TP4 – Systems Security
- DNSSEC, DoT, DoH -

IFRIM Vasile-Alexandru
M2 SSI

The objective of this TP is to secure a Bind9 domain name server by using 3 methods: DNSSEC, DNS over TLS (DoT), and DNS over HTTPS (DoH). We will have to

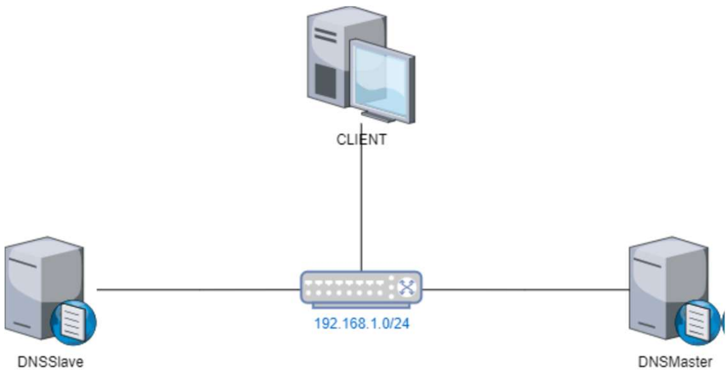
- set up a small infrastructure for which we will use LXD containers,
- configure a Bind9 server and DNSSEC on it,
- configure DoT to secure DNS requests,
- configure DoH to secure DNS requests via HTTPS,
- secure the transfer zones between the master server and a slave server using TSIG,
- and finally, test and validate each configuration.

By default, DNS queries and responses are sent in plaintext (via UDP), for anyone to see. **DNSSEC** is a set of security extensions for verifying the identity of DNS root servers and authoritative nameservers in communications with DNS resolvers. It is designed to prevent DNS cache poisoning, MITM, and other attacks. It **does not encrypt** communications.

DNS over TLS, or **DoT**, is a standard for **encrypting DNS** queries to keep them secure and private. DoT uses the same security protocol, TLS, that HTTPS websites use to encrypt and authenticate communications; it adds the TLS encryption on top of the UDP and uses **port 853**.

DNS over HTTPS, or **DoH**, is an alternative to DoT. With DoH, DNS queries and responses are encrypted, but they are sent via the HTTP or HTTP/2 protocols instead of directly over UDP; because it uses the HTTPS protocol**, port 443** is used.

Through either of these 2 methods, requests and responses cannot be tampered with or forged via on-path attacks.



To create this infrastructure, we will first set up an LXD bridge network:

```
$ lxc network create dnsnet ipv4.address=192.168.1.1/24
  ipv6.address=none ipv4.nat=false ipv4.dhcp.ranges=192.168.1.2-
  192.168.1.254
```

where

- `ipv4.address=192.168.1.1/24` defines the IP range for the subnet
- `ipv6.address=none` disables IPv6.
- `ipv4.nat=false` disables Network Address Translation to ensure routing is done directly between the containers. The DNSMaster and DNSSlave servers must interact directly, as DNSSEC and zone transfer mechanisms require unaltered source and destination IPs for authentication. Clients querying the servers should also reach them directly using their private network addresses.

- `ipv4.dhcp.ranges` specifies the range of automatically served addresses. For the 2 servers, we will be assigning static addresses.

We can check the network config by

```
$ lxc network show dnsnet
```

```
root Jan 18 18:53 ~ > lxc network show dnsnet
name: dnsnet
description: ""
type: bridge
managed: true
status: Created
config:
  ipv4.address: 192.168.1.1/24
  ipv4.dhcp.ranges: 192.168.1.2-192.168.1.254
  ipv4.nat: "false"
  ipv6.address: none
used_by: []
locations:
- none
```

Next, we launch 3 containers using an alpine image:

```
$ lxc launch images:alpine/3.21 Client -n dnsnet
$ lxc launch images:alpine/3.21 DNSMaster -n dnsnet
$ lxc launch images:alpine/3.21 DNSSlave -n dnsnet
```

```
root Jan 18 19:02 ~ > lxc ls
+-----------+---------+----------------------+------+-----------+-----------+
|   NAME    |  STATE  |         IPV4         | IPV6 |   TYPE    | SNAPSHOTS |
+-----------+---------+----------------------+------+-----------+-----------+
| Client    | RUNNING | 192.168.1.120 (eth0) |      | CONTAINER | 0         |
+-----------+---------+----------------------+------+-----------+-----------+
| DNSMaster | RUNNING | 192.168.1.244 (eth0) |      | CONTAINER | 0         |
+-----------+---------+----------------------+------+-----------+-----------+
| DNSSlave  | RUNNING | 192.168.1.141 (eth0) |      | CONTAINER | 0         |
+-----------+---------+----------------------+------+-----------+-----------+
```

For the servers, we are going to set static IP addresses within the given subnet, going into each one and modifying the `/etc/network/interfaces` file.

