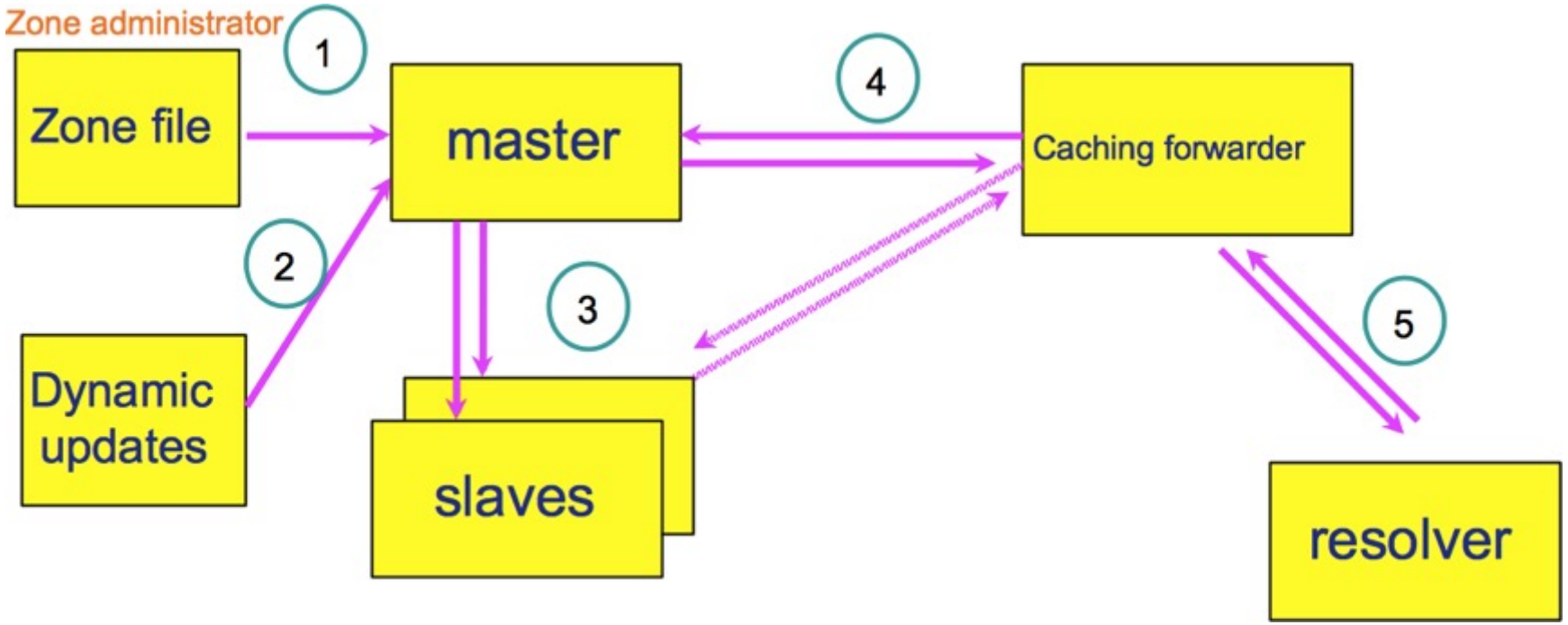
DNS Operations

"TSIG"

