DNSSEC validation

During this practice we are only going to access the following equipment:

- **grpX-cli** : client
- **grpX-resolv1** & **grpX-resolv2** : recursive servers (resolvers)

Setting up a BIND validating recursive server.

We use the container "Resolv 1" (recursive server) [**grpX-resolv1**].

This container already has the BIND9 packages downloaded and installed.

We switch to the root user:

```
$ sudo su -
```

We go to the /etc/bind directory:

```
# cd /etc/bind
```

At this point we must configure some BIND9 options.
To do this we edit the file `/etc/bind/named.conf.options` :

```
# nano named.conf.options
```

Now we add the options to indicate (when resolving) which are the IP addresses that will be able to send DNS queries and at the same time to which IP addresses our server will listen on port 53 (in this case both prefixes are identical). The file should be as follows:

```
options {
    directory "/var/cache/bind";

    // If there is a firewall between you and nameservers you want
    // to talk to, you may need to fix the firewall to allow multiple
    // ports to talk. See http://www.kb.cert.org/vuls/id/800113

    // If your ISP provided one or more IP addresses for stable
    // nameservers, you probably want to use them as forwarders.
    // Uncomment the following block, and insert the addresses replacing
    // the all-0's placeholder.

    // forwarders {
    //  0.0.0.0;
    // };

    //==================================================================
    // If BIND logs error messages about the root key being expired,
    // you will need to update your keys. See https://www.isc.org/bind-k
    //==================================================================
    dnssec-validation auto;

    listen-on port 53 { localhost; 100.100.0.0/16; };
    listen-on-v6 port 53 { localhost; fd89:59e0::/32; };

    allow-query { localhost; 100.100.0.0/16; fd89:59e0::/32; };     <---

    recursion yes;
};
```

Once we finish editing the configuration file, we execute a command that allows us to quickly check if the configuration is semantically correct (if the command does not return anything, it means that it did not find errors in the configuration files):

```
# named-checkconf
```

Finally we restart the server so that it takes the configuration changes:

```
# systemctl restart bind9
```

And we check the status of the bind9 process:

```
# systemctl status bind9
```

We should obtain an output similar to the following:

```
● named.service - BIND Domain Name Server
    Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor pre
   Drop-In: /etc/systemd/system/service.d
        └─lxc.conf
    Active: **active (running)** since Thu 2021-05-13 01:38:27 UTC; 4s ago
      Docs: man:named(8)
  Main PID: 849 (named)
     Tasks: 50 (limit: 152822)
    Memory: 103.2M
    CGroup: /system.slice/named.service
        └─849 /usr/sbin/named -f -u bind

May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: *
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: ma
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: zo
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: zo
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: zo
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: zo
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: *
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: *
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: ma
May 13 01:38:27 resolv1.grpX.<lab_domain>.te-labs.training named[849]: re
```

Test your new validating resolver

Run the following commands and confirm if you receive the "ad" flag:

1. dig SOA com. @100.100.X.67
2. dig SOA com. @100.100.X.67 +dnssec
3. dig A www.icann.org @100.100.X.67
4. dig NS icann.org @100.100.X.67
5. dig DNSKEY *grpX.<lab_domain>*.te-labs.training @100.100.X.67
6. dig NS *grpX.<lab_domain>*.te-labs.training @100.100.X.67 +dnssec
7. dig SOA *grpX.<lab_domain>*.te-labs.training @100.100.X.67
8. dig DNSKEY *grpX.<lab_domain>*.te-labs.training @100.100.X.130
9. dig DNSKEY *grpX.<lab_domain>*.te-labs.training @100.100.X.130 +multi
10. dig SOA *grpX.<lab_domain>*.te-labs.training @100.100.X.131 +dnssec +multi

Did you receive the "ad" flag for the last three dig queries ? Why ?

Now try the below:

1. dig www.dnssec-failed.org @9.9.9.9
2. dig www.dnssec-failed.org @100.100.X.67
3. dig www.dnssec-failed.org +dnssec @100.100.X.67

4. dig www.dnssec-failed.org +dnssec +cd @100.100.X.67

What do you notice ? Why ?

Set your recursive resolver OS to use your local validating recursive resolver sofltware.

Still in **grpX-resolv1** server, edit the **/etc/resolv.conf** config file and replace whatever is there by the following only:

```
nameserver 100.100.X.67
```

Save and exit. Then try the following queries:

1. dig SOA com.
2. dig SOA com. +dnssec
3. dig A www.icann.org
4. dig NS icann.org
5. dig NS *grpX.<lab_domain>*.te-labs.training
6. dig NS *grpX.<lab_domain>*.te-labs.training +dnssec
7. dig SOA *grpX.<lab_domain>*.te-labs.training
8. dig SOA *grpX.<lab_domain>*.te-labs.training +dnssec +multi
9. dig DNSKEY *grpX.<lab_domain>*.te-labs.training
10. dig DNSKEY *grpX.<lab_domain>*.te-labs.training +multi

Did you get the "ad" flag in all the cases ?

Setting up an Unbound recursive server

We use the container "Resolv 2" (recursive server) [**grpX-resolv2**].

This container already has the UNBOUND packages downloaded and installed.