- DNSMaster will have 192.168.1.10
- DNSSlave will have 192.168.1.11

```
auto eth0
iface eth0 inet static
        address 192.168.1.10
        netmask 255.255.255.0
        gateway 192.168.1.1
hostname $(hostname)
```

Sample /etc/network/interfaces of DNSMaster

After doing a `service networking restart` on both Alpine containers, we check again the state of our infrastructure:

```
root Jan 18 19:05 ~ > lxc ls
+-----------+---------+----------------------+------+-----------+-----------+
|   NAME    |  STATE  |         IPV4         | IPV6 |   TYPE    | SNAPSHOTS |
+-----------+---------+----------------------+------+-----------+-----------+
| Client    | RUNNING | 192.168.1.120 (eth0) |      | CONTAINER | 0         |
+-----------+---------+----------------------+------+-----------+-----------+
| DNSMaster | RUNNING | 192.168.1.10 (eth0)  |      | CONTAINER | 0         |
+-----------+---------+----------------------+------+-----------+-----------+
| DNSSlave  | RUNNING | 192.168.1.11 (eth0)  |      | CONTAINER | 0         |
+-----------+---------+----------------------+------+-----------+-----------+
```

After doing a ping between the machines, we should see they are able to communicate. For them to be able to access the internet, we have to make sure IPv4 forwarding is enabled on the host machine and that the host machine has NAT enabled – for this we check the `iptables` and, if necessary, add the following rule:

```
$ sudo iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth1 -j
  MASQUERADE
```

so that the rule is applied to packets after they are routed but before they leave the network interface, replacing the source IP address of outgoing packets with the external IP of the host.

Finally, we update the packages and install the following tools on both servers:

```
$ lxc exec <dns_container> -- apk add bind bind-tools bind-dnssec-tools
  curl
```

Exercise 1

We start off by creating the 2 zones our DNS server using BIND9 will manage:

- A forward zone, `ssi.edu`, that handles queries for domain names like `ns1.ssi.edu`
- A reverse zone, `1.168.192.in-addr.arpa`, that maps IP addresses back to hostnames

On DNSMaster, we create a directory `/etc/bind` where we will put a file, `named.conf`, with the following configuration that enables DNSSEC and sets the 2 zones:

```
options {
        directory "/var/bind";
        dnssec-validation auto;
        listen-on { 192.168.1.10; };
};
```

```
# forward
zone "ssi.edu" {
        type master;
        file "/etc/bind/db.ssi.edu";
        allow-transfer { 192.168.1.11; }; # allow DNSSlave
};
```

```
# reverse
zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.1.168.192";
        allow-transfer { 192.168.1.11; }; # allow DNSSlave
};
```

Next, we create the zone files `/etc/bind/db.ssi.edu` and `/etc/bind/db.1.168.192` where DNS records are defined. The zone files are the heart of the DNS system, providing the actual data (records) that clients query.

- `SOA` (Start of Authority) identifies the authoritative server for the zone, `ns1.ssi.edu`, who refers to DNSMaster (192.168.1.10), followed by the email address of the person responsible (in DNS, the @ in an email is replaced by a dot – `admin.ssi.edu` becomes `admin@ssi.edu`); a serial number, YYYYMMDDNN (where NN – revision number), refresh,retry and expire interval for secondary (slave) servers
- `NS` (Nameserver) identifies the server handling DNS queries for the domain
- `A` (Address) and PTR (Pointer) map domain names to IPs and vice versa

Here we also define the `$TTL` that indicates, in seconds, how long a DNS record should be cached by resolvers (clients) before being discarded or revalidated; we set it to 24h.

```
$TTL 86400
@       IN      SOA     ns1.ssi.edu. admin.ssi.edu. (
                          2025011801 ; Serial
                          3600       ; Refresh
                          1800       ; Retry
                          604800     ; Expire
                          86400 )    ; Minimum TTL
;
@       IN      NS      ns1.ssi.edu.
ns1     IN      A       192.168.1.10
```

/etc/bind/db.ssi.edu, the forward zone

```
$TTL 86400
@       IN      SOA     ns1.ssi.edu. admin.ssi.edu. (
                          2025011801 ; Serial
                          3600       ; Refresh
                          1800       ; Retry
                          604800     ; Expire
                          86400 )    ; Minimum TTL
;
@       IN      NS      ns1.ssi.edu.
10      IN      PTR     ns1.ssi.edu.
```

