Sessão 2: Firewall e DNS

O ponto de partida para a segurança da rede em qualquer organização é, certamente, o *firewall*. Utilizando-se do princípio de segurança do *chokepoint*, ou gargalo, tem como premissa atuar como um ponto focal no controle do tráfego de rede, permitindo aos administradores configurar acessos e proibições de acordo com as políticas organizacionais. Conceitualmente, o firewall consiste em um filtro de pacotes e tem sua atuação restrita até a camada 4 (transporte) do modelo OSI, ficando a tarefa de filtragem em nível de aplicação para outros programas e *appliances* de controle da rede, como IDSs (*Intrusion Detection System*), IPSs (*Intrusion Protection System*), *proxies* ou WAFs (*Web Application Firewalls*).

Juntamente com o firewall, o serviço de resolução de nomes (DNS, ou *Domain Name System*) também é peça-chave na segurança da rede. Provendo um serviço essencial — tradução de endereços de rede para nomes e vice-versa — o serviço DNS deve ser cuidadosamente configurado pelos administradores para evitar ataques de *spoofing* e prover redundância em caso de falhas inesperadas. Os servidores DNS se dividem em primário, secundário e recursivo; destes, os dois primeiros são responsáveis por responder por um domínio de rede (ditos **autoritativos**, portanto), e o último apenas funciona como uma fonte de consulta e *cache* para os clientes da rede.

Neste curso, faremos uma configuração extremamente restritiva do firewall de nosso *datacenter* simulado, e a cada novo serviço implementado iremos retornar à sua configuração para abrir as portas e protocolos adequados. Já no caso do DNS, configuraremos um servidor primário/secundário autoritativo com suporte a DNSSEC para a rede interna `intnet.`, restringindo todas as máquinas a realizarem suas consultas exclusivamente nesses servidores. Iremos, ainda, configurar um servidor DNS recursivo em um *daemon* distinto exclusivamente para consultas e *cache* de resultados, visando o aumento da segurança do sistema de resolução de nomes.

1) Topologia desta sessão

A figura abaixo mostra a topologia de rede que será utilizada nesta sessão, com as máquinas relevantes em destaque.

