Práctica 11

Seguridad en un servidor DNS

Práctica 11 securizar un servidor DNS

AUTOR

José Daniel Coso del Camino

Asignatura

Seguridad en redes

IES Vidal i Barraquer de Tarragona

Mayo 2026

Índice de contenidos

1 Instalación de BIND9 y configuración de Bind9

1.1 Configuración de red

Algo indispensable es configurar la red correctamente, en nuestro caso como usamos Debian, nos olvidamos del netplan y vamos directamente al directorio `/etc/network` dentro hay un archivo que debemos modificar para modificar la IP del adaptador secundario que hemos añadido en la configuración de la máquina virtual:

```
~ 11ms
23:32:28 z network
etc/network 17ms
23:32:32 ls
if-down.d  if-post-down.d  if-pre-up.d  if-up.d  interfaces  interfaces.d
etc/network 11ms
23:32:39
```

Antes de editar el archivo, usaremos el comando `ip a` para saber el adaptador que no tiene IP así no hay posibilidad de error en la configuración de red:

```
etc/network 11ms
23:37:15 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:37:f0:79 brd ff:ff:ff:ff:ff:ff
    altname enx08002737f079
    inet 192.168.1.127/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
       valid_lft 41360sec preferred_lft 35960sec
    inet6 fe80::ca9d:2212:41f0:102b/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:5f:cd:c3 brd ff:ff:ff:ff:ff:ff
    altname enx0800275fcdc3
etc/network 17ms
23:39:07
```

En nuestro caso, la interfaz es la `enp0s8` por ende en el archivo de `/etc/network/interfaces` pondremos lo siguiente:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp

allow-hotplug enp0s8
iface enp0s8 inet static
address 172.16.1.1
netmask 255.255.255.0
gateway 172.16.1.1
```

Por último solo ejecutamos el comando `sudo systemctl restart networking`

```
etc/network 17ms
23:39:07 nvim interfaces
etc/network 7m 3.478s
23:47:02 sudo systemctl restart networking
```

Comprobamos si funciona usando de nuevo el comando `ip a`

```
~ 0ms
23:48:01 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:37:f0:79 brd ff:ff:ff:ff:ff:ff
    altname enx08002737f079
    inet 192.168.1.127/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
       valid_lft 43182sec preferred_lft 37782sec
    inet6 fe80::ca9d:2212:41f0:102b/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:5f:cd:c3 brd ff:ff:ff:ff:ff:ff
    altname enx0800275fcdc3
    inet 172.16.1.1/24 brd 172.16.1.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe5f:cdc3/64 scope link proto kernel_ll
       valid_lft forever preferred_lft forever
~ 11ms
23:48:03
```

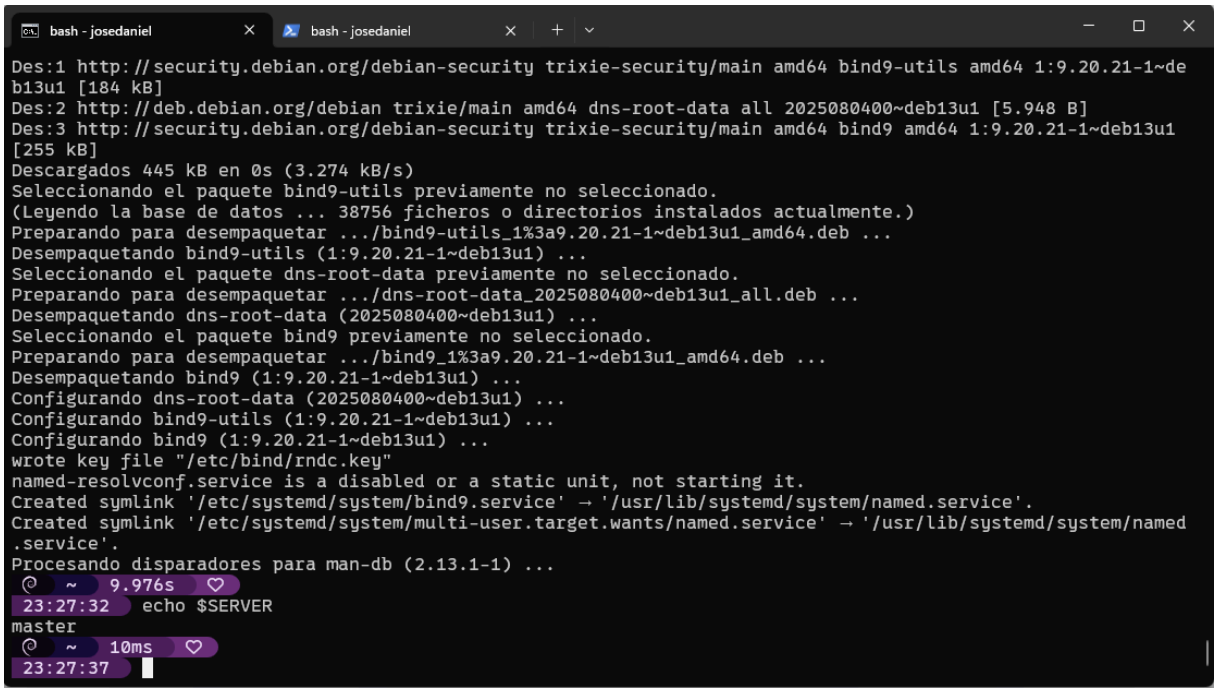
Como se puede apreciar, tenemos ahora la IP `172.16.1.1` configurada correctamente en nuestra red interna.

```
~ 4.029s
23:49:30 ping -c 4 172.16.1.1
PING 172.16.1.1 (172.16.1.1) 56(84) bytes of data.
64 bytes from 172.16.1.1: icmp_seq=1 ttl=64 time=0.017 ms
64 bytes from 172.16.1.1: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 172.16.1.1: icmp_seq=3 ttl=64 time=0.027 ms
64 bytes from 172.16.1.1: icmp_seq=4 ttl=64 time=0.032 ms

--- 172.16.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.017/0.028/0.038/0.007 ms
~ 3.079s
23:49:39
```

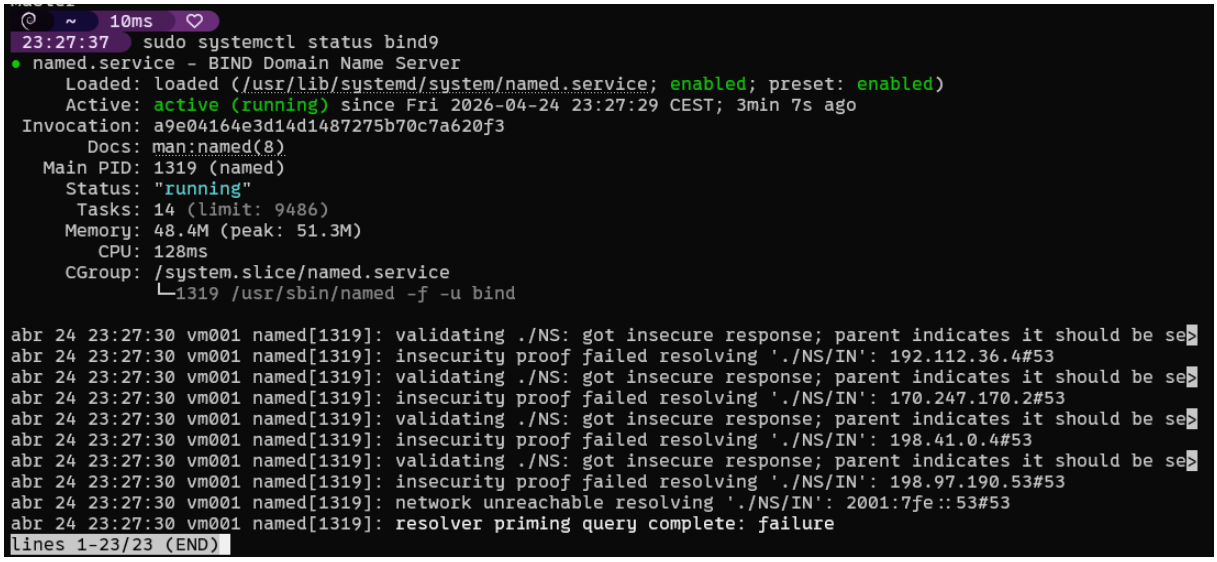
Tendremos que repetir el mismo proceso con el otro servidor, solo que en vez de la IP `172.16.1.1` será la `172.16.1.2`

1.2 Instalación de bind9 en el servidor maestro

Para instalar bind9 en Debian, solo debemos ejecutar el comando `sudo apt update -y && sudo apt upgrade -y && sudo apt install bind9 -y` una vez lo ejecutamos nos saldrá lo siguiente:



Acto seguido comprobaremos si funciona el bind9, más bien si está arrancado y funcionando usando el comando `sudo systemctl status bind9`:



En nuestro caso está funcionando correctamente

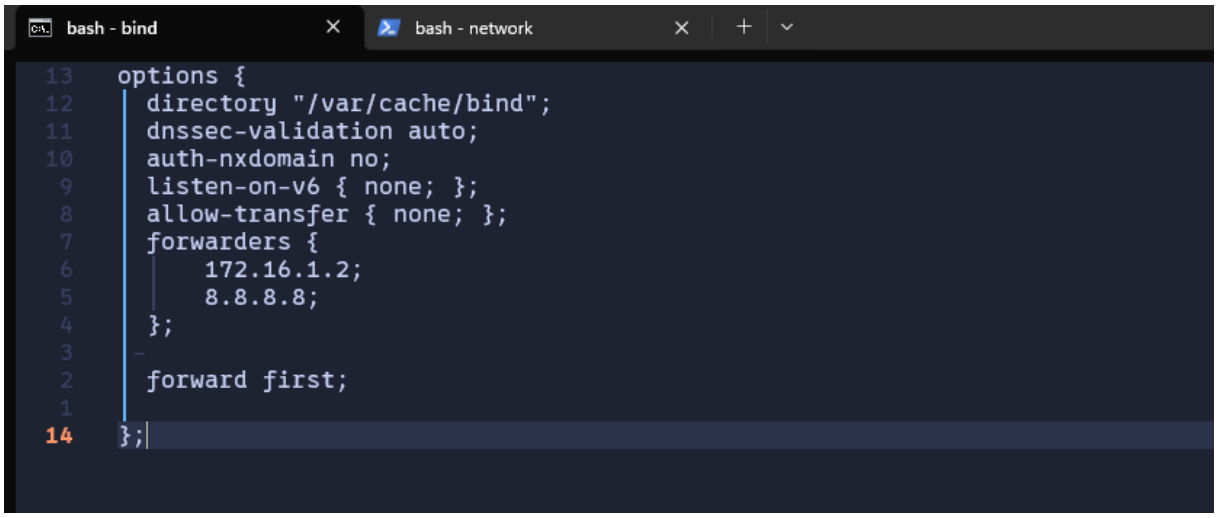
1.3 Instalación de bind9 en el servidor esclavo

Repetimos el mismo proceso con el servidor esclavo, instalar bind9 y probar si el servicio está arrancado y funciona correctamente.