/etc/bind/db.1.168.192, the reverse zone

We use named-checkzone to validate the syntax of the zone files:

```
DNSMaster:~# named-checkzone ssi.edu /etc/bind/db.ssi.edu
zone ssi.edu/IN: loaded serial 2025011801
OK
DNSMaster:~# named-checkzone 1.168.192.in-addr.arpa /etc/bind/db.192.168.1
zone 1.168.192.in-addr.arpa/IN: loaded serial 2025011801
OK
DNSMaster:~#
```

If everything is valid, we can start the service via:

```
$ service named start
```

```
options {
        directory "/var/bind";
        dnssec-validation yes;
        allow-query { any; };
        allow-transfer { none; };
};
```

DNSSlave named.conf options block

To be able to transfer queries to DNSSlave, we must configure on DNSMaster a TSIG key:

```
$ tsig-keygen > /etc/bind/tsig.key
```

```
DNSMaster:/etc/bind# cat tsig.key
key "tsig-key" {
        algorithm hmac-sha256;
        secret "TDdcWYLNAR6E+dcQoqVDWzccMZYm1fmgSu4kXfQu/Zw=";
};
DNSMaster:/etc/bind# vim named.conf
```

This configuration contained in the `tsig.key` must then be added to `named.conf` on both servers. The transfer of this file/configuration must be done in a confidential, authenticated manner. On DNSMaster, we don't allow any more transfer to just 192.168.1.11, but to anyone that provides DNSMaster proof, meaning the key's **secret**.

```
# forward
zone "ssi.edu" {
        type master;
        file "/etc/bind/db.ssi.edu";
        # allow transfer IF authenticated with TSIG key
        allow-transfer { key "tsig-key"; };
        allow-update { none; };
        also-notify { 192.168.1.11; };
};

# reverse
zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.1.168.192";
        # allow transfer IF authenticated with TSIG key
        allow-transfer { key "tsig-key"; };
        allow-update { none; };
        also-notify {192.168.1.11; };
};
```

DNSMaster

```
zone "ssi.edu" {
        type slave;
        masters {
                192.168.1.10 key "tsig-key";
        };
        file "/var/bind/db.ssi.edu";
        allow-notify { 192.168.1.10; };
};

zone "1.168.192.in-addr.arpa" {
        type slave;
        file "/var/bind/db.1.168.192";
        masters {
                192.168.1.10 key "tsig-key";
        };
        allow-notify { 192.168.1.10; };
};
```

DNSSlave

```
DNSSlave:/etc/bind# ls /var/bind/ -l
total 18
-rw-r--r--    1 named    named          815 Jan 19 15:26 db.1.168.192
-rw-r--r--    1 named    named          747 Jan 19 15:26 db.ssi.edu
drwxrwx---    2 root     named            2 Jan 18 22:08 dyn
-rw-r--r--    1 named    named          221 Jan 19 13:22 managed-keys.bind
-rw-r--r--    1 named    named         2438 Jan 19 13:21 managed-keys.bind.jnl
lrwxrwxrwx    1 root     root            41 Jan 18 22:08 named.ca -> ../../usr/share/dns-root-hin
ts/named.root
drwxr-x---    2 root     named            4 Jan 18 22:08 pri
lrwxrwxrwx    1 root     root             8 Jan 18 22:08 root.cache -> named.ca
drwxrwx---    2 root     named            2 Jan 18 22:08 sec
DNSSlave:/etc/bind#
```

We see that DNSSlave manages to cache

The key block specifies the TSIG key with the algorithm hmac-sha256 used for signing and the secret, base64-encoded, generated on DNSMaster. On DNSSlave, the zone blocks are of type "slave", they specify the master server's IP and the local file where zone data will be **cached**. DNSSlave replicates the reverse zone too, for redundancy/performance, but it is not necessary.