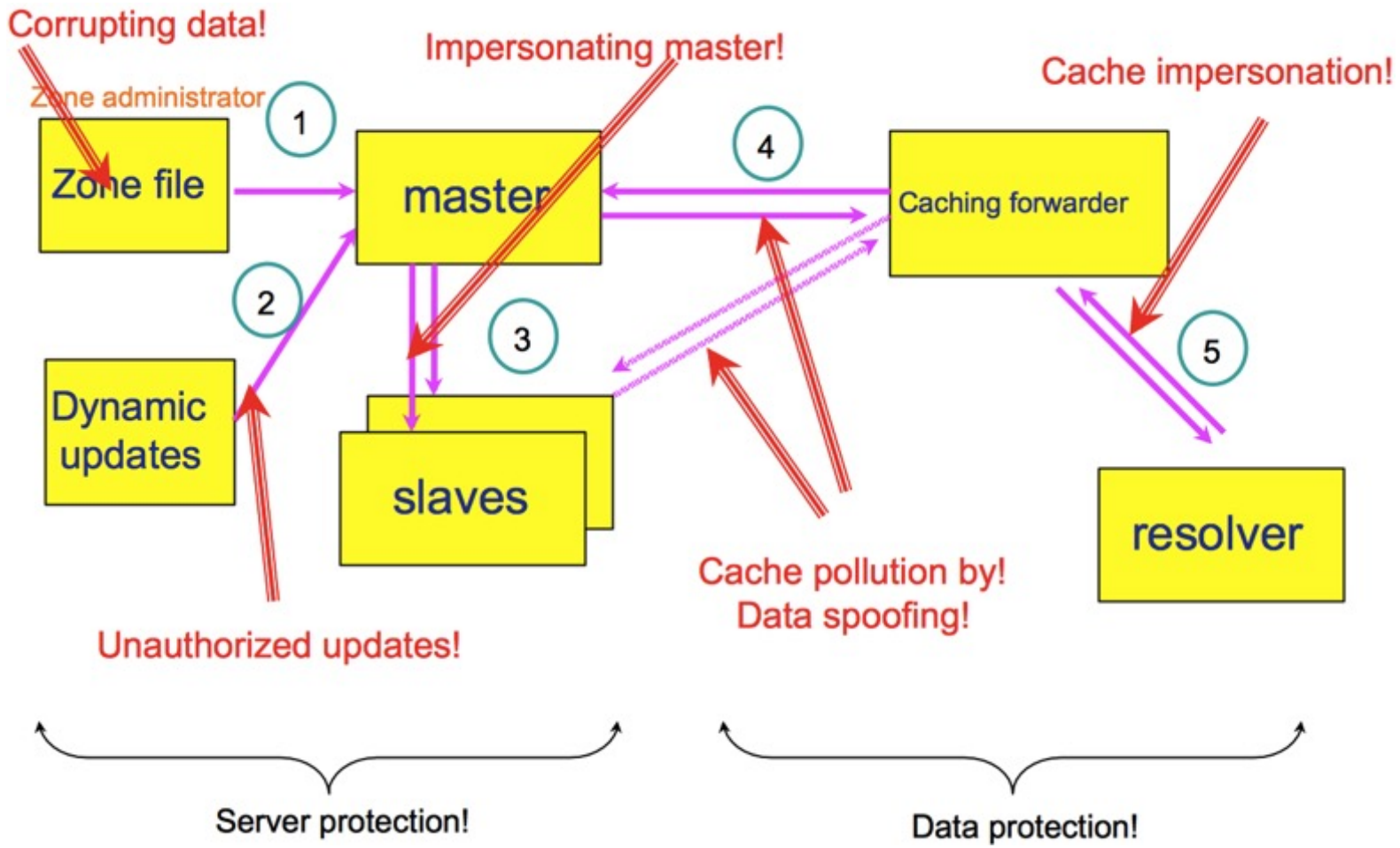
Summary

- DNS: Data Flow
- DNS Vulnerabilities
- TSIG protected vulnerabilities
- What is TSIG?
- TSIG steps
- TSIG – Generating a Secret
- TSIG – Testing with dig
- TSIG – Time