```
~ 10ms 127
23:24:58 echo $SERVER
slave
~ 9ms
23:25:26 sudo apt install bind9
bind9 ya está en su versión más reciente (1:9.20.21-1~deb13u1).
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libice6              libmsgpack-c2       libunibilium4  libxt6t64  neovim-runtime    python3-pynvim
  libluajit-5.1-2      libsm6              libvterm0      lua-lpeg   python3-greenlet  x11-common
  libluajit-5.1-common libtree-sitter0.22  libxmu6        lua-luv    python3-msgpack   xclip
Utilice «sudo apt autoremove» para eliminarlos.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
~ 470ms
23:25:31 sudo systemctl status bind9
[sudo] contraseña para josedaniel:
● named.service - BIND Domain Name Server
     Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
     Active: active (running) since Fri 2026-04-24 23:24:29 CEST; 21min ago
 Invocation: fbf06cee5de84529924329193ab1a324
       Docs: man:named(8)
   Main PID: 1837 (named)
     Status: "running"
      Tasks: 14 (limit: 9486)
     Memory: 47.8M (peak: 49.9M)
        CPU: 160ms
     CGroup: /system.slice/named.service
             └─1837 /usr/sbin/named -f -u bind

abr 24 23:24:31 vm001 named[1837]: validating ./NS: got insecure response; parent indicates it should be secure
abr 24 23:24:31 vm001 named[1837]: insecurity proof failed resolving './NS/IN': 192.33.4.12#53
abr 24 23:24:31 vm001 named[1837]: validating ./NS: got insecure response; parent indicates it should be secure
abr 24 23:24:31 vm001 named[1837]: insecurity proof failed resolving './NS/IN': 199.7.91.13#53
abr 24 23:24:31 vm001 named[1837]: validating ./NS: got insecure response; parent indicates it should be secure
abr 24 23:24:31 vm001 named[1837]: insecurity proof failed resolving './NS/IN': 199.7.83.42#53
abr 24 23:24:31 vm001 named[1837]: validating ./NS: got insecure response; parent indicates it should be secure
abr 24 23:24:31 vm001 named[1837]: insecurity proof failed resolving './NS/IN': 193.0.14.129#53
abr 24 23:24:31 vm001 named[1837]: network unreachable resolving './NS/IN': 2001:dc3::35#53
abr 24 23:24:31 vm001 named[1837]: resolver priming query complete: failure
~ 2.887s
23:46:06
```

1.4 Configuración servidor master

Primero empezaremos por configurar el archivo `named.conf.options` el cual pondremos el siguiente contenido:

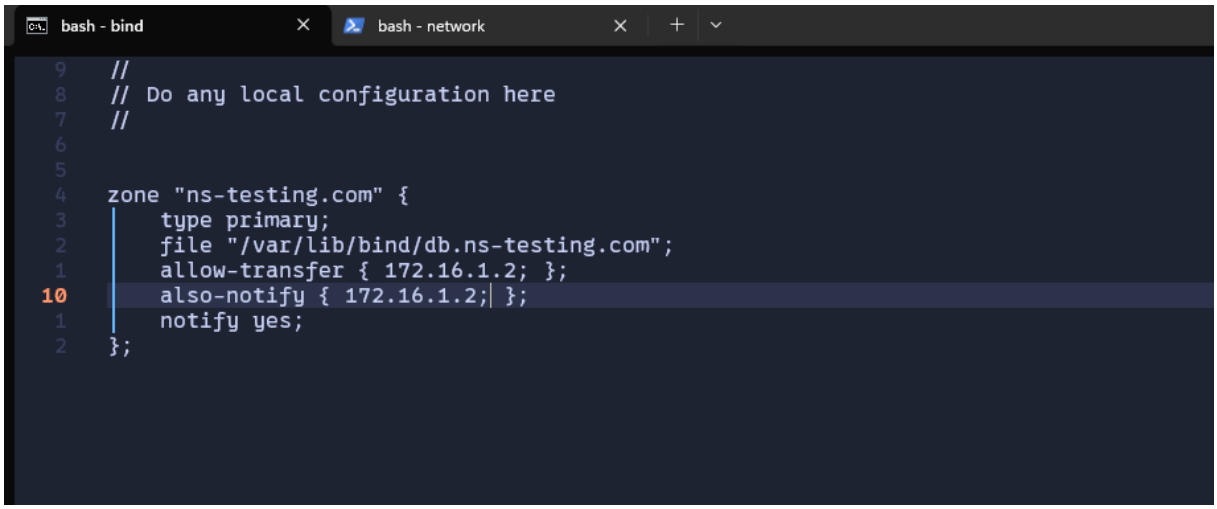


Detallemos brevemente que hace cada directiva de este archivo:

1. `directory: "/var/cache/bind";`: Define el directorio de trabajo del servidor DNS. Bind9 usa esta ruta como base para guardar archivos de caché, buscar archivos de zona si se especifican con rutas relativas y escribir archivos temporales internos.
2. `dnssec-validation auto;`: Activa la validación DNSSEC automática. Bind9 verifica que las respuestas DNS estén correctamente firmadas criptográficamente.

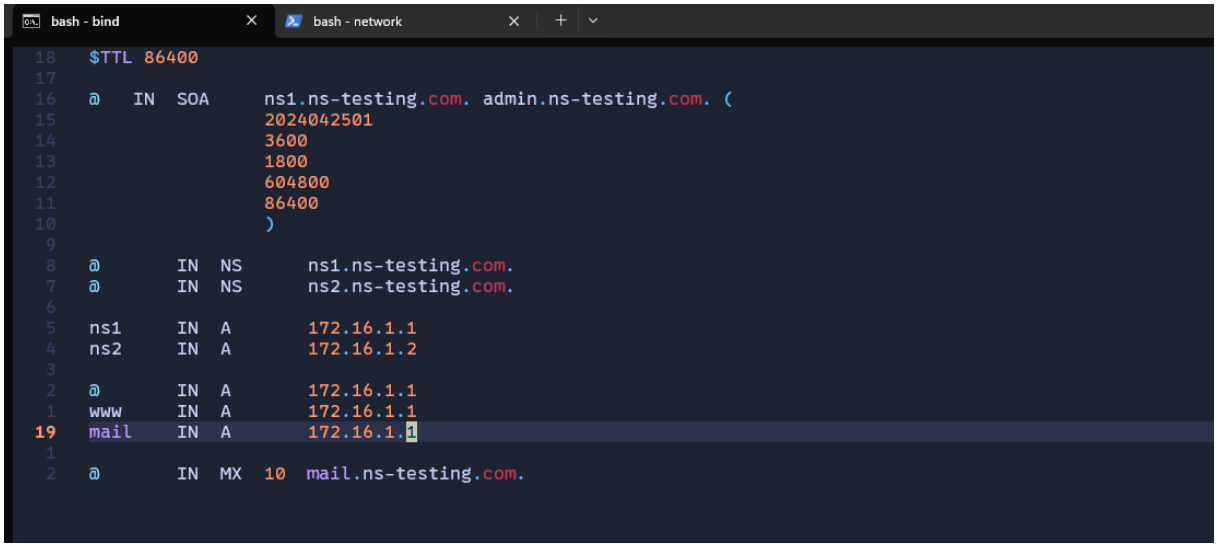
3. `auth-nxdomain no;`: Controla si el servidor bind puede responder con el flag AA (Authoritative Answer) en respuestas NXDOMAIN.
4. `listen-on-v6 { no; };`: Indica que bind no debe escuchar en interfaces ipv6.
5. `forwarders { 172.16.1.2; 8.8.8.8; };`: Define los servidores DNS a los que bind reenvía las consultas que no puede resolver por si mismo.
6. `allow-transfer { none; };` bloquea la transferencia hacia otros servidores DNS