To test the zone transfer, we can issue the following command from either DNSSlave or the Client:

```
$ dig @192.168.1.10 ssi.edu AXFR -y <hmac_algorithm>:<key_name>:<secret>
```

```
Client:~# dig @192.168.1.10 ssi.edu AXFR -y hmac-sha256:tsig-key:TDdcWYLNAR6E+dcQoqVDWzccMZYm1fmg
Su4kXfQu/Zw=

; <<>> DiG 9.18.32 <<>> @192.168.1.10 ssi.edu AXFR -y hmac-sha256:tsig-key:TDdcWYLNAR6E+dcQoqVDWz
ccMZYm1fmgSu4kXfQu/Zw=
; (1 server found)
;; global options: +cmd
ssi.edu.                86400   IN      SOA     ns1.ssi.edu. admin.ssi.edu. 2025011801 3600 1800
604800 86400
ssi.edu.                86400   IN      NS      ns1.ssi.edu.
ssi.edu.                86400   IN      DNSKEY  256 3 8 AwEAAbh972llSShHb6o34zWrlqTM/qT1x564DL6vN
Xct99QTpIe3t9aS siSAazYCZPNQ0KOd+iHtynyUjaOpj3rwSnnxghS+DWfsquYNSxja2AGz KVU/IbIr+j+zBnB58w80qRsr
nCkmypKp8NDonp7zUXsJK6YCvoB8TFSh 6vLjU+4j/eQCNmegICfgSj8KlSfFUq9xMqDKoRjeXCrHhFEFYYiL4qyp wE6sevO
VJkNSStON88tVx0/2Ld94TIc2c+zpMa05wvwPKE15tnUPbHwc Suu0A7Xx7+MkOaPLDxSr6BvSHAwN/16fGxHExzdIJAbP9AQ
vr/SYPUQC Hk7bwUMC2jk=
ssi.edu.                86400   IN      DNSKEY  257 3 8 AwEAAdp6h2PK42dXBsXhfTAp8ct66Bx6XVO/8XC4S
jjn8hRcMVtb4Bh+ ABfP438bAzCEaCf7Rw5SCF2WoGruwZxVHY61TdjJ9uDPUbHLNU+IW4NE AuTNnmuwJYkUhZVlVnJgamNy
G5bPlzbMnuxn10HmsTx5lsnBnS7q2xt3 gNidI1+EudlzuZporFDmuIf2J7K8udhcow3q6pYubtyHAZQjMzPSxfLt rlLxnIi
IF2nsv4hBvcQ7TZzYZI5+p2mlknb7he9q3suu+idjqOq1vK9s u1cONs1nDn/MNBmxgjLuCqloWR+6G/8ndPuAqVRbfC45lWX
```

If we try without providing the key, even from the slave:

```
DNSSlave:/etc/bind# dig @192.168.1.10 ssi.edu AXFR

; <<>> DiG 9.18.32 <<>> @192.168.1.10 ssi.edu AXFR
; (1 server found)
;; global options: +cmd
; Transfer failed.
DNSSlave:/etc/bind#
```

Now, we can generate on DNSMaster a zone signing key (ZSK) and a key signing key (KSK) for each zone. The ZSK signs individual DNS records within the zone, while the KSK signs the ZSK to establish a chain of trust. Considering the latest recommendations of NIST for RSA, the minimum key size should actually be 3072 bits. KSK keys should be changed annually, while for ZSK it’s recommended to do it at least quarterly.

```
$ mkdir -p /etc/bind/keys/ssi.edu
$ cd /etc/bind/keys/ssi.edu
$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -n ZONE ssi.edu
$ dnssec-keygen -a RSASHA256 -b 2048 -n ZONE ssi.edu
```

```
$ mkdir -p /etc/bind/keys/1.168.192.in-addr.arpa
$ cd /etc/bind/keys/1.168.192.in-addr.arpa
$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -n ZONE 1.168.192.in-
  addr.arpa
$ dnssec-keygen -a RSASHA256 -b 2048 -n ZONE 1.168.192.in-addr.arpa
```

Note: `-S` option is for smart signing: instructs dnssec-signzone to search the key repository for keys that match the zone being signed, and to include them in the zone if appropriate.

The generated keys can now be appended to the zone files. Note that we might be undermining the principle of separation of trust by reusing the `ssi.edu` keys for the reverse zone.

```
$ cd /etc/bind/keys/ssi.edu/
$ cat Kssi.edu.+*.key >> /etc/bind/db.ssi.edu
$ dnssec-signzone -A -3 abcdabcd -N INCREMENT -o ssi.edu -t
  /etc/bind/db.ssi.edu
```

```
$ cd /etc/bind/keys/168.192.in-addr.arpa
$ cat K1.168.192.in-addr.arpa.+*.key >> /etc/bind/db.1.168.192
$ dnssec-signzone -A -3 abcdabcd -N INCREMENT -o 1.168.192.in-addr.arpa
  -t /etc/bind/db.1.168.192
```