At this point we must configure some UNBOUND options.
To do this we switch to the root user and edit **/etc/unbound/unbound.conf** config file:

```
$ sudo su -
# nano /etc/unbound/unbound.conf
```

Now we add the options to indicate (when resolving) which are the interfaces on which it will listen for queries, the IP addresses that will be able to send it DNS queries, the port it will use (53), and some other parameters. The file should be as follows:

```
# Unbound configuration file for Debian.
#
# See the unbound.conf(5) man page.
#
# See /usr/share/doc/unbound/examples/unbound.conf for a commented
# reference config file.
#
# The following line includes additional configuration files from the
# /etc/unbound/unbound.conf.d directory.

server:
        interface: 0.0.0.0
        interface: ::0

        access-control: 127.0.0.0/8 allow
        access-control: 100.100.0.0/16 allow
        access-control: fd89:59e0::/32 allow

        port: 53

        do-udp: yes
        do-tcp: yes
        do-ip4: yes
        do-ip6: yes

include: "/etc/unbound/unbound.conf.d/*.conf"
```

Once we finish editing the configuration file, we execute a command that allows us to quickly check if the configuration is semantically correct:

```
# unbound-checkconf
```

If it is correct, it will return something similar to the following:

```
unbound-checkconf: no errors in /etc/unbound/unbound.conf
```

Finally we restart the server so that it takes the configuration changes:

```
# systemctl restart unbound
```

And we check the status of the UNBOUND process:

```
# systemctl status unbound
```

We should obtain an output similar to the following:

```
● unbound.service - Unbound DNS server
Loaded: loaded (/lib/systemd/system/unbound.service; enabled; vendor pre
Drop-In: /etc/systemd/system/service.d
     └─lxc.conf     Active: active (running) since Thu 2021-05-13 03:49:1
    Process: 571 ExecStartPre=/usr/lib/unbound/package-helper chroot_set
    Process: 574 ExecStartPre=/usr/lib/unbound/package-helper root_trust_
    CGroup: /system.slice/unbound.service                   └─578 /usr/s
May 13 03:49:10 resolv2.grpX.<lab_domain>.te-labs.training unbound[178]:
May 13 03:49:10 resolv2.grpX.<lab_domain>.te-labs.training unbound[178]:
May 13 03:49:10 resolv2.grpX.<lab_domain>.te-labs.training unbound[178]:
May 13 03:49:11 resolv2.grpX.<lab_domain>.te-labs.training package-helpe
May 13 03:49:11 resolv2.grpX.<lab_domain>.te-labs.training package-helpe
May 13 03:49:11 resolv2.grpX.<lab_domain>.te-labs.training unbound[578]:
May 13 03:49:11 resolv2.grpX.<lab_domain>.te-labs.training unbound[578]:
May 13 03:49:11 resolv2.grpX.<lab_domain>.te-labs.training unbound[578]:
May 13 03:49:11 resolv2.grpX.<lab_domain>.te-labs.training unbound[578]:
May 13 03:49:11 resolv2.grpX.<lab_domain>.te-labs.training systemd[1]: S
```

Test your new validating resolver

Run the following commands and confirm if you receive the "ad" flag:

1. dig SOA com. @100.100.X.68
2. dig SOA com. @100.100.X.68 +dnssec
3. dig A www.icann.org @100.100.X.68
4. dig NS icann.org @100.100.X.68
5. dig DNSKEY *grpX.<lab_domain>*.te-labs.training @100.100.X.68
6. dig NS *grpX.<lab_domain>*.te-labs.training @100.100.X.68 +dnssec
7. dig SOA *grpX.<lab_domain>*.te-labs.training @100.100.X.68
8. dig DNSKEY *grpX.<lab_domain>*.te-labs.training @100.100.X.130
9. dig DNSKEY *grpX.<lab_domain>*.te-labs.training @100.100.X.130 +multi
10. dig SOA *grpX.<lab_domain>*.te-labs.training @100.100.X.131 +dnssec +multi

Did you receive the "ad" flag for the last three dig queries ? Why ?

Now try the below:

1. dig www.dnssec-failed.org @9.9.9.9
2. dig www.dnssec-failed.org @100.100.X.68
3. dig www.dnssec-failed.org +dnssec @100.100.X.68
4. dig www.dnssec-failed.org +dnssec +cd @100.100.X.68

What do you notice ? Why ?

Set your recursive resolver OS to use your local validating

recursive resolver software.

Still in **grpX-resolv2** server, edit the **/etc/resolv.conf** config file and replace whatever is there by the following only:

```
nameserver 100.100.X.68
```

Save and exit. Then try the following queries:

1. dig SOA com.
2. dig SOA com. +dnssec
3. dig A www.icann.org
4. dig NS icann.org
5. dig NS *grpX*.<*lab_domain*>.te-labs.training
6. dig NS *grpX*.<*lab_domain*>.te-labs.training +dnssec
7. dig SOA *grpX*.<*lab_domain*>.te-labs.training
8. dig SOA *grpX*.<*lab_domain*>.te-labs.training +dnssec +multi
9. dig DNSKEY *grpX*.<*lab_domain*>.te-labs.training
10. dig DNSKEY *grpX*.<*lab_domain*>.te-labs.training +multi

Did you get the "ad" flag in all the cases ?

Temporary DNSSEC validation deactivation for broken domains

A zone can become broken due to issues with its DNSSEC configuration. While it is the responsibility of the domain administrator to do their best to avoid such things, it can still happen. And when it happens, users behind a validating recursive resolver usually do not have access to the domain data.

We can configure the recursive resolver to temporarily skip DNSSEC validation for this category of broken domains only. This can also serve when you have internal domains on which DNSSEC validation is not really necessary as those domains are trusted by default.

BIND

We can edit the file **/etc/bind/named.conf.options** and add the following:

```
validate-except
   {
       "some_domain_1_to_exclude.TLD";
       "another.example.TLD";
   };
```

Apply the config and test.

Unbound

To temporarily stop a broken chain of trust, add the following to the **unbound.conf** file:

```
server:
    domain-insecure: "example.TLD"
```

Test it !

You can *dig* to confirm if validation is disabled for your excluded domain only.