Acto seguido configuraremos lo siguiente en el archivo `named.conf.local`



1. `zone "ns-testing.com" { ... };`: define el inicio del bloque de configuración para la zona de dominio específica denominada «ns-testing.com».
2. `type primary;`: indica que este servidor es el servidor maestro o principal para esta zona, lo que significa que posee la copia original y editable del archivo de registros.
3. `file "/var/lib/bind/db.ns-testing.com";`: especifica la ruta absoluta en el sistema de archivos donde se encuentra almacenado el archivo de base de datos que contiene los registros dns de la zona.
4. `allow-transfer { 172.16.1.2; };`: define una directiva de seguridad que permite únicamente a la dirección ip 172.16.1.2 solicitar y recibir una copia completa de la zona (transferencia de zona).
5. `also-notify { 172.16.1.2; };`: establece que, además de los servidores listados en los registros ns, se debe enviar un aviso de actualización específicamente a la ip 172.16.1.2 cuando haya cambios.
6. `notify yes;`: activa el envío automático de notificaciones a los servidores secundarios cada vez que el número de serie de la zona se incrementa, para que estos inicien la sincronización.

Seguidamente, crearemos el archivo de la zona directa en el directorio `/var/lib/bind/db.ns-testing.com` con el siguiente contenido:



1. `$TTL 86400`: define el tiempo de vida (time to live) por defecto para todos los registros de la zona, en este caso 24 horas (86400 segundos).
2. `@ IN SOA ...`: registro de inicio de autoridad (start of authority). Contiene el servidor dns principal (ns1), el correo del administrador (admin) y los parámetros de sincronización (serial, refresh, retry, expire y negative cache ttl).
3. `@ IN NS ...`: define los servidores de nombres (name servers) autoritativos para el dominio, que son ns1 y ns2.
4. `ns1 IN A 172.16.1.1`: registro de dirección que vincula el nombre del servidor dns principal con su dirección ipv4.
5. `ns2 IN A 172.16.1.2`: registro de dirección que vincula el nombre del servidor dns secundario con su dirección ipv4.
6. `@ IN A 172.16.1.1`: apunta el dominio base (ns-testing.com) directamente a la ip 172.16.1.1.
7. `www IN A 172.16.1.1`: define la dirección ip para el subdominio web clásico (www).
8. `mail IN A 172.16.1.1`: define la dirección ip del servidor que gestionará el correo electrónico.
9. `@ IN MX 10 mail...`: registro de intercambio de correo (mail exchanger) que indica que los correos enviados al dominio deben dirigirse al servidor «mail» con una prioridad de 10.

Acto seguido, reiniciamos el servidor bind9 con el comando `sudo systemctl restart bind9` y si todo ha ido bien si ejecutamos `sudo systemctl status bind9` nos debería salir lo siguiente:

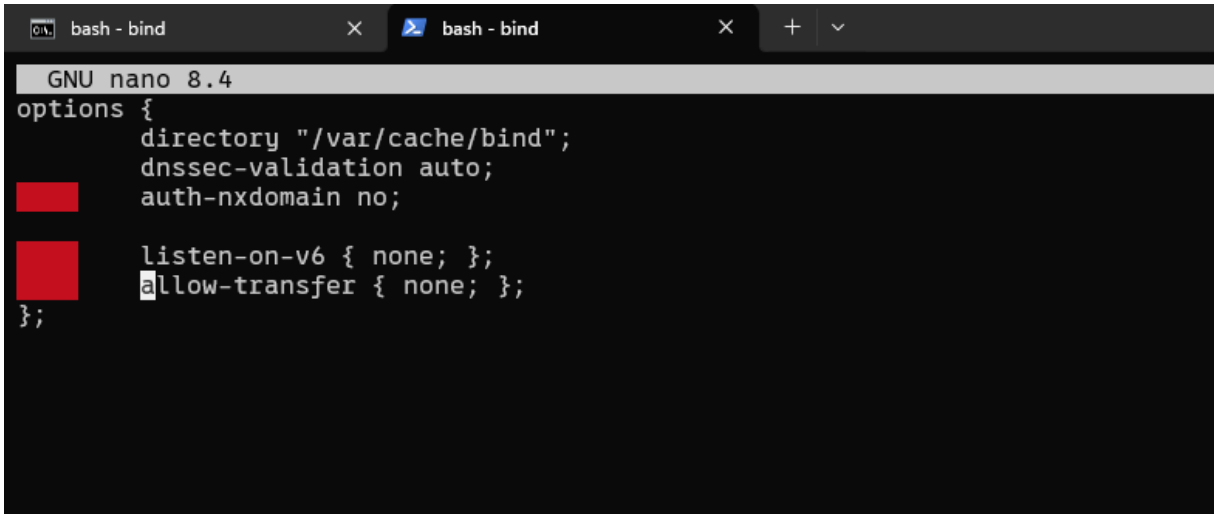
```
 etc/bind  1m 57.557s
01:31:32  sudo systemctl restart bind9
 etc/bind  143ms
01:31:39  sudo systemctl status bind9
● named.service - BIND Domain Name Server
     Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
     Active: active (running) since Sat 2026-04-25 01:31:39 CEST; 5s ago
 Invocation: 74bdffbaa6e8477d865cdddb41335a28
       Docs: man:named(8)
   Main PID: 5917 (named)
     Status: "running"
      Tasks: 10 (limit: 9486)
     Memory: 41M (peak: 41.5M)
        CPU: 71ms
     CGroup: /system.slice/named.service
             └─5917 /usr/sbin/named -f -u bind