NSEC & NSEC3 solve the problem of proving non-existence, explained in-depth by the DNS Institute [dnsinstitute.com/documentation/dnssec-guide/ch06s02.html]. It is typically used in preventing attackers from enumerating records.

```
DNSMaster:/etc/bind/keys/ssi.edu# dnssec-signzone -A -3 abcdabcd -N INCREMENT -o ssi.edu -t /etc/
bind/db.ssi.edu
Verifying the zone using the following algorithms:
- RSASHA256
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                      ZSKs: 1 active, 0 stand-by, 0 revoked
/etc/bind/db.ssi.edu.signed
Signatures generated:                        8
Signatures retained:                         0
Signatures dropped:                          0
Signatures successfully verified:            0
Signatures unsuccessfully verified:          0
Signing time in seconds:                 0.030
Signatures per second:                 266.666
Runtime in seconds:                      0.045
```

```
DNSMaster:/etc/bind/keys/1.168.192.in-addr.arpa# dnssec-signzone -A -3 abcdabcd -N INCREMENT -o 1
.168.192.in-addr.arpa -t /etc/bind/db.1.168.192
Verifying the zone using the following algorithms:
- RSASHA256
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                      ZSKs: 1 active, 0 stand-by, 0 revoked
/etc/bind/db.1.168.192.signed
Signatures generated:                        8
Signatures retained:                         0
Signatures dropped:                          0
Signatures successfully verified:            0
Signatures unsuccessfully verified:          0
Signing time in seconds:                 0.020
Signatures per second:                 400.000
Runtime in seconds:                      0.027
DNSMaster:/etc/bind/keys/1.168.192.in-addr.arpa#
```

Next, we modify `named.conf` to use the signed zone files, then we reload the service:

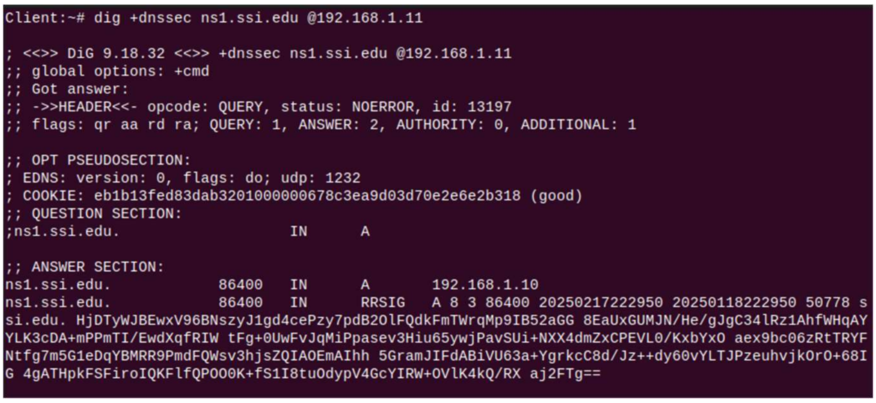
```
# forward
zone "ssi.edu" {
        type master;
        file "/etc/bind/db.ssi.edu.signed";
        # allow transfer IF authenticated with TSIG key
        allow-transfer { key "tsig-key"; };
        allow-update { none; };
        also-notify { 192.168.1.11; };
};

# reverse
zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.1.168.192.signed";
        # allow transfer IF authenticated with TSIG key
        allow-transfer { key "tsig-key"; };
        allow-update { none; };
        also-notify {192.168.1.11; };
};
```

To test the DNSMaster, from the client:

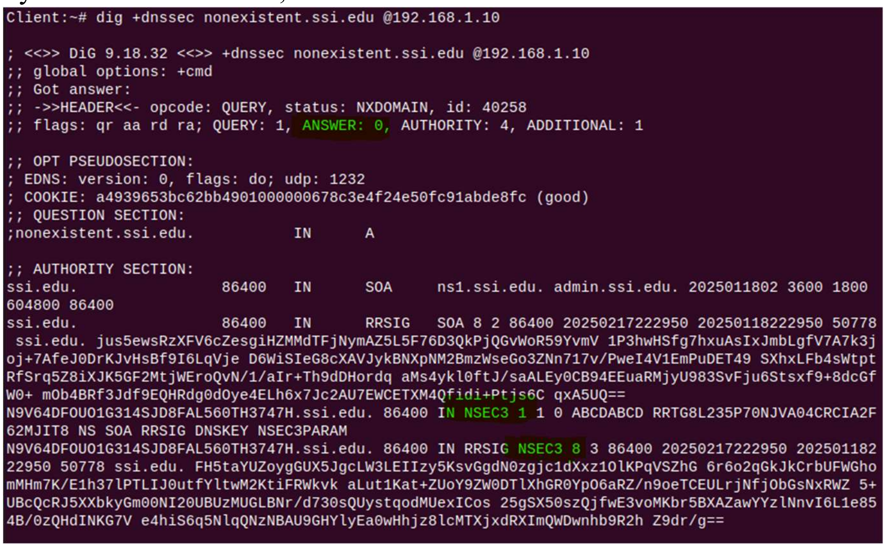
```
$ dig +dnssec ns1.ssi.edu @192.168.1.10
```

```
Client:~# dig +dnssec ns1.ssi.edu @192.168.1.10                                16:39:30 [6/1245]

; <<>> DiG 9.18.32 <<>> +dnssec ns1.ssi.edu @192.168.1.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36062
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 20e5eb07e85f70580100000678d1cb26bb0264c20b469df (good)
;; QUESTION SECTION:
;ns1.ssi.edu.                   IN      A

;; ANSWER SECTION:
ns1.ssi.edu.            86400   IN      A       192.168.1.10
ns1.ssi.edu.            86400   IN      RRSIG   A 8 3 86400 20250217222950 20250118222950 50778 s
si.edu. HjDTyWJBEwxV96BNszyJ1gd4cePzy7pdB2OlFQdkFmTWrqMp9IB52aGG 8EaUxGUMJN/He/gJgC34lRz1AhfWHqAY
YLK3cDA+mPPmTI/EwdXqfRIW tFg+0UwFvJqMiPpasev3Hiu65ywjPavSUi+NXX4dmZxCPEVL0/KxbYxO aex9bc06zRtTRYF
Ntfg7m5G1eDqYBMRR9PmdFQWsv3hjsZQIAOEmAIhh 5GramJIFdABiVU63a+YgrkcC8d/Jz++dy60vYLTJPzeuhvjkOrO+68I
G 4gATHpkFSFiroIQKFlfQPOO0K+fS1I8tuOdypV4GcYIRW+OVlK4kQ/RX aj2FTg==
```

To test the DNSSlave:



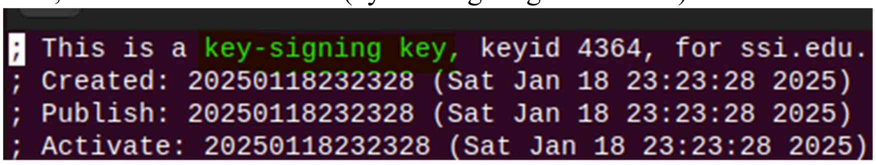
And if we try a nonexistent record, we would receive some NSEC3 records:



The DS (Delegation Signer) are used to secure delegations. A DS record with the name of the sub-delegated zone (e.g. `ssi.edu`) is placed in the parent zone (e.g. `.edu`) along with the delegating NS Records. This DS record references a DNSKEY record in the sub-delegated zone.

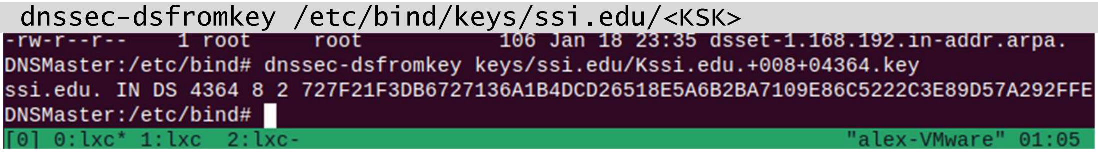
The DS record serves as a crucial link in the chain of trust, providing a cryptographic hash (digest) of the DNSKEY record in the child zone. The digest is then signed with the private key of the parent zone, adding an additional layer of security to the delegation process. By validating the DS record, DNS resolvers can verify the authenticity and integrity of the DNSKEY records associated with the sub-delegated zone, thereby improving the overall security of the DNS infrastructure.

The DSSET file is typically shared with the parent zone administrator to publish the DS record. To generate the DSSET, we locate the KSK file (by investigating the content).