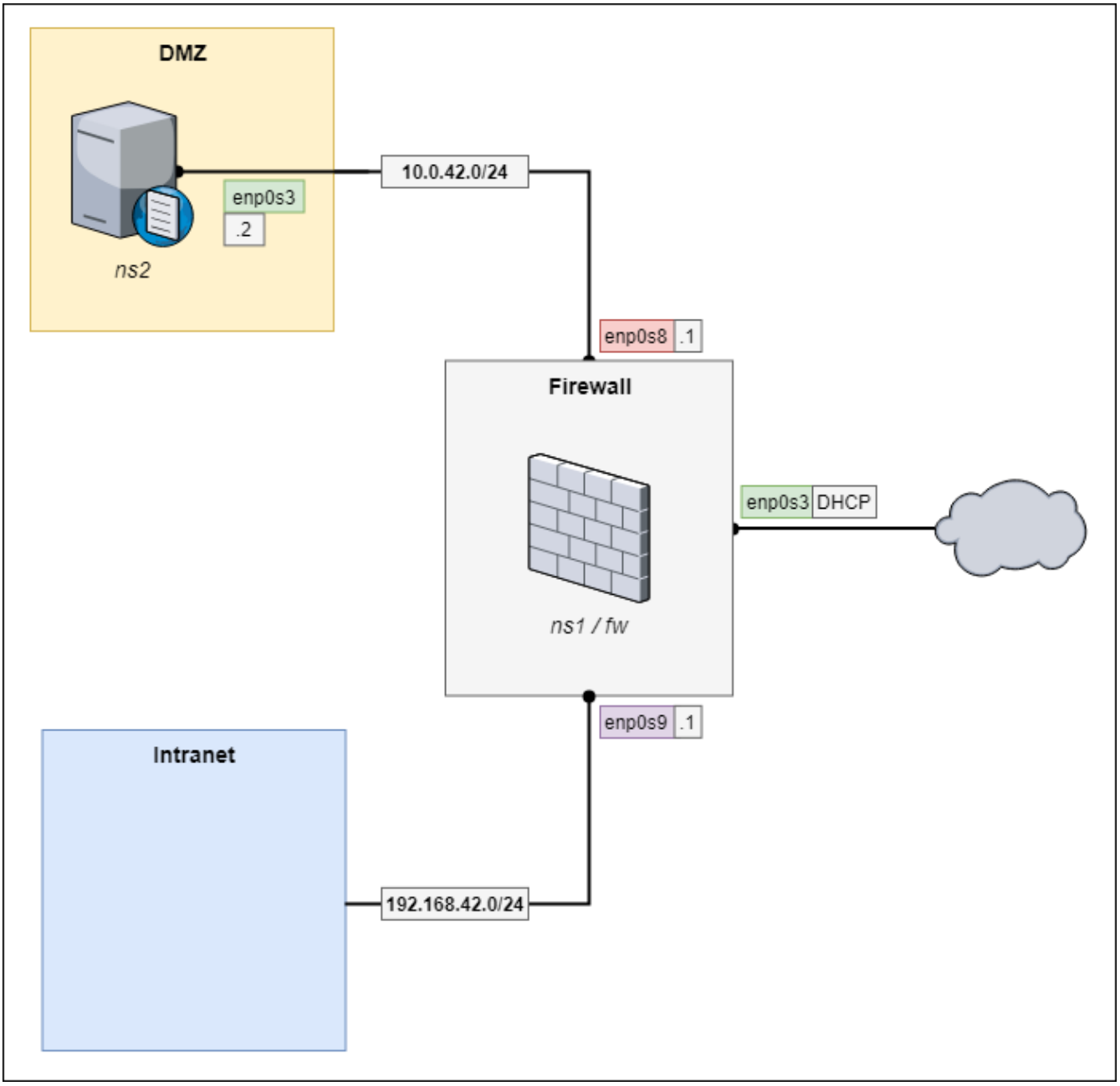

Figura 1. Topologia de rede desta sessão

Teremos apenas duas máquinas, por enquanto:

- `ns1`, atuando como firewall de rede e DNS primário. Endereços IP via DHCP (interface *bridge*), 10.0.42.1/24 (interface DMZ) e 192.168.42.1/24 (interface Intranet).
- `ns2`, atuando como DNS secundário e localizada na DMZ (*Demilitarized Zone*, ou zona desmilitarizada). Endereço IP 10.0.42.2/24.

1. Antes de começarmos, precisamos configurar corretamente as redes virtuais no Virtualbox. Acesse o menu *File* > *Host Network Manager* e crie as seguintes redes:

Tabela 1. Redes host-only no Virtualbox

Rede	Endereço IPv4	Máscara de rede	Servidor DHCP
Virtualbox Host-Only Ethernet Adapter #2	10.0.42.254	255.255.255.0	Desabilitado

Rede	Endereço IPv4	Máscara de rede	Servidor DHCP
Virtualbox Host-Only Ethernet Adapter #3	192.168.42.254	255.255.255.0	Desabilitado

Visualmente, sua janela deve ficar parecida com o seguinte:

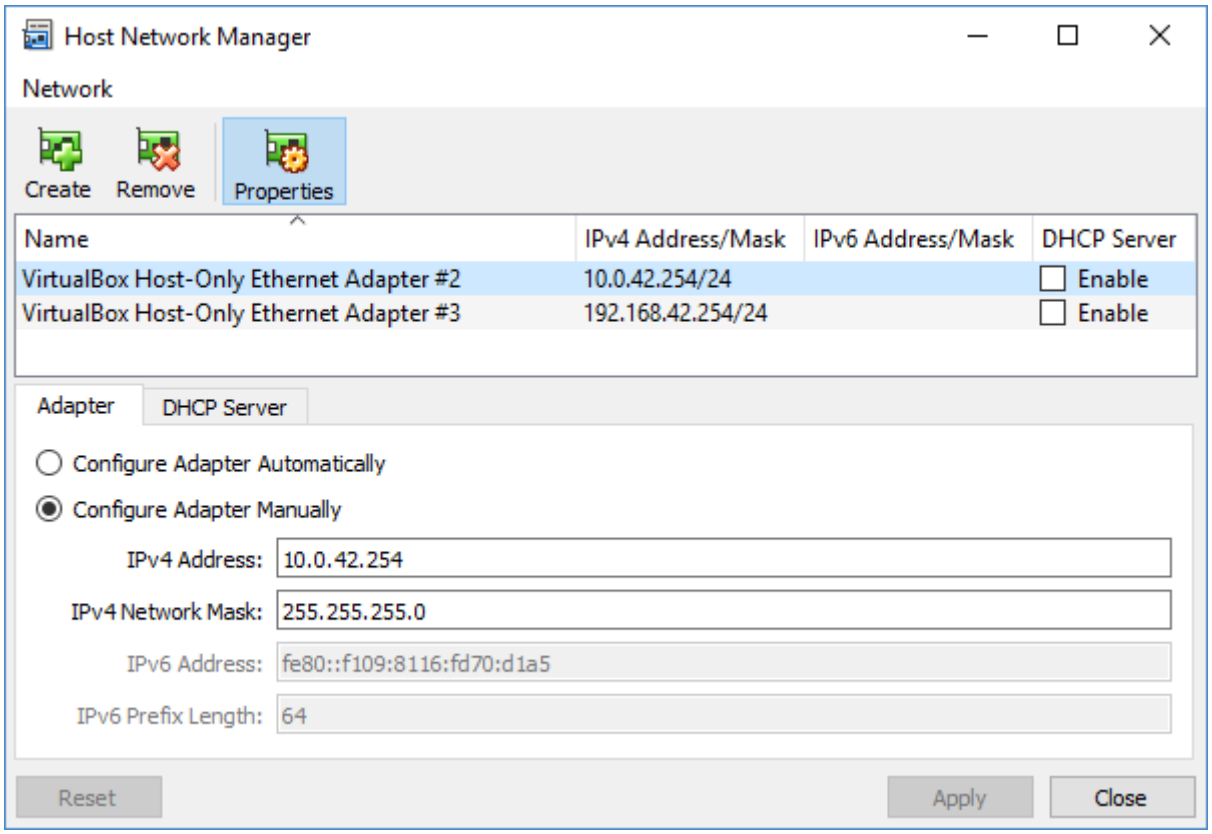

Figura 2. Redes host-only no Virtualbox

É possível que os números específicos das redes (`#2` para a DMZ e `#3` para a Intranet, na imagem acima) não fiquem exatamente iguais aos exemplificados. Nesse caso, faça uma anotação indicando qual número de rede *host-only* corresponde a cada uma das redes configuradas. Verifique, ainda, que o servidor DHCP interno do Virtualbox está desabilitado em ambas as redes.

2. As configurações de rede realizadas internamente em cada máquina virtual foram apresentados de forma sucinta na topologia desta sessão. Iremos detalhar as configurações logo abaixo:

Tabela 2. Configurações de rede de cada VM

VM Nome	Interface	Modo	Endereço	Gateway
ns1	enp0s3	DHCP	Automático	Automático
	enp0s8	Estático	10.0.42.1/24	n/a
	enp0s9	Estático	192.168.42.1/24	n/a
ns2	enp0s3	Estático	10.0.42.2/24	10.0.42.1

A partir do Debian 9, a nomenclatura padrão de interfaces de rede foi alterada. Ao invés de denotarmos as interfaces como `eth0`, `eth1` ou `eth2`, o `systemd/udev` utiliza, a partir da versão v197, um método de nomenclatura de interfaces usando *biosdevnames*, como documentado oficialmente em https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/ . Com efeito, esse novo sistema suporta cinco meios de nomeação de interfaces de rede:

1. Nomes incorporando números de índice providos pelo firmware/BIOS de dispositivos *on-board* (p.ex.: `eno1`)
2. Nomes incorporando números de índice providos pelo firmware/BIOS de encaixes *hotplug* PCI Express (p.ex.: `ens1`)
3. Nomes incorporando localização física/geográfica do conector do hardware (p.ex.: `enp2s0`)
4. Nomes incorporando o endereço MAC da interface (p.ex.: `enx78e7d1ea46da`)
5. Nomes clássicos, usando nomenclatura não-previsível nativa do kernel (p.ex.: `eth0`)