abr 25 01:31:39 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 192.58.128.30#53
abr 25 01:31:39 vm001 named[5917]: validating ./NS: got insecure response; parent indicates it should be secure
abr 25 01:31:39 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 192.33.4.12#53
abr 25 01:31:40 vm001 named[5917]: validating ./NS: got insecure response; parent indicates it should be secure
abr 25 01:31:40 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 202.12.27.33#53
abr 25 01:31:40 vm001 named[5917]: validating ./NS: got insecure response; parent indicates it should be secure
abr 25 01:31:40 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 193.0.14.129#53
abr 25 01:31:40 vm001 named[5917]: validating ./NS: got insecure response; parent indicates it should be secure
abr 25 01:31:40 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 192.5.5.241#53
abr 25 01:31:40 vm001 named[5917]: resolver priming query complete: failure
 etc/bind  41ms
01:31:44
```

Podemos ejecutar más comprobaciones con los siguientes comandos:

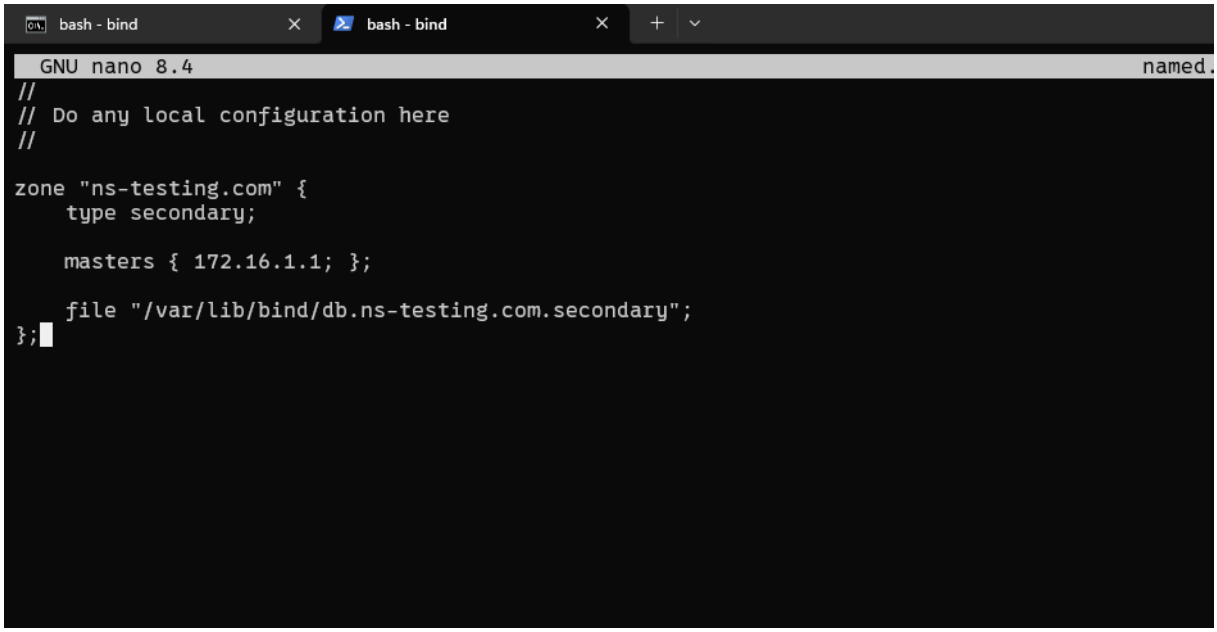
```
 etc/bind  41ms
01:31:44  sudo named-checkconf
 etc/bind  42ms
01:36:57  sudo named-checkzone ns-testing.com /var/lib/bind/db.ns-testing.com
zone ns-testing.com/IN: loaded serial 2024042501
OK
 etc/bind  106ms
01:37:03  sudo rndc reload ns-testing.com
zone reload up-to-date
 etc/bind  42ms
01:37:07  sudo systemctl status bind9
● named.service - BIND Domain Name Server
     Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
     Active: active (running) since Sat 2026-04-25 01:31:39 CEST; 5min ago
 Invocation: 74bdffbaa6e8477d865cdddb41335a28
       Docs: man:named(8)
   Main PID: 5917 (named)
     Status: "running"
      Tasks: 14 (limit: 9486)
     Memory: 48.8M (peak: 51M)
        CPU: 74ms
     CGroup: /system.slice/named.service
             └─5917 /usr/sbin/named -f -u bind

abr 25 01:31:39 vm001 named[5917]: validating ./NS: got insecure response; parent indicates it should be secure
abr 25 01:31:39 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 192.33.4.12#53
abr 25 01:31:40 vm001 named[5917]: validating ./NS: got insecure response; parent indicates it should be secure
abr 25 01:31:40 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 202.12.27.33#53
abr 25 01:31:40 vm001 named[5917]: validating ./NS: got insecure response; parent indicates it should be secure
abr 25 01:31:40 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 193.0.14.129#53
abr 25 01:31:40 vm001 named[5917]: validating ./NS: got insecure response; parent indicates it should be secure
abr 25 01:31:40 vm001 named[5917]: insecurity proof failed resolving './NS/IN': 192.5.5.241#53
abr 25 01:31:40 vm001 named[5917]: resolver priming query complete: failure
abr 25 01:37:07 vm001 named[5917]: received control channel command 'reload ns-testing.com'
 etc/bind  39ms
01:37:11
```

1.5 Configuración servidor slave

Una vez hemos configurado el servidor maestro, pasamos al servidor esclavo en el que empezaremos configurando el archivo `named.conf.options` con el siguiente contenido:

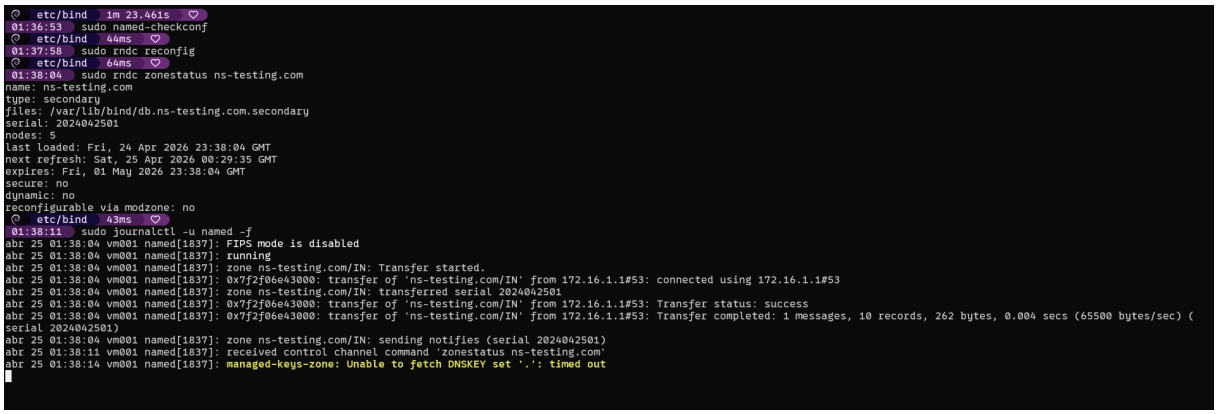


Una vez hemos configurado dicho archivo, pasamos a configurar el archivo `named.conf.local` con el siguiente contenido:

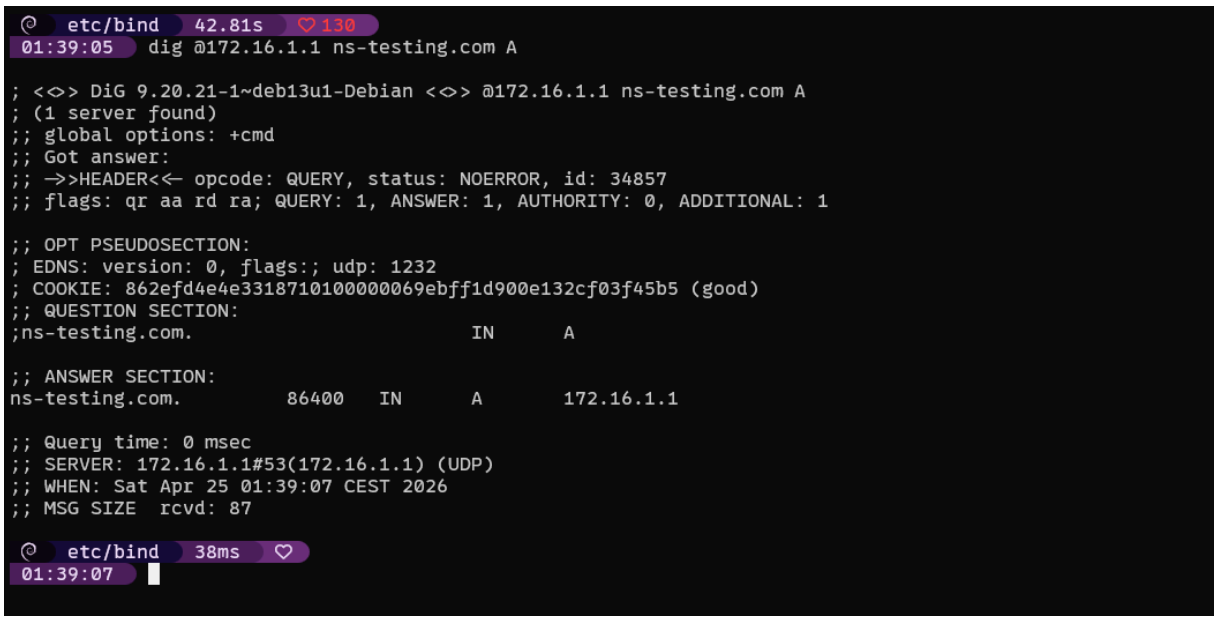


1. `zone "ns-testing.com" { ... };`: al igual que en el primario, inicia el bloque de configuración para el dominio «ns-testing.com».
2. `type secondary;`: define que este servidor actúa como secundario para la zona. En versiones antiguas de BIND se usaba el término «slave». Significa que el servidor no edita los registros, sino que los replica del maestro.
3. `masters { 172.16.1.1; };`: especifica la dirección ip del servidor maestro (el servidor primario) desde el cual este servidor debe solicitar las transferencias de zona para mantener los datos actualizados.
4. `file "/var/lib/bind/db.ns-testing.com.secondary";`: indica dónde se guardará la copia local de los datos recibidos del maestro. Es una buena práctica usar una extensión como «.secondary» o «.bak» para diferenciarlo visualmente del archivo original en el nodo principal.

Una vez hecho esto, recargamos el servicio y comprobaremos que todo está funcionando como debería:



Como se puede apreciar, se ha hecho la transferencia desde el servidor maestro correctamente. Hagamos un par de comprobaciones más con el comando `dig`:



```
etc/bind 38ms
01:39:07 dig @172.16.1.2 ns-testing.com A

; <<>> DiG 9.20.21-1~deb13u1-Debian <<>> @172.16.1.2 ns-testing.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51050
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 3c22c7855acbbd070100000069ebff3ef1234287449eb999 (good)
;; QUESTION SECTION:
;ns-testing.com.                        IN      A

;; ANSWER SECTION:
ns-testing.com.         86400   IN      A       172.16.1.1

;; Query time: 4 msec
;; SERVER: 172.16.1.2#53(172.16.1.2) (UDP)
;; WHEN: Sat Apr 25 01:39:42 CEST 2026
;; MSG SIZE  rcvd: 87

etc/bind 30ms
01:39:42
```

1.6 Configuración servidor recursivo

Acto seguido configuraremos el servidor recursivo, para ello tenemos que hacer un par de modificaciones al archivo `named.conf.options` en mi caso he creado una máquina virtual nueva con un adaptador nuevo con la IP `172.16.1.3` y he configurado lo siguiente en el archivo:

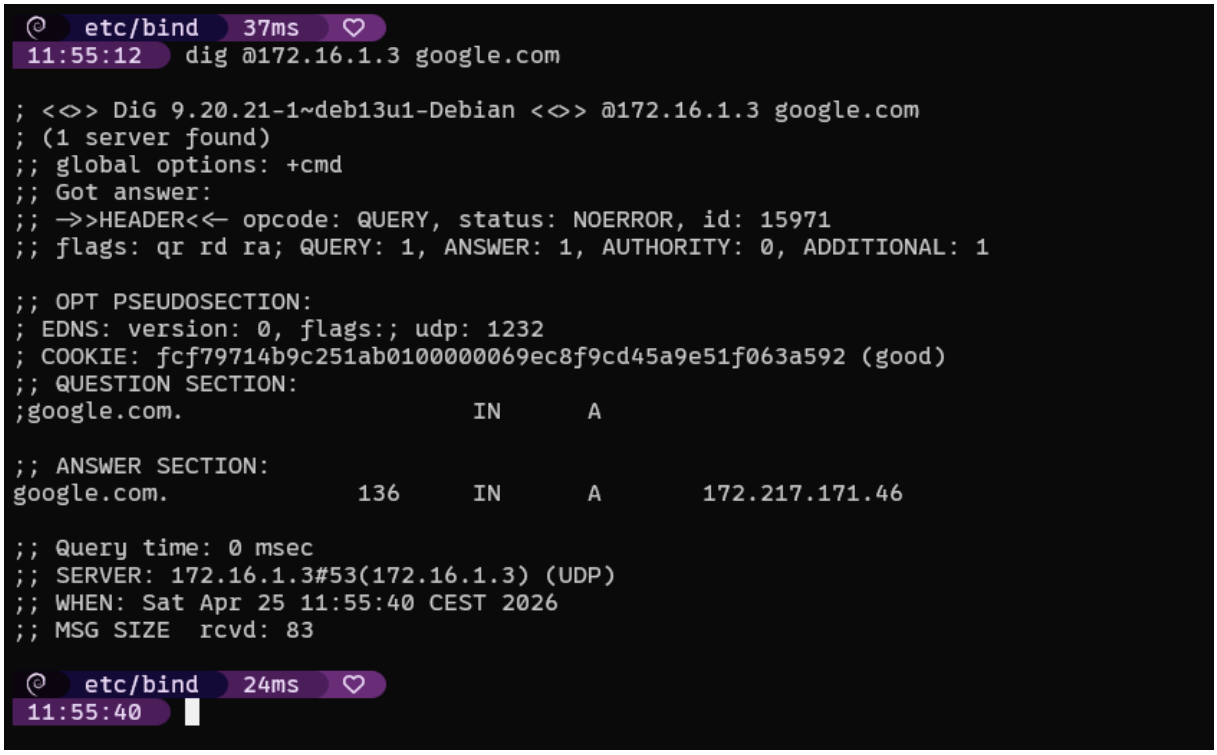
```
1   acl "trusted" {
1           127.0.0.0/8;
2           172.16.1.0/24;
3   };
4
5   options {
6           directory "/var/cache/bind";
7
8           recursion yes;
9           allow-query { trusted; };
10          allow-recursion { trusted; };
11
12          dnssec-validation no;
13
14          listen-on-v6 { any; };
15
16          forwarders {
17                  8.8.8.8;
18                  1.1.1.1;
19          };
20
21          forward first;
22  };
```

1. `acl "trusted" { ... };`: define una lista de control de acceso (access control list) llamada «trusted» que agrupa redes de confianza: el bucle local (127.0.0.0/8) y la red privada 172.16.1.0/24.
2. `directory "/var/cache/bind";`: establece el directorio de trabajo donde el servidor almacenará archivos temporales y de caché.
3. `recursion yes;`: habilita la capacidad del servidor para realizar consultas recursivas en nombre de los clientes, buscando respuestas en otros servidores si no las tiene.
4. `allow-query { trusted; };`: restringe quién puede realizar consultas estándar al servidor, permitiéndolo únicamente a los hosts definidos en la acl «trusted».
5. `allow-recursion { trusted; };`: limita la función de recursividad solo a los clientes de la lista «trusted», evitando que el servidor sea explotado en ataques de amplificación por terceros.
6. `dnssec-validation no;`: desactiva la validación de firmas dnssec. En un entorno de producción, esto debería estar en «yes» o «auto» para garantizar la integridad de las respuestas.
7. `listen-on-v6 { any; };`: configura al servidor para que escuche peticiones dns a través de todas las interfaces de red ipv6 disponibles.
8. `forwarders { 8.8.8.8; 1.1.1.1; };`: define servidores dns externos (en este caso Google y Cloudflare) a los que se enviarán las consultas que este servidor no pueda resolver por sí mismo.
9. `forward first;`: indica que el servidor intentará primero consultar a los «forwarders» y, solo si estos no responden, intentará resolver la consulta por su cuenta usando los servidores raíz.

Comprobamos si funciona con los siguientes comandos y reiniciamos el servicio:

```
 etc/bind  53.953s 
11:55:07  sudo systemctl status bind9
● named.service - BIND Domain Name Server
     Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
     Active: active (running) since Sat 2026-04-25 11:53:47 CEST; 1min 25s ago
 Invocation: 352529354ea24fc98addce8d8e1e704b
       Docs: man:named(8)
   Main PID: 1740 (named)
     Status: "running"
      Tasks: 10 (limit: 9486)
     Memory: 46.7M (peak: 46.8M)
        CPU: 51ms
     CGroup: /system.slice/named.service
             └─1740 /usr/sbin/named -f -u bind

abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:7fe::53#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:500:1::53#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:500:2d::d#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:7fd::1#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:500:12::d0d#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:500:9f::42#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2801:1b8:10::b#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:500:2::c#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:500:a8::e#53
abr 25 11:53:51 vm001 named[1740]: network unreachable resolving './NS/IN': 2001:dc3::35#53
 etc/bind  37ms 
11:55:12 
```

Como se puede ver ha arrancado correctamente, ahora si intentamos hacer una consulta a fuera a internet nos debería de responder:

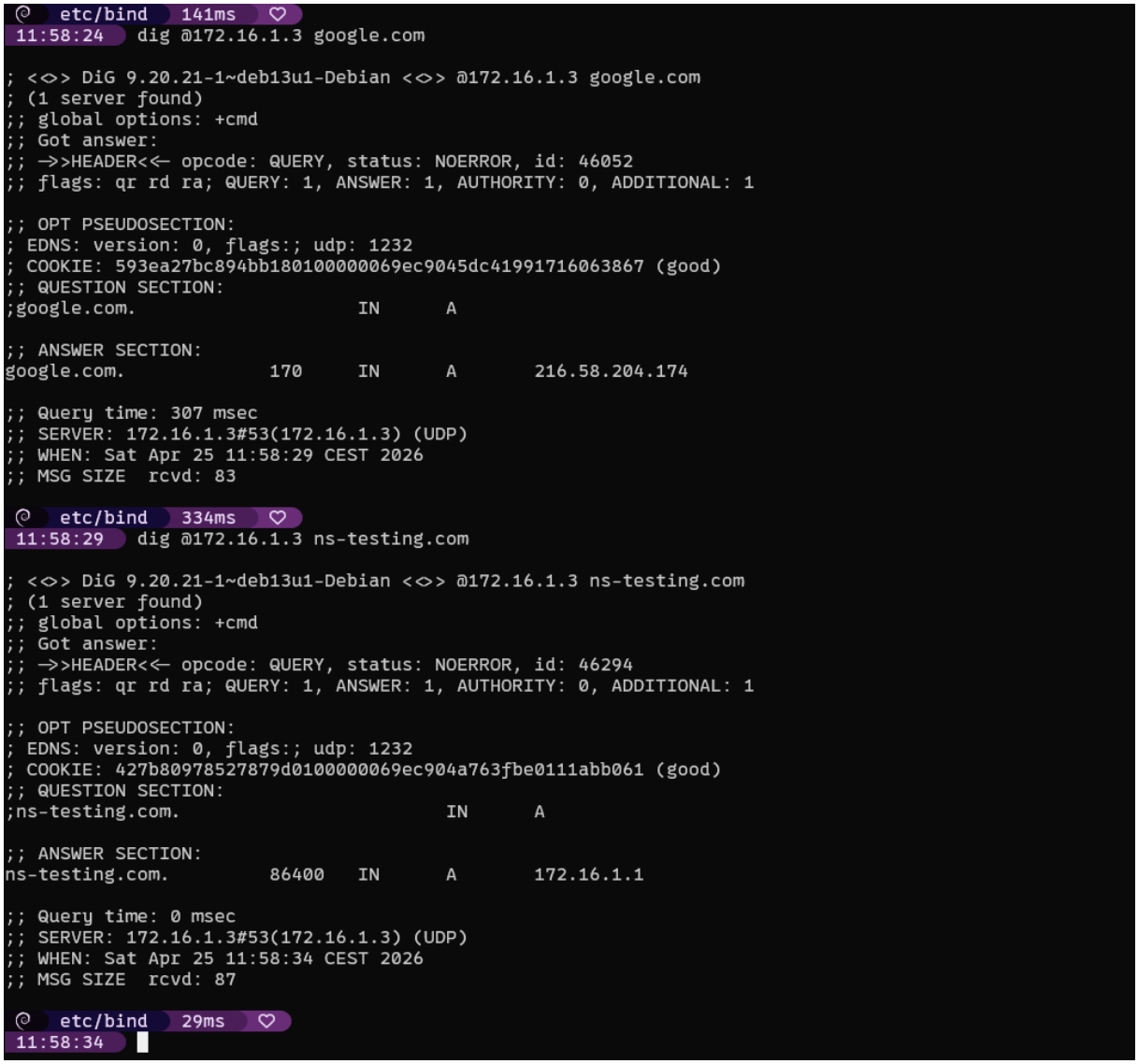


Ahora configuraremos una zona tipo forward para resolver los dominios internos que hemos configurado anteriormente editando el archivo `/etc/bind/named.conf.local` y pondremos el siguiente contenido:

```
zone "ns-testing.com" {
        type forward;
        forwarders { 172.16.1.1; 172.16.1.2; };
};
```

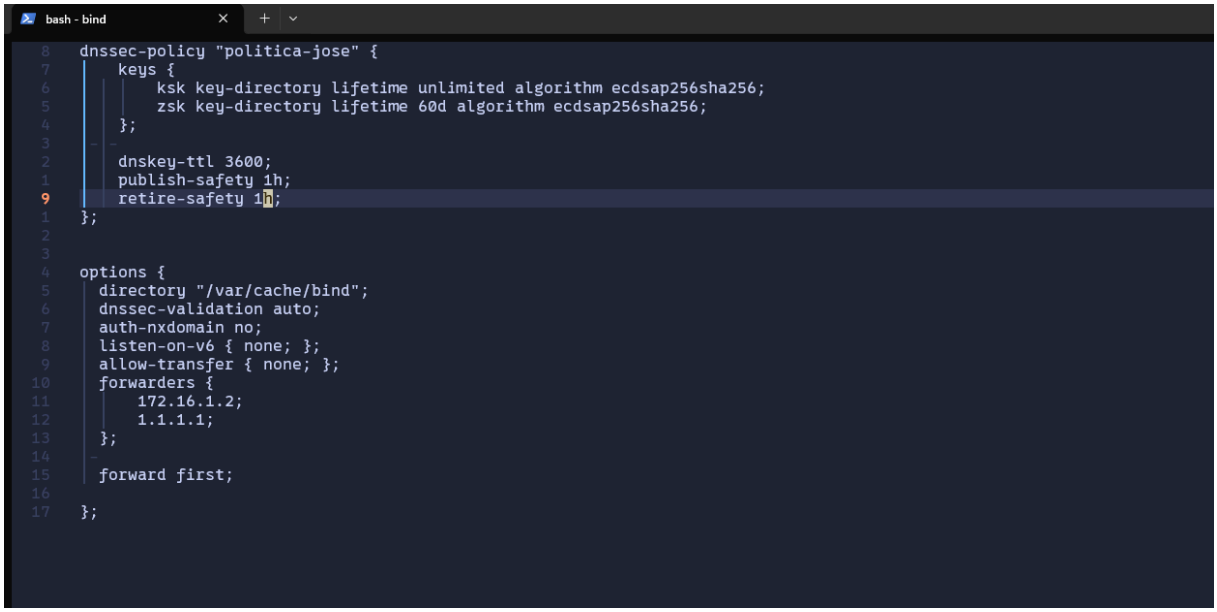
1. `zone "ns-testing.com" { ... };`: identifica el bloque de configuración para el dominio «ns-testing.com».
2. `type forward;`: define que esta zona es de tipo reenvío. A diferencia de las zonas primarias o secundarias, este servidor no contiene ninguna copia de la base de datos de registros; simplemente sabe a quién debe preguntarle por ellos.
3. `forwarders { 172.16.1.1; 172.16.1.2; };`: especifica las direcciones ip de los servidores dns específicos a los que se deben delegar todas las consultas relacionadas con este dominio en particular. Es útil para redirigir tráfico de un dominio específico hacia servidores internos o remotos que tienen la autoridad real sobre él.

Guardamos el archivo y reiniciamos, una vez hecho esto probamos de nuevo con dig:



1.7 Configuración de DNSSEC en servidores autoritativos

Primeramente para configurar DNSSEC, necesitamos crear una política en mi caso la he llamado `politica-jose`:



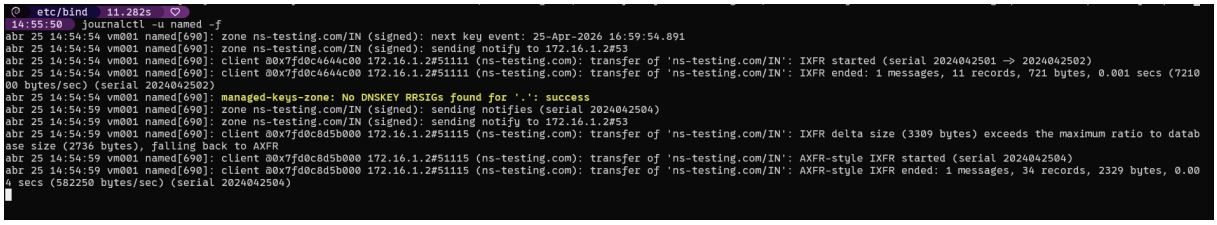
1. `dnssec-policy "politica-jose" { ... };`: define un conjunto de reglas personalizadas bajo el nombre «politica-jose» para automatizar la gestión de claves y firmas dnssec (key rollover).
2. `ksk key-directory lifetime unlimited algorithm ecdsap256sha256;`: configura la clave de firma de claves (key signing key). Se establece que no tiene fecha de expiración automática (lifetime unlimited) y utiliza el algoritmo ecdsa con p-256 y sha-256, que es más eficiente y genera firmas más pequeñas que rsa.
3. `zsk key-directory lifetime 60d algorithm ecdsap256sha256;`: configura la clave de firma de zona (zone signing key). A diferencia de la ksk, esta tiene un tiempo de vida de 60 días, tras los cuales bind generará una nueva automáticamente para rotarla.
4. `dnskey-ttl 3600;`: define el tiempo de vida (ttl) para los registros dnskey publicados en la zona, establecido en una hora (3600 segundos).
5. `publish-safety 1h;`: establece un margen de seguridad temporal al publicar nuevas claves, asegurando que se propaguen por la red antes de empezar a usarlas para firmar.
6. `retire-safety 1h;`: define un periodo de espera adicional antes de eliminar por completo una clave antigua de la zona, permitiendo que cualquier respuesta aún presente en cachés externas expire.

Seguidamente, modificamos en el archivo de la zona `/etc/bind/named.conf.local` añadiremos dicha política y el `inline-signing yes;`

bash - bind

```
//
// Do any local configuration here
//

zone "ns-testing.com" {
    type primary;
    file "/var/lib/bind/db.ns-testing.com";
    dnssec-policy "politica-jose";
    inline-signing yes;
    allow-transfer { 172.16.1.2; };
    also-notify { 172.16.1.2; };
    notify yes;
};
```

Guardamos los archivos, recargamos y comprobamos ejecutando el comando `journal -u named -f`:



Como se puede ver ya ha generado las claves y las ha firmado.

Si queremos hacer una comprobación adicional podemos ejecutar el comando `rndc dnssec -status ns-testing.com`