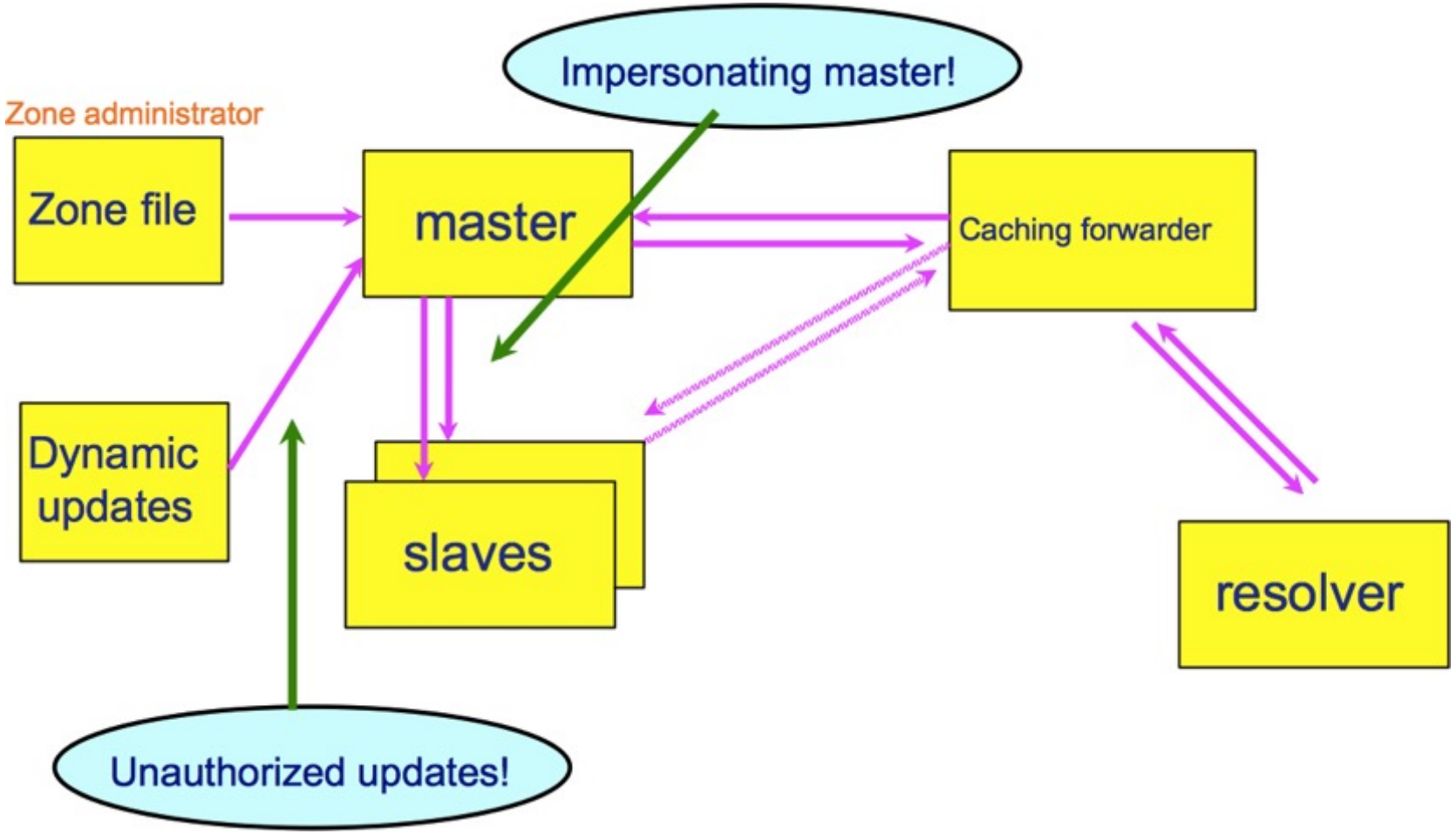
DNS: Data Flow



DNS Vulnerabilities



TSIG protected vulnerabilities



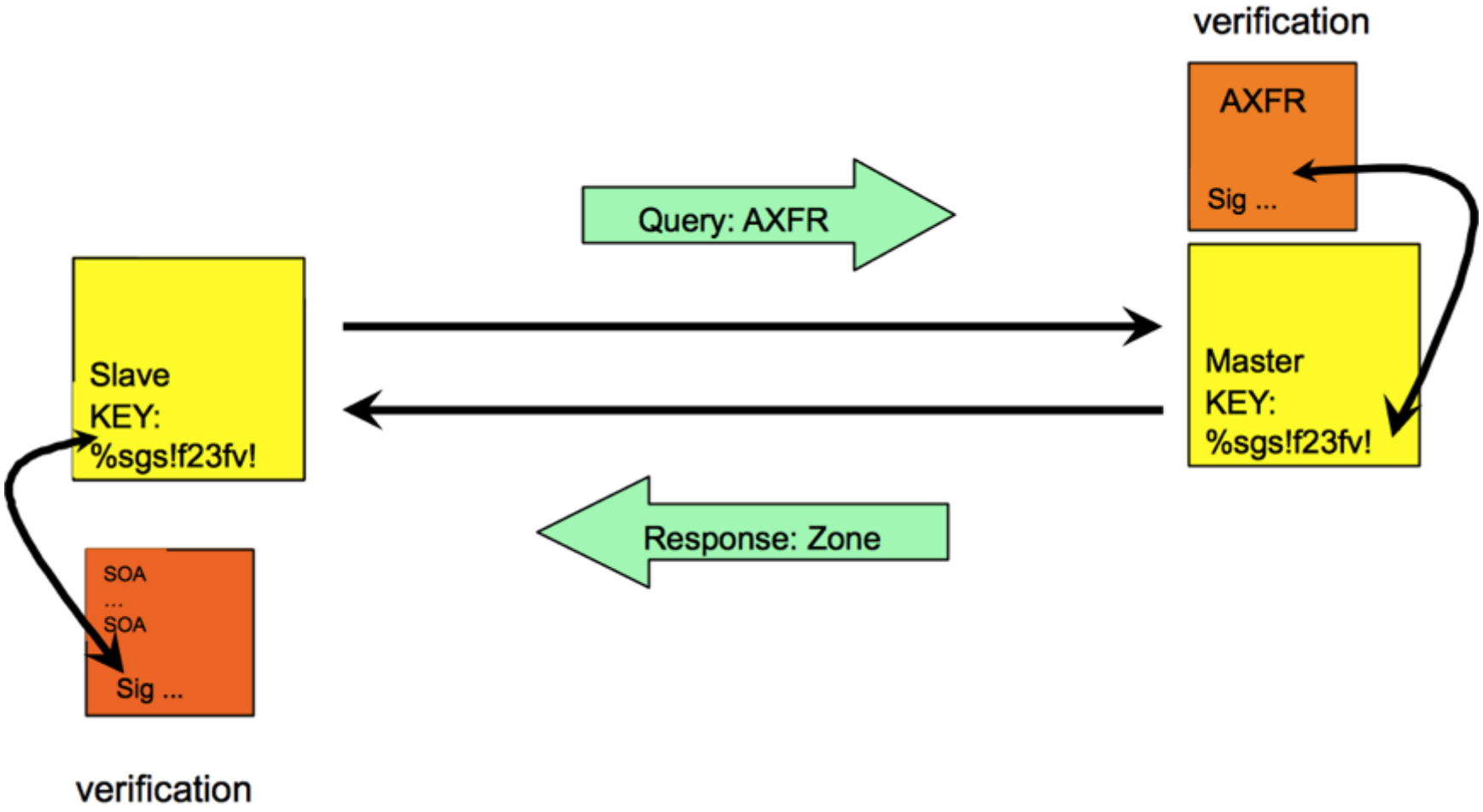
What is TSIG?