Todas as VMs utilizadas neste curso serão derivadas da máquina `debian-template`, configurada com o Debian 9. A equivalência da nova nomenclatura de interfaces de rede, se comparada com a antiga, ficaria assim:

Tabela 3. Nomenclatura de interfaces de máquinas Debian 9

Interface antiga	Interface nova
eth0	enp0s3
eth1	enp0s8
eth2	enp0s9

Observe, por exemplo, como é feita a detecção de interfaces durante o *boot* da máquina `ns1`, que criaremos a seguir:

```
# dmesg | grep 'renamed from'
[    1.658908] e1000 0000:00:09.0 enp0s9: renamed from eth2
[    1.659807] e1000 0000:00:08.0 enp0s8: renamed from eth1
[    1.660711] e1000 0000:00:03.0 enp0s3: renamed from eth0
```

2) Criação da VM de firewall e DNS primário

Iremos agora criar a primeira máquina virtual efetiva de nosso *datacenter* simulado, a máquina `ns1`. Essa máquina atuará como um firewall de borda e DNS primário da rede, como configuraremos a seguir. Por se tratar de um firewall, é necessário que ela possua ao menos duas (ou, em nosso caso específico, três) interfaces de rede interconectando redes distintas.

1. Clone a máquina `debian-template` seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha `ns1`.
2. Após a clonagem, na janela principal do Virtualbox, clique com o botão direito sobre a VM `ns1` e depois em *Settings*.

   Em *Network* > *Adapter 1* > *Attached to*, mantenha escolhida a opção *Bridged Adapter*, já que esta será a interface de conexão externa da máquina.

   Em *Adapter 2*, marque a caixa *Enable Network Adapter* e em *Attached to* selecione *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a **DMZ**, como indicado na atividade (1) desta sessão. Seguindo o exemplo mostrado na figura da atividade, a rede escolhida seria portanto a `Virtualbox Host-Only Ethernet Adapter #2`.

   Em *Adapter 3*, marque a caixa *Enable Network Adapter* e em *Attached to* selecione *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a **Intranet**, como indicado na atividade (1) desta sessão. Seguindo o exemplo da figura, escolheríamos então `Virtualbox Host-Only Ethernet Adapter #3`.

   Clique em *OK*, e ligue a máquina `ns1`.

3. Após o *boot*, faça login como o usuário `root`. Primeiro, vamos configurar a rede: edite o arquivo `/etc/network/interfaces` como se segue:

```
# nano /etc/network/interfaces
(...)
```

```
# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo enp0s3 enp0s8 enp0s9

iface lo inet loopback

iface enp0s3 inet dhcp

iface enp0s8 inet static
address 10.0.42.1/24

iface enp0s9 inet static
address 192.168.42.1/24
```

Para garantir que nenhum endereço IP antigo, primário, se mantenha alocado às interfaces, execute o comando `flush` e em seguida reinicie a rede do sistema:

```
# ip addr flush label 'enp0s*' ; systemctl restart networking
```

Verifique que as interfaces estão com os endereços corretamente alocados:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}'
192.168.29.104/24 enp0s3
10.0.42.1/24 enp0s8
192.168.42.1/24 enp0s9
```

4. Usando o script `/root/scripts/changehost.sh` que criamos anteriormente, renomeie a máquina:

```
# hostname
debian-template
```

```
# bash ~/scripts/changehost.sh ns1
```

```
# hostname
ns1
```

3) Configuração inicial do firewall

Para garantir a segurança da rede iremos configurar o firewall de forma extremamente restritiva, como se segue:

1. Tráfego oriundo do firewall (*chain* OUTPUT) será permitido.
2. Todo o tráfego na interface *loopback* será permitido.
3. Serão permitidos pacotes destinados ao firewall (*chain* INPUT) ou passando pelo firewall (*chain* FORWARD) cujo estado seja relacionado ou estabelecido.
4. Serão permitidos pacotes ICMP oriundos das redes DMZ e Intranet com destino ao firewall *FWGW1-G*.
5. Será permitida gerência via `ssh` do firewall a partir de máquinas da Intranet.
6. Será autorizado o tráfego na Internet das máquinas da DMZ e Intranet **exclusivamente** nas portas TCP/80 e TCP/443.
7. Todos os demais acessos serão bloqueados.

À medida que novas regras forem necessárias, nas sessões seguintes, iremos criar regras de exceção pontualmente durante a execução das atividades, explicando os motivos das liberações. Vamos,

ponto a ponto, realizar as configurações explicitadas acima:

1. Primeiro, como em qualquer firewall de rede, devemos habilitar o repasse de pacotes entre interfaces. Para isso, edite o arquivo `/etc/sysctl.conf` e descomente a linha com a diretiva `net.ipv4.ip_forward`, como se segue:

   ```
   # sed -i '/net.ipv4.ip_forward/s/^#//' /etc/sysctl.conf
   ```

   Processe as alterações no arquivo com:

   ```
   # sysctl -p
   net.ipv4.ip_forward = 1
   ```

   Observe que esse arquivo é lido durante o *boot* do sistema, o que garante que nossa configuração perdurará mesmo após reiniciarmos a máquina.

2. Agora, vamos retomar as diretivas informadas no início desta atividade: para a requisição (a) não precisamos fazer nada, já que a política da *chain* OUTPUT encontra-se em ACCEPT:

   ```
   # iptables -L OUTPUT
   Chain OUTPUT (policy ACCEPT)
   target     prot opt source               destination
   ```

3. A diretiva (b) pode ser atendida com a regra que se segue:

   ```
   # iptables -A INPUT -i lo -j ACCEPT
   ```

   Já tratamos do caso da *chain* OUTPUT, e não faz sentido falarmos em tráfego na interface *loopback* na *chain* FORWARD.

4. Para a diretiva (c) devemos usar uma regra de estados nas *chains* INPUT e FORWARD, da seguinte forma:

   ```
   # iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
   ```

   ```
   # iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
   ```

5. Como a diretiva (d) diz especificamente de pacotes "com destino ao firewall", fica claro que devemos adicionar uma regra à *chain* INPUT. Note ainda que a diretiva é precisa em especificar que **apenas** o tráfego ICMP das redes DMZ e Intranet deve ser autorizado, e nenhum outro. Finalmente, não é especificado qual tipo de pacote ICMP será aceito, o que nos permite deduzir que todos serão aceitos.

Para inserir uma regra que inclua ambas as redes de origem, basta separá-las com vírgula, como se segue:

```
# iptables -A INPUT -s 10.0.42.0/24,192.168.42.0/24 -p icmp -m icmp --icmp-type any
-j ACCEPT
```

6. A diretiva (e) é clara ao especificar que a gerência será via `ssh` (portanto, na porta 22 do protocolo TCP), a ser feita no firewall (ou seja, *chain* INPUT), e apenas a partir de máquinas da Intranet. Podemos atender a esse requisito com a seguinte regra:

   ```
   # iptables -A INPUT -s 192.168.42.0/24 -p tcp -m tcp --dport 22 -j ACCEPT
   ```

7. A diretiva (f) diz que o tráfego na Internet das máquinas da DMZ e Intranet deve ser autorizado apenas nas portas TCP/80 e TCP/443. O requisito de IP de origem é bastante claro, mas o de destino não — de fato, máquinas na Internet podem ter, a princípio, qualquer endereço IP. Faz sentido, então, indicarmos a interface de saída dos pacotes, `enp0s3`.

   Outro aspecto a ser observado é que os pacotes desta vez não se destinam ao firewall, mas sim passam por ele para atingir máquinas na Internet, indicando que a regra deve ser inserida na *chain* FORWARD.

   Podemos ainda usar o módulo `multiport` para evitar a digitação de duas regras similares. Então, execute:

   ```
   # iptables -A FORWARD -s 10.0.42.0/24,192.168.42.0/24 -o enp0s3 -p tcp -m multiport
   --dports 80,443 -j ACCEPT
   ```

   Há ainda que se considerar a necessidade de realizar a tradução dos endereços de saída, pois não é possível que as máquinas da DMZ/Intranet naveguem com seus IPs em faixas privadas. Temos que criar uma regra de SNAT para permitir a navegação — levando em conta que o endereço da interface `enp0s3` é dinâmico, faz sentido usar o alvo MASQUERADE, nesse caso.

   Um adendo final: podemos fazer uma regra tão restritiva quanto a que fizemos na *chain* FORWARD acima, especificando também o protocolo e porta em que o SNAT será realizado. Tenha em mente apenas que, em caso de adição de exceções futuras, será necessário adicionar o protocolo/porta de exceção em **ambas** as *chains*, `filter/FORWARD` e `nat/POSTROUTING`.

   A regra fica assim:

   ```
   # iptables -t nat -A POSTROUTING -s 10.0.42.0/24,192.168.42.0/24 -o enp0s3 -p tcp
   -m multiport --dports 80,443 -j MASQUERADE
   ```

8. Atender a diretiva (g) final é bastante fácil: basta alterar a política das *chains* INPUT e FORWARD para DROP:

```
# iptables -P INPUT DROP
```

```
# iptables -P FORWARD DROP
```

9. Verifique a configuração final do firewall, comparando com os requisitos iniciais. Consulte primeiro a tabela *filter*:

```
# iptables -L -vn
Chain INPUT (policy DROP 189 packets, 52988 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
  982 67024 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       10.0.42.0/24         0.0.0.0/0
icmptype 255
    0     0 ACCEPT     icmp --  *      *       192.168.42.0/24      0.0.0.0/0
icmptype 255
    0     0 ACCEPT     tcp  --  *      *       192.168.42.0/24      0.0.0.0/0
tcp dpt:22

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      enp0s3  10.0.42.0/24         0.0.0.0/0
multiport dports 80,443
    0     0 ACCEPT     tcp  --  *      enp0s3  192.168.42.0/24      0.0.0.0/0
multiport dports 80,443

Chain OUTPUT (policy ACCEPT 16 packets, 1264 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

E, depois, a tabela *nat*:

```
# iptables -L -vn -t nat
Chain PREROUTING (policy ACCEPT 10 packets, 1485 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 6 packets, 1012 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  tcp  --  *      enp0s3  10.0.42.0/24         0.0.0.0/0
multiport dports 80,443
    0     0 MASQUERADE  tcp  --  *      enp0s3  192.168.42.0/24      0.0.0.0/0
multiport dports 80,443
```

10. As configurações realizadas até aqui estão todas em memória — em caso de *reboot* da máquina `ns1`, elas serão perdidas. Para gravar as regras e integrá-las ao sistema de *init* do SO, o pacote `iptables-persistent` é uma excelente opção. Instale-o com:

```
# apt-get install iptables-persistent
```

Na instalação do pacote, quando perguntado, responda:

Tabela 4. Configurações do iptables-persistent

Pergunta	Resposta
Salvar as regras IPv4 atuais?	Sim
Salvar as regras IPv6 atuais?	Sim

As regras IPv4 e IPv6 serão gravadas nos arquivos `/etc/iptables/rules.v4` e `/etc/iptables/rules.v6`, respectivamente. Em caso de alterações futuras nas configurações do firewall, é possível gravá-las de forma fácil com o comando:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

Se quiser limpar todas as regras de firewall e começar do zero, execute:

```
# /etc/init.d/netfilter-persistent flush
[....] Flushing netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables flush
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables flush
done.
```

```
# iptables -L -vn
Chain INPUT (policy ACCEPT 13 packets, 1558 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 11 packets, 1068 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Se cometer qualquer erro durante o processo de configuração ou simplesmente quiser recarregar o conjunto de regras gravado em `/etc/iptables/rules.v4`, execute:

```
# /etc/init.d/netfilter-persistent reload
[....] Loading netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables start
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables start
done.
```

```
# iptables -L -vn
Chain INPUT (policy DROP 70 packets, 19740 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
  111  5861 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       10.0.42.0/24         0.0.0.0/0
icmptype 255
    0     0 ACCEPT     icmp --  *      *       192.168.42.0/24      0.0.0.0/0
icmptype 255
    0     0 ACCEPT     tcp  --  *      *       192.168.42.0/24      0.0.0.0/0
tcp dpt:22

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      enp0s3  10.0.42.0/24         0.0.0.0/0
multiport dports 80,443
    0     0 ACCEPT     tcp  --  *      enp0s3  192.168.42.0/24      0.0.0.0/0
multiport dports 80,443

Chain OUTPUT (policy ACCEPT 120 packets, 12617 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

4) Configuração do servidor DNS primário

Iremos agora configurar a máquina `ns1` como o servidor DNS primário da rede; para tanto, usaremos os programas NSD e Unbound.