```
 etc/bind  0ms  130
 14:57:43  sudo rndc dnssec -status ns-testing.com
dnssec-policy: politica-jose
current time:  Sat Apr 25 14:57:45 2026

key: 32224 (ECDSAP256SHA256), ZSK
  published:      yes - since Sat Apr 25 14:54:54 2026
  zone signing:   yes - since Sat Apr 25 14:54:54 2026

  Next rollover scheduled on Wed Jun 24 12:49:54 2026
  - goal:           omnipresent
  - dnskey:         rumoured
  - zone rrsig:     rumoured

key: 10683 (ECDSAP256SHA256), KSK
  published:      yes - since Sat Apr 25 14:54:54 2026
  key signing:    yes - since Sat Apr 25 14:54:54 2026

  No rollover scheduled
  - goal:           omnipresent
  - dnskey:         rumoured
  - ds:             hidden
  - key rrsig:      rumoured

 etc/bind  41ms  
 14:57:45
```

Probemos a hacer una consulta desde el DNS recursivo que hemos configurado anteriormente:

```
 ~  17ms  
 14:59:01  dig @172.16.1.1 ns-testing.com +dnssec

; <<>> DiG 9.20.21-1~deb13u1-Debian <<>> @172.16.1.1 ns-testing.com +dnssec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44228
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 0bica7c2cb8a8e350100000069ecba9abda20ea484f571a2 (good)
;; QUESTION SECTION:
;ns-testing.com.                        IN      A

;; ANSWER SECTION:
ns-testing.com.         86400   IN      A       172.16.1.1
ns-testing.com.         86400   IN      RRSIG   A 13 2 86400 20260509090011 20260425115454 32224 ns-testing.com. FngqWC2gKGZWGpm1wxI1u+/fUszkeXH3vzDftZ0lKJDkdDGO4cL1fhPr bQOe2asMbNY/+dRGro
Xy9HmW1H+cUg==

;; Query time: 0 msec
;; SERVER: 172.16.1.1#53(172.16.1.1) (UDP)
;; WHEN: Sat Apr 25 14:59:05 CEST 2026
;; MSG SIZE  rcvd: 197

 ~  39ms  
 14:59:05
```

Un mini cambio que debemos hacer es en el DNS recursivo habilitar el `dnssec-validation auto;`

```
acl "trusted" {
        127.0.0.0/8;
        172.16.1.0/24;
};

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

        recursion yes;
        allow-query { trusted; };
        allow-recursion { trusted; };

        dnssec-validation auto;

        listen-on-v6 { any; };

        forwarders {
                8.8.8.8;
                1.1.1.1;
        };

        forward first;
};
```

1.7.1 Configuración DS en servidor maestro

Con el comando que se ve en la foto `dnssec-dsfromkey Kns-testing.com.+013+10683.key` generamos el registro DS que es un resumen o hash de mi clave pública. Sirve para enlazar la seguridad del padre con el hijo. El DS vive en el servidor padre y la DNSKEY vive en mi servidor (hijo), cuando el recursivo pide a mi dominio, el padre le da el DS. El recursivo entonces pide mi DNSKEY y le hace un hash, si coinciden, sabe que nadie ha suplantado mi identidad.

Para la simulación como no tenemos acceso real al servidor DNS de la generalitat o de la ICANN para subir este DS, simplemente en un entorno real se enviaría el DS al registrador para que lo publique en la zona padre.

```
 etc/bind  9ms
15:07:38  z /var/cache/bind
 var/cache/bind  9ms
15:07:46  ls
Kns-testing.com.+013+10683.key      Kns-testing.com.+013+10683.state  Kns-testing.com.+013+32224.private  managed-keys.bind
Kns-testing.com.+013+10683.private  Kns-testing.com.+013+32224.key    Kns-testing.com.+013+32224.state    managed-keys.bind.jnl
 var/cache/bind  11ms
15:07:47  dnssec-dsfromkey Kns-testing.com.+013+10683.key
ns-testing.com. IN DS 10683 13 2 D0519E921FE979CF36928EA98671C6B17AEFD1CE5B218FE68ACD1ECE93EC3736
 var/cache/bind  23ms
15:08:07
```

1.7.2 Verificaciones adicionales

```
 etc/bind  1m 28.929s
15:05:04  dig @172.16.1.1 ns-testing.com DNSKEY +multi

; <<>> DiG 9.20.21-1~deb13u1-Debian <<>> @172.16.1.1 ns-testing.com DNSKEY +multi
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22922
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 5bcbeb620bfd79260100000069ecbdba47e1a41e00454793 (good)
;; QUESTION SECTION:
;ns-testing.com.                IN DNSKEY

;; ANSWER SECTION:
ns-testing.com.         3600 IN DNSKEY 256 3 13 (
                                yHSLCW1HSrVdH0mGW+4yV4ye+xIuxouw/fhzHgho4bkd
                                TJ50KlisVfYsquUrQUj+l6dk3IwVIBEqCdnzFX4yJQ==
                                ) ; ZSK; alg = ECDSAP256SHA256 ; key id = 32224
ns-testing.com.         3600 IN DNSKEY 257 3 13 (
                                PQQN7rttAg1OPBB1zciMWD0NBmTTypkY0TLY661hE4Wg
                                etfbFId/Lf/rtDkHfiHcN6RcUeF1SJ4IOj3OwtOYog==
                                ) ; KSK; alg = ECDSAP256SHA256 ; key id = 10683

;; Query time: 0 msec
;; SERVER: 172.16.1.1#53(172.16.1.1) (UDP)
;; WHEN: Sat Apr 25 15:12:26 CEST 2026
;; MSG SIZE  rcvd: 231

 etc/bind  44ms
15:12:27
```

```
 etc/bind  44ms
15:12:27  dig @172.16.1.1 ns-testing.com A +dnssec

; <<>> DiG 9.20.21-1~deb13u1-Debian <<>> @172.16.1.1 ns-testing.com A +dnssec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57291
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 236631b06ae27fe60100000069ecbe22c8441ab80e88e5e7 (good)
;; QUESTION SECTION:
;ns-testing.com.                IN      A

;; ANSWER SECTION:
ns-testing.com.         86400   IN      A       172.16.1.1
ns-testing.com.         86400   IN      RRSIG   A 13 2 86400 20260509090011 20260425115454 32224 ns-testing.com. FngqWC2gKGZWGpm1wxI1u+/fUszkeXH3vzDftZ0lKJDkdDGO4cL1fhPr bQOe2asMbNY/+dRGrc
Xy9HmW1H+cUg==

;; Query time: 0 msec
;; SERVER: 172.16.1.1#53(172.16.1.1) (UDP)
;; WHEN: Sat Apr 25 15:14:10 CEST 2026
;; MSG SIZE  rcvd: 197