Then we generate the DS record:

```
$ dnssec-dsfromkey /etc/bind/keys/ssi.edu/<KSK>
```



Exercise 2

As we have already configured the TSIG key on the servers, we can test by forcing a zone transfer on DNSSlave:

```
$ rndc retransfer ssi.edu
```

This command will immediately have the slave request a full zone transfer (AXFR) from the master. We can see this by checking /var/log/messages:

```
Jan 19 15:47:07 DNSSlave daemon.info named[1556]: received control channel command 'retransfer ss
i.edu'
Jan 19 15:47:07 DNSSlave daemon.info named[1556]: zone ssi.edu/IN: Transfer started.
Jan 19 15:47:07 DNSSlave daemon.info named[1556]: transfer of 'ssi.edu/IN' from 192.168.1.10#53:
connected using 192.168.1.10#53 TSIG tsig-key
Jan 19 15:47:07 DNSSlave daemon.info named[1556]: zone ssi.edu/IN: transferred serial 2025011802:
 TSIG 'tsig-key'
Jan 19 15:47:07 DNSSlave daemon.info named[1556]: transfer of 'ssi.edu/IN' from 192.168.1.10#53:
Transfer status: success
Jan 19 15:47:07 DNSSlave daemon.info named[1556]: transfer of 'ssi.edu/IN' from 192.168.1.10#53:
Transfer completed: 1 messages, 17 records, 3318 bytes, 0.001 secs (3318000 bytes/sec) (serial 20
25011802)
```

We can also test via `dig`:

```
DNSSlave:/etc/bind# clear                                                    16:49:41 [55/1835]
DNSSlave:/etc/bind# dig @192.168.1.10 ssi.edu AXFR -y hmac-sha256:tsig-key:TDdcWYLNAR6E+dcQoqVDWz
ccMZYm1fmgSu4kXfQu/Zw=

; <<>> DiG 9.18.32 <<>> @192.168.1.10 ssi.edu AXFR -y hmac-sha256:tsig-key:TDdcWYLNAR6E+dcQoqVDWz
ccMZYm1fmgSu4kXfQu/Zw=
; (1 server found)
;; global options: +cmd
ssi.edu.                86400   IN      SOA     ns1.ssi.edu. admin.ssi.edu. 2025011802 3600 1800
604800 86400
ssi.edu.                86400   IN      RRSIG   SOA 8 2 86400 20250217222950 20250118222950 50778
 ssi.edu. jus5ewsRzXFV6cZesgiHZMMdTFjNymAZ5L5F76D3QkPjQGvWoR59YvmV 1P3hwHSfg7hxuAsIxJmbLgfV7A7k3j
oj+7AfeJ0DrKJvHsBf9I6LqVje D6WiSIeG8cXAVJykBNXpNM2BmzWseGo3ZNn717v/PweI4V1EmPuDET49 SXhxLFb4sWtpt
RfSrq5Z8iXJK5GF2MtjWEroQvN/1/aIr+Th9dDHordq aMs4ykl0ftJ/saALEy0CB94EEuaRMjyU983SvFju6Stsxf9+8dcGf
W0+ mOb4BRf3Jdf9EQHRdg0dOye4ELh6x7Jc2AU7EWCETXM4Qfidi+Ptjs6C qxA5UQ==
ssi.edu.                86400   IN      NS      ns1.ssi.edu.
ssi.edu.                86400   IN      RRSIG   NS 8 2 86400 20250217222950 20250118222950 50778
ssi.edu. t5n6R7PA8VAhYfWV8GkD9gpWwW3VEEeeEBH4uj9AQiuxX2KhXIaQArGT 8BhuKtoUEMM4AjGsb6t+E0w5dNkKQc2
CqC9707EOJp/QcFaxMJ1/q1he jXo9P9GgLM9oCbkXpKomITiZmkBG8Ce6mnprSVPdUFXhcopahXDi+ew4 RvM9Fp6QS+tGxe
yBjk8ZKVzeoLRsUdzTp2QhiPonB2uA4ZiabtHC+FbO 2AjzVJ04dq3CxJPelG2LhPg3Xsf7YIJ0A+8vdaR0afkfk4Nzczsy5R
Wp ZnshYm0Xd1NoukEGfTpYCBvYTv6TF1KyJJxGUIDmsMxChkY8cPdeqq2a FVo2zA==
ssi.edu.                86400   IN      DNSKEY  256 3 8 AwEAAbh972llSShHb6o34zWrlqTM/qT1x564DL6vN
```

We can verify zone integrity on the DNSSlave by querying its local database and confirm the records are available:

```
$ dig @localhost ssi.edu
$ dig @localhost -x 192.168.1.11
```

```
DNSSlave:/etc/bind# dig @localhost -x 192.168.1.10

; <<>> DiG 9.18.32 <<>> @localhost -x 192.168.1.10
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40176
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 0b7f700ec4daf6a401000000678d0601f867931474c9e00c (good)
;; QUESTION SECTION:
;10.1.168.192.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
10.1.168.192.in-addr.arpa. 86400 IN     PTR     ns1.ssi.edu.
```