O NSD (*Name Server Daemon*) é um servidor DNS *open-source* desenvolvido pela NLNet de Amsterdã em parceria com a RIPE NCC, o registro regional de Internet da Europa, oeste da Ásia e ex-países da União Soviética. Ele foi projetado para atuar exclusivamente como um servidor DNS autoritativo, não implementando portanto funções de *cache* recursiva como sua contraparte mais famosa, o BIND. A ideia por trás disso é aumentar a variabilidade de implementações DNS disponíveis, aumentando a resiliência do sistema DNS contra falhas de software e *exploits*.

O Unbound, por outro lado, é um *resolver* DNS com funções de validação, *cache* e recursividade também implementado pela NLNet Labs. Como se pode observar, então, ele faz as funções complementares às do NSD. Por sua percepção como sendo um software mais moderno, enxuto e seguro que seus concorrentes, o Unbound foi adotado como o *resolver* padrão do sistema-base dos sistemas FreeBSD e OpenBSD em 2014.

1. O primeiro passo, naturalmente, é instalar os pacotes dos programas mencionados:

```
# apt-get install nsd unbound dnsutils
```

2. Vamos começar configurando o **NSD**. Primeiro, pare o *daemon*:

```
# systemctl stop nsd
```

O programa pode ser controlado no terminal usando o comando `nsd-control` — para tanto, é necessário gerar as chaves TLS de controle com o comando:

```
# nsd-control-setup
setup in directory /etc/nsd
nsd_server.key exists
nsd_control.key exists
create nsd_server.pem (self signed certificate)
create nsd_control.pem (signed client certificate)
Signature ok
subject=CN = nsd-control
Getting CA Private Key
Setup success. Certificates created.
```

Verifique a criação das chaves com o comando:

```
# ls -1 /etc/nsd | grep '.key\|.pem'
nsd_control.key
nsd_control.pem
nsd_server.key
nsd_server.pem
```

3. Faça um backup do arquivo de configuração original `/etc/nsd/nsd.conf`:

```
# mv /etc/nsd/nsd.conf /etc/nsd/nsd.conf.orig
```

Em seguida, crie o arquivo novo `/etc/nsd/nsd.conf` com o seguinte conteúdo:

```
server:
  ip-address: 127.0.0.1
  ip-address: 10.0.42.1
  do-ip4: yes
  port: 8053
  username: nsd
  zonesdir: "/etc/nsd/zones"

  logfile: "/var/log/nsd.log"
  pidfile: "/run/nsd/nsd.pid"
  hide-version: yes
  version: "intnet DNS"
  identity: "unidentified server"

remote-control:
  control-enable: yes
  control-interface: 127.0.0.1
  control-port: 8952
  server-key-file: "/etc/nsd/nsd_server.key"
  server-cert-file: "/etc/nsd/nsd_server.pem"
  control-key-file: "/etc/nsd/nsd_control.key"
  control-cert-file: "/etc/nsd/nsd_control.pem"

key:
  name: "inkey"
  algorithm: sha512
  secret: "TSIGKEY"

pattern:
  name: "inslave"
  notify: 10.0.42.2@8053 inkey
  provide-xfr: 10.0.42.2 inkey

zone:
  name: "intnet"
  include-pattern: "inslave"
  zonefile: "intnet.zone"

zone:
  name: "42.0.10.in-addr.arpa"
  include-pattern: "inslave"
  zonefile: "10.0.42.zone"
```

Abaixo, destacamos e explicamos algumas das diretivas mais relevantes do arquivo acima:

- `server/ip-address`: define os endereços de rede nos quais o NSD irá escutar — atenderemos requisições de *localhost* e do servidor DNS secundário, que será instalado na DMZ a seguir. O controle de quais *hosts* estarão autorizados a consultar o NSD será feito via diretivas `notify` e `provide-xfr`, bem como restrição via regra de firewall.

