①                               ②
```

A DS record contains a key tag (marked as ①), which is a short numeric value identifying the referenced key. This key is the Key Signing Key (KSK) for the domain. The most important element in a DS record is the digest (marked by ②), which is the one-way hash value of the KSK. By putting this digest value in the parent zone (i.e., the `edu` zone), the integrity of the sub-delegated zone's KSK can be verified. That is the main purpose of this DS record.

4.2 Task 2.b: Set up the edu server

Follow the instruction in Task 1 to generate the ZSK and KSK keys for this domain, and then use the keys to sign the zone file. Before signing the zone file, we need to add an entry to it. The entry is the DS record of the sub-delegated zone. In our case, the `example.edu` zone is delegated to another nameserver, so the digest of its Key Signing Key must be included in the `edu` zone; this way, the integrity of the key can be verified.

We can copy and paste the content of the DS record to `edu`'s zone file, or use the following `INCLUDE` entry to include the file. The latter method is more convenient for the lab, because even if we change the Key Signing Key of a zone, the DS record's file name stays the same, so there is no need to change the parent's zone file.

```
$INCLUDE ../edu.example/dsset-example.edu.
```

4.3 Task 2.c: Testing

Build and run all the containers. Then run the following commands to get different types of records from the `edu` nameserver. Provide an explanation for each of the records in the response. In our setup, `10.9.0.60` is the IP address assigned to the `edu` nameserver.

```
$ dig @10.9.0.60 edu DNSKEY +dnssec
$ dig @10.9.0.60 edu NS +dnssec
$ dig @10.9.0.60 example.edu +dnssec
```

5 Task 3: Set Up the Root Server

The procedure to set up the root server is the same as that in the previous task. Remember to add the `edu` zone's DS record to the zone file, before signing the zone.

Testing. Run the following commands to get different types of records from the root nameserver. Provide an explanation for each of the records in the response. In our setup, `10.9.0.30` is the IP address assigned to the root nameserver.

```
$ dig @10.9.0.30 . DNSKEY +dnssec
$ dig @10.9.0.30 . NS +dnssec
$ dig @10.9.0.30 edu +dnssec
$ dig @10.9.0.30 example.edu +dnssec
```

6 Task 4: Set Up 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 DNSSEC, each nameserver provides its own public keys (both ZSK and KSK) to the client, so the client can use KSK to verify the signature on ZSK, and then use ZSK to verify the signature on each record. That leaves one thing uncertain: how to verify the authenticity of KSK? That is the purpose of the DS record in the parent zone. The parent will provide the DS record for its sub-zone, and this DS record is used to verify the KSK of sub-zone.

Since the root server does not have a parent zone, how do we verify the authenticity of the root server? The root servers' public keys are the root of the trust, so they are the most important keys in the DNSSEC infrastructure, and they are called *trust anchors*. They must be obtained using a secure way.

BIND 9 has built-in DNSSEC trust anchors, but they can be overridden by the content inside `/etc/bind/bind.keys`. In this task, we will put our root server's public key into this file. Make sure to use KSK here, not ZSK.

```
trust-anchors {
    . initial-key 257 3 8 " <Root's Key Signing Key> ";
};
```

We also need to enable DNSSEC on the local DNS server, so it will conduct DNSSEC validation. We just need to modify the `named.conf.options` file for the local DNS server container, changing the DNSSEC entries to the following:

```
dnssec-validation auto;
dnssec-enable yes;
```

Testing. After starting the local server container, run the following command (10.9.0.53 is the IP address assigned to the local DNS server in our setup).

```
$ dig @10.9.0.53 www.example.edu +dnssec
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37130
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ...

;; ANSWER SECTION:
www.example.edu.      259200  IN      A       1.2.3.5
www.example.edu.      259200  IN      RRSIG   ... <signature>
```

If everything has been set up correctly, you should be able to get the answer, the IP address of `www.example.edu` if the reply is authentic. Pay attention to the `ad` flag, which means *authentic data*. If this flag is set, all the records in the reply are authentic, i.e., the DNSSEC validation is successful. If the DNSSEC is not enabled, you will still get answers, but this flag will not be set.

If the reply is not authentic, you will get an error message. Please design an experiment to demonstrate that when the reply is fake, the local DNS server will be able to detect it and show an error message. You do not need to conduct a real attack in this experiment. You can directly modify some of the existing records without regenerating a signature (you can also corrupt a signature).

7 Task 5: Adding Another Domain Nameserver

Assume that you own a domain name inside `edu`, and you want to set up a DNSSEC-enabled nameserver for this domain. We have already created a spare container for this purpose (the BIND9 server is already installed on this container). Your task is to configure the nameserver, so it supports DNSSEC. The name of the domain should contain your last name and the year when this lab is conducted. For example, if your last name is Smith and the year is 2022, the domain name should be `smith2022.edu`. You do not need to actually own this domain in the real world. We only set it up in the DNS infrastructure within our lab environment.

8 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.