Exercise 3

DNS-over-TLS (DoT) is a protocol for encrypting and securing DNS queries between a DNS client and a DNS server. Traditional DNS traffic is sent in plaintext, making it vulnerable to interception and spoofing. DoT leverages Transport Layer Security (TLS) to:

- encrypt the DNS requests and responses, ensuring privacy
- authenticate the server, reducing the risk of man-in-the-middle attacks
- preserve the DNS protocol itself, but wrapped within a secure TLS tunnel on port 853 by default

By implementing DoT, we protect the DNS traffic from eavesdropping and tampering. We will create an SSL certificate and key in the directory `/etc/bind/keys`. We will be needing OpenSSL installed on the DNSMaster to create a self-signed SSL certificate:

```
$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key
  -out server.crt -subj "/CN=ns1.ssi.edu" -addext
  "subjectAltName=DNS:ns1.ssi.edu"
```

The configuration relies heavily on the BIND9 version. We are using Bind 9.18.32.

```
options {
        directory "/var/bind";
        dnssec-validation auto;

        listen-on { 192.168.1.10; };

        // Enable DNS-over-TLS
        listen-on port 853 tls "my-dot" {192.168.1.10; };

        allow-query { any; };
        allow-transfer { none; };
};

tls "my-dot" {
        cert-file "/etc/bind/keys/server.crt";
        key-file "/etc/bind/keys/server.key";
};
```

To test it, in the dig command we have to specify `-p <port>` and `+tls`. Note that without any pre-knowledge – a recognized CA / pinned certificate / trust anchor, we cannot guarantee the client is talking to the *right* DNS server.

```
Client:~# dig @192.168.1.10 -p 853 ssi.edu +tls

; <<>> DiG 9.18.32 <<>> @192.168.1.10 -p 853 ssi.edu +tls
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26554
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 9645c6c3f594d89501000000678d2f09e7055bea7a48f98b (good)
;; QUESTION SECTION:
;ssi.edu.                       IN      A

;; AUTHORITY SECTION:
ssi.edu.                86400   IN      SOA     ns1.ssi.edu. admin.ssi.edu. 2025011802 3600 1800
604800 86400
```

And if we analyze the packets exchanged,

1	0.000000000	192.168.1.120	192.168.1.10	TCP	74	43223 → 853 [SYN] Seq=0 Win=64660 Len=0 MSS=1220 SACK_PERM TSval=2969325675 TSecr=0 WS=128
2	0.000046854	192.168.1.10	192.168.1.120	TCP	74	853 → 43223 [SYN, ACK] Seq=0 Ack=1 Win=65232 Len=0 MSS=1220 SACK_PERM TSval=688149913 TSecr
3	0.000063042	192.168.1.120	192.168.1.10	TCP	66	43223 → 853 [ACK] Seq=1 Ack=1 Win=64768 Len=0 TSval=2969325675 TSecr=688149913
4	0.000551018	192.168.1.120	192.168.1.10	TLSv1.3	375	Client Hello
5	0.000582023	192.168.1.10	192.168.1.120	TCP	66	853 → 43223 [ACK] Seq=1 Ack=310 Win=65024 Len=0 TSval=688149913 TSecr=2969325675
6	0.002480107	192.168.1.10	192.168.1.120	TLSv1.3	1441	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, App
7	0.002512627	192.168.1.120	192.168.1.10	TCP	66	43223 → 853 [ACK] Seq=310 Ack=1376 Win=67200 Len=0 TSval=2969325677 TSecr=688149915
8	0.003752707	192.168.1.120	192.168.1.10	TLSv1.3	146	Change Cipher Spec, Application Data
9	0.004082113	192.168.1.10	192.168.1.120	TLSv1.3	608	Application Data, Application Data
10	0.004112082	192.168.1.120	192.168.1.10	TLSv1.3	138	Application Data
11	0.004331605	192.168.1.10	192.168.1.120	TLSv1.3	200	Application Data
12	0.005111253	192.168.1.120	192.168.1.10	TCP	66	43223 → 853 [FIN, ACK] Seq=462 Ack=2052 Win=71936 Len=0 TSval=2969325680 TSecr=688149917
13	0.005181937	192.168.1.10	192.168.1.120	TCP	66	853 → 43223 [FIN, ACK] Seq=2052 Ack=463 Win=65024 Len=0 TSval=688149918 TSecr=2969325680
14	0.005200402	192.168.1.120	192.168.1.10	TCP	66	43223 → 853 [ACK] Seq=463 Ack=2053 Win=71936 Len=0 TSval=2969325680 TSecr=688149918