- **Transaction SIGnature:** a mechanism for protecting communication between name servers and between stub resolvers and nameservers.
- A keyed-hash is applied (like a digital signature), so the recipient of the message can verify that it hasn't been tampered with:
  - DNS question / answer
  - timestamp
- Based on a shared secret
  - Both the sender and recipient must be configured with it
  - ACLs
  - In some contexts, names of keys (more on this later)

What is TSIG?

- RFC 2845 – TSIG
- Can also be used to authorize:
  - zone transfers
  - dynamic updates
  - authentication of caching forwarders
- Used in server configuration – not in the zone file

TSIG example



TSIG steps

- Generate secret
- Communicate secret
- Configure servers
- Test

TSIG – Names & Secrets

- TSIG name
  - A name is given to the key. The name is what is transmitted in the message (so the receiver knows what key the sender has used, out of possibly many)
- TSIG secret value
  - A value determined during key generation
  - Usually seen encoded in BASE64

TSIG – Generating a Secret

- tsig-keygen Simple tool to generate TSIG keys
  - Usage: tsig-keygen [keyname]

    # tsig-keygen myserv

```
key ”myserv" {
      algorithm hmac-sha256;
      secret "YITBJ0GT9CVVznDdR4cms7Em1qP7o/f5ft+60aKNOnI=";
};
```

TSIG – Generating a Secret

- TSIG keys are **NEVER** put in the zone files
  - There was a KEY Resource Record that was replaced by DNSKEY (for DNSSEC)
  - Could cause some confusion as TSIG keys can look like those KEY RRs:

ns1-ns2.grp2.net. IN KEY 128 157 nEfRx9…bbPn7lyQtE=

TSIG – Generating a Secret

- Configuring the key:
  - in ***named.conf.local***, same syntax as for the rndc statement:

```
key "soa-ns1.grpX" {
  algorithm hmac-sha256;
  secret "3etWczaeGesi0cEpeef7PhiKCkgC2sw==";
};
```

- Using the key:
  - in ***named.conf.local***, add:

server a.b.c.d { key " soa-ns1.grpX"; };

where ‘a.b.c.d’ is the IPv4 address of the REMOTE server

  - can use IPv6 address here as an alternative

Configuration example – named.conf.local

- SOA server 100.100.X.66

```
key soa-ns1.grpX {
    algorithm hmac-sha256;
    secret “iw7tgeybfaj3u#yt…”;
};
server 100.100.X.130 {
    keys {soa-ns1.grpX ; };
};
zone “grpX.lactld.te-labs.training” {
    type primary;
    file “db.grpX”;
    allow-transfer {key soa-ns1.grpX; };
};
```