 etc/bind  24ms
15:14:10
```

2 Apartado teórico

2.1 ¿Qué es la cadena de confianza?

La cadena de confianza o chain of trust es el mecanismo central de DNSSEC, el cual constituye una jerarquía criptográfica que permite a un resolvedor validador verificar la autenticidad de cualquier registro DNS remontándose desde la respuesta hasta un trust anchor, normalmente la clave pública de la zona raíz, de forma análoga a como funciona la cadena de certificados X.509 en TLS/HTTPS

2.1.1 Nuevos tipos de registro DNS

DNSSEC introduce cuatro tipos de registros de recursos o RR fundamentales, definidos en el RFC 4034:

1. DNSKEY: Almacena la clave pública usada para verificar firmas digitales dentro de una zona. Coexisten dos subtipos funcionales:
   - Zone signing key (ZSK): Clave de vida corta que firma todos los RRset de la zona. Al ser más corta, permite rotaciones frecuentes sin interacción con la zona padre.
   - Key signing key (KSK): Clave de vida larga que firma únicamente el RRset DNSKEY. Su hash se publica en la zona padre como registro DS, anclando la cadena de confianza.
2. RRSIG (Resource Record Signature): Contiene la firma digital de un RRset concreto. Incluye el algoritmo de firma (por ejemplo RSA/SHA-256, ECDSA P-256), la fecha de inicio y expiración de la firma, el nombre del propietario y el identificador o key tag de la clave que la generó.
3. DS (Delegation Signer): Publicado en la zona padre, contiene el hash criptográfico de la KSK de la zona hija. Actúa como el eslabón que une dos niveles jerárquicos de la cadena de confianza. El RFC 4034 especifica el uso de SHA-1 (algoritmo 1) y el SHA-256 (algoritmo 2), siendo este el último recomendado en la actualidad.
4. NSEC/NSEC3 (Next Secure): Mecanismos de prueba de no existencia autenticada. NSEC crea una lista enlazada ordenada de nombres en la zona, NSEC3 hace lo propio pero usando hashes de los nombres para dificultar la enumeración.

2.1.2 Funcionamiento del proceso de validación

Cuando un resolvedor validador recibe una consulta DNS para, por ejemplo, www.ejemplo.com, el proceso de validación sigue estos pasos desde la hoja hasta la raíz:

1. El resolvedor obtiene el RRset A (o AAAA en caso de IPv6) junto con su RRSIG correspondiente para www.ejemplo.com desde el servidor autoritativo de la zona ejemplo.com
2. Solicita el DNSKEY RRset de ejemplo.com, localiza la ZSK indicada en el RRSIG por su key tag y verifica la firma del RRset A. Sí la firma es válida y la clave es de confianza, el registro puede ser considerado auténtico.
3. Para validar el DNSKEY RRset de ejemplo.com, verifica que esta firmado por la KSK, a continuación el registro DS de ejemplo.com en la zona padre (.com), calcula el hash de la KSK de ejemplo.com y lo compara con el DS publicado en .com.

4. El proceso se repite recursivamente hacia la raíz: valida el DNSKEY de .com con su DS en la zona raíz y finalmente valida la zona raíz usando el trust anchor precargado en el resolvedor.

Este proceso forma la cadena completa de delegación criptográfica. El RFC 4033 define explícitamente el concepto de security-aware resolver como aquel capaz de realizar esta validación, distinguiéndolo del resolvedor clásico, que acepta respuestas sin verificación.

2.2 Ventajas y desventajas de DNSSEC

2.2.1 Ventajas

2.2.1.1 Autenticación de origen e integridad de datos

La ventaja cardinal de **dnssec** es la provisión de autenticación criptográfica de origen e integridad de datos para las respuestas **dns**. Un resolvedor validador puede determinar con certeza matemática que un **rrset** fue generado por el administrador legítimo de la zona y que no ha sido alterado en tránsito. Esto elimina la superficie de ataque de las amenazas clásicas:

- **Cache poisoning** (ataque Kaminsky): un atacante que inyecte respuestas falseadas en la caché de un resolvedor no podrá producir un **rrsig** válido sin tener acceso a la clave privada **zsk** del propietario legítimo de la zona. El intento de falsificación será detectado y la respuesta rechazada.
- **Man-in-the-middle** contra el flujo **dns**: cualquier modificación de los datos **dns** en tránsito invalidará la firma **rrsig**, lo que provocará que el paquete sea descartado por el resolvedor.

2.2.1.2 Prueba de no existencia autenticada

dnssec resuelve un problema que el **dns** clásico no puede abordar: cómo demostrar criptográficamente que un nombre de dominio o tipo de registro no existe. Sin esta capacidad, un atacante podría interceptar la respuesta **nxdomain** legítima y sustituirla por una respuesta positiva falsa. Los registros **nsec** y **nsec3**, firmados digitalmente, permiten al resolvedor verificar que el dominio consultado efectivamente no se encuentra en la zona.

2.2.1.3 Habilitación de dane (dns-based authentication of named entities)

dnssec es un requisito previo para **dane** (**rfc 6698**), un protocolo que permite publicar certificados **tls** o sus hashes directamente en el **dns** mediante registros **tlsa**. Esto crea un mecanismo de autenticación alternativo y complementario a la infraestructura de clave pública (**pki**) basada en autoridades certificadoras (**ca**), lo que elimina el punto único de fallo que representan las **ca** comprometidas. **dane** es especialmente relevante para la seguridad del correo electrónico (**smtp** con **starttls**), donde el modelo actual de certificadoras es particularmente débil.

2.2.1.4 Fundamento para protocolos de seguridad superiores

dnssec constituye la infraestructura de confianza sobre la que se construyen otros protocolos de seguridad. Entre ellos destaca el **rfc 9102** (**tls dnssec chain extension**), que define cómo un servidor **tls** puede incluir la cadena de autenticación **dnssec** completa en el saludo o **handshake**, permitiendo al cliente verificar el certificado sin necesidad de realizar consultas

dns adicionales. Asimismo, iniciativas como **sshfp** (**rfc 4255**) permiten publicar huellas de claves **ssh** en el **dns** con autenticación **dnssec**.

2.2.2 Desventajas y limitaciones técnicas

2.2.2.1 dnssec como vector de amplificación ddos

La inclusión obligatoria de firmas **rrsig** junto a cada respuesta **dns** aumenta significativamente el tamaño de los paquetes de respuesta. Este diferencial de tamaño puede ser explotado en ataques de reflexión y amplificación **ddos**, donde el atacante envía consultas con una dirección ip de origen falsificada (**spoofed**) para que las respuestas ampliadas sean redirigidas hacia la víctima.

El estudio de van Rijswijk-Deij et al. (**imc 2014, acm**) midió el factor de amplificación en 2,5 millones de zonas firmadas. Sus conclusiones son contundentes: una consulta **nxdomain** a una zona firmada con **rsa/sha-1** de 1.024 bits y **nsec3** puede producir un factor de amplificación de hasta 10 veces respecto al tamaño de la consulta. Las consultas **any** resultan todavía más eficaces como vector de amplificación. La consulta a un **tld** inexistente en la zona raíz tiene un factor de aproximadamente 6,5 veces.

La adopción de **ecdsa** (algoritmo 13) mitiga parcialmente este problema, dado que los tamaños de clave y firma son significativamente menores: conseguir una seguridad de 128 bits con **ecdsa** requiere claves de 256 bits frente a los 3.072 bits del equivalente en **rsa**.

2.2.2.2 Enumeración de zona (zone walking)

El mecanismo **nsec** original, al publicar una lista enlazada y ordenada de todos los nombres existentes en la zona, permite a cualquier tercero enumerar todos los registros con solo iterar las respuestas negativas, una técnica denominada **zone walking**. Para una zona con **r** nombres, se necesitan aproximadamente **r** consultas para enumerarla completamente.

nsec3 fue introducido para mitigar este problema sustituyendo los nombres en texto plano por sus hashes **sha-1**. Sin embargo, investigadores del **mit** y **bu** (Goldberg et al., **ndss 2015**) demostraron formalmente que **nsec3** sigue siendo vulnerable: mediante ataques de diccionario fuera de línea sobre los hashes publicados, un atacante puede recuperar los nombres originales. La misma investigación probó que es matemáticamente imposible proteger simultáneamente contra la enumeración de zona y contra atacantes de red sin realizar operaciones criptográficas en línea en el servidor.

La propuesta académica **nsec5** utiliza funciones aleatorias verificables (**vrf**) para resolver este problema teórico: incluso si la clave **nsec5key** es comprometida, el atacante solo degrada la seguridad a la de **nsec3**, sin poder falsificar registros. **nsec5** aún no está estandarizado en un **rfc** definitivo.

2.2.2.3 Ausencia de confidencialidad

dnssec proporciona autenticación e integridad, pero no confidencialidad. Las consultas y respuestas, incluidas las firmadas, se transmiten en texto claro. Un observador pasivo puede conocer qué dominios consulta un usuario. Esta limitación es abordada por protocolos com-

plementarios como **dns over tls** (**dot**) y **dns over https** (**doh**), que no son incompatibles con **dnssec** sino complementarios.

2.2.2.4 Vulnerabilidades emergentes en la interacción de estándares

Investigaciones recientes publicadas en diciembre de 2025 han aplicado modelos formales de verificación al protocolo, descubriendo que la coexistencia de registros **nsec** y **nsec3** en la misma zona crea una brecha de autenticación (**authentication gap**). Un resolvedor puede ser engañado para que acepte una prueba de no existencia basada en **nsec3** que omite dominios existentes cubiertos por registros **nsec**. Este no es un fallo de implementación sino una consecuencia de la especificación del protocolo, y afecta a sistemas como **bind**, **unbound**, **powerdns** y **knot**.

2.2.2.5 La vulnerabilidad keytrap (cve-2023-50387)

Descubierta en 2023 y divulgada en 2024, **keytrap** demostró que un único paquete **dns** maliciosamente construido podía agotar los recursos de cpu de un resolvedor validador, causando una denegación de servicio efectiva. La vulnerabilidad explota el comportamiento del resolvedor al validar múltiples firmas y claves en respuestas intencionalmente complejas. Afectó a prácticamente todas las implementaciones conocidas de resolvedores **dnssec**.

2.2.2.6 Complejidad operacional y adopción baja

Estudios de **apnic** revelan que la adopción de **dnssec** entre dominios **.com**, **.net** y **.org** es de aproximadamente el 1%, y que más del 30% de los dominios que sí lo tienen configurado presentan errores activos. La complejidad del protocolo es una barrera significativa para su adopción masiva.