- NS1 server 100.100.X.130

```
key soa-ns1.grpX {
    algorithm hmac-sha256;
    secret “iw7tgeybfaj3u#yt…”;
};
server 100.100.X.66 {
    keys {soa-ns1.grpX ; };
};
zone “grpX.lactld.te-labs.training” {
    type secondary;
    file “db.grpX.replica”;
     masters { 100.100.X.66; }; };
};
```

TSIG – Testing with dig

- You can use dig to check TSIG configuration:
  dig @<server> <zone> AXFR –k <TSIG keyfile>
  or
  dig @<server> <zone> AXFR –y “TSIG secret”
- Wrong key will return “Transfer failed”, and a message will be logged in the security category on the server being queried
- For instance,
- dig axfr @A.B.C.D ZONE -y Algorithm:TSIG-NAME:KEY

; <<>> DiG 9.8.3-P1 <<>> axfr grpX.lab_domain… @100.100.X.66

;; global options: +cmd

; **Transfer failed.**

TSIG – Time!

- TSIG is time sensitive (to avoid replays)
- **message protection expires in 5 minutes**
- make sure time is synchronized! (NTP)
- for testing, set the time
- in operations, use NTP!

Questions!!!

DNS Operations
"DNS Access Lists in BIND"

Summary

- DNS ACLs
- Elements in an address match list
- Purposes in BIND
- Notes on Address Match list
- Example: Address match lists
- The ACL statement
- Notes on the ACL statement
- Blackhole
- allow-transfer
- allow-query
- Listen-on

DNS ACLs

- ACLs and configuration options in BIND can be used to create **more secure configurations**
- Good operational practice suggests that ACLs and configuration options be **reviewed regularly**
- Make sure they are still relevant: review them regularly to ensure they accurately reflect what you are trying to do
- They can be cumbersome and difficult to maintain

Elements in an address match list

- Individual IP addresses
- Addresses/netmask pairs
- Names of other ACLs
- In some contexts, names of keys (more on this later)

Purposes in BIND

- Restricting queries & zone xfer
- Authorizing/restricting dynamic updates
- Selecting interface to listen on
- Sorting responses
- Address match lists are always enclosed in curly braces

Notes on Address Match list

- Elements must be separated by semicolons ‘;’
- The list must be terminated with “;”
- Elements address match lists are checked sequentially
- To negate elements of the address match list, prepend them with ‘!’
- Use ACL statements to name an address match list
- ACLs must be defined before they can be used elsewhere

Example: Address match lists

- For network 192.168.0.0 255.255.255.0:

  {192.168.0.0/24; };

- For network plus loopback:

  {192.168.0.0/24; 127.0.0.1; };

- Addresses plus key name:

  {192.168.0.0/24;127.0.0.1; noc.nsrc.org; };

The ACL statement

- Syntax:

  acl acl_name { address_match_list; };

- Example:

  acl internal { 127.0.0.1; 192.168.0/24; };

  acl dynamic-update { key dhcp.ws.nsrc.org; };

Notes on the ACL statement

- The acl name need not be quoted
- There are four predefined ACLs:

any	- any IP address
none	- no IP address
localhost	- loopback, 127.0.0.1, ::1
localnets	- all the networks the name server is directly connected to.

blackhole

- specifies a list of addresses which the server does not accept queries from or use to resolve a query.
- Queries from these addresses are not responded to. The default is none.

```
options {
        blackhole { ACL-name or itemized list; };
};
```

allow-transfer

- specifies which hosts are allowed to receive zone transfers from the server.
- May also be specified in the zone statement, in which case it overrides the allow-transfer statement set in options or view.
- Default is to allow transfers to all hosts.

```
zone “myzone.example” {
  type master;
  file “myzone.example”;
  allow-transfer { ACL-name or itemized list; };
  };
```

allow-query

- specifies which hosts are allowed to ask ordinary DNS questions.
- default is to allow queries from all hosts.

```
zone “myzone.example {
  type master;
  file “myzone.example”;
  allow-query { ACL-name or itemized list; };
  };
```

listen-on

- specifies the IPv4 addresses on which the server will listen.
- listen-on and listen-on-v6 statements can each take an optional port, TLS configuration identifier, and/or HTTP configuration identifier, in addition to an address_match_list.

```
options {
    listen-on port # { ACL-name or itemized list; };
};
```

Ex:

```
listen-on { 5.6.7.8; };
listen-on port 1234 { !1.2.3.4; 1.2/16; };
```

Time for practice !!!