- `server/port`: define a porta em que o NSD irá escutar; como iremos instalar o Unbound na mesma máquina (escutando na porta 53/UDP), escolhemos a porta alternativa 8053 para evitar conflitos de *bind*.
- `server/username`: usuário com o qual o NSD irá operar. É possível aplicar um controle adicional além da redução de privilégios para um usuário comum, o uso de *chroot*, que não faremos nesta atividade.
- `server/zonesdir`: diretório em que serão armazenados os arquivos de zonas do NSD.
- `server/hide-version`, `server/version` e `server/identity`: os três controles objetivam mascarar o software rodando na máquina local, dificultando a tarefa de *banner grabbing* e identificação de *exploits* por parte de eventuais atacantes.
- `remote-control/control-enable` e `remote-control/interface`: define que o NSD poderá ser controlado "remotamente", mas logo depois restringe o IP de escuta para 127.0.0.1, efetivamente autorizando conexão de controle exclusivamente a partir de *localhost*.
- `key/secret`: define uma chave secreta que servidores primário e secundário deverão possuir em comum para que a transferência de zona seja feita com sucesso. Faremos a geração dessa chave e sua substituição de forma automática no arquivo a seguir.
- `pattern`: define um conjunto de diretivas que serão inseridas em diversas zonas a seguir (efetivamente, como uma *macro*). No caso, estamos indicando que o servidor 10.0.42.2 será notificado de alterações de zona (`notify`), e transferências de zona partir desse endereço serão autorizadas (`provide-xfr`).
- `zone/intnet`: define um arquivo de zona direta para o qual o servidor NSD será autoritativo, o nome de domínio `intnet`. A sintaxe do arquivo é idêntica à usada no software BIND.
- `zone/42.0.10-in-addr.arpa`: define um arquivo de zona reversa para o qual o servidor NSD será autoritativo, a faixa de endereços `10.0.42.0/24`. A sintaxe do arquivo é idêntica à usada no software BIND.

A transferência de zonas entre servidor primário e secundário é assegurada, além dos controles de interface de escuta e regras de firewall, por uma chave secreta TSIG (*Transaction SIGnature*) em formato Base64. Para inseri-la no arquivo use o comando:

```
# tsigkey_t=$( dd if=/dev/urandom of=/dev/stdout count=1 bs=32 2> /dev/null| base64
); sed -i "s|TSIGKEY|$tsigkey_t|" /etc/nsd/nsd.conf ; unset tsigkey_t
```

Verifique a inserção da chave com:

```
# grep 'secret:' /etc/nsd/nsd.conf
  secret: "i7IoB5VDHVOCW9wvuOGQuFLNu8hzfAblVAbCD1SbPL4="
```

Lembre-se que precisaremos dessa mesma chave quando estivermos configurando o servidor secundário.

4. Crie o diretório de zonas `/etc/nsd/zones`:

```
# mkdir /etc/nsd/zones
```

Agora, crie o arquivo novo `/etc/nsd/zones/intnet.zone`, com as configuração de zona direta para o domínio `intnet.`, com o seguinte conteúdo:

```
$TTL 86400 ; (1 day)
$ORIGIN intnet.

@       IN   SOA   ns1.intnet.  admin.intnet. (
                   2018111000   ;serial (YYYYMMDDnn)
                   14400        ;refresh (4 hours)
                   1800         ;retry (30 minutes)
                   1209600      ;expire (2 weeks)
                   3600         ;negative cache TTL (1 hour)
                   )

@       IN   NS                 ns1.intnet.
@       IN   NS                 ns2.intnet.

@       IN   MX    10           mx1.intnet.
@       IN   MX    20           mx2.intnet.

ns1     IN    A                 10.0.42.1
ns2     IN    A                 10.0.42.2

mx1     IN    A                 10.0.42.91
mx2     IN    A                 10.0.42.92

fw      IN    CNAME             ns1
```

A sintaxe é exatamente a mesma utilizada para um arquivo de zonas do BIND. Note que:

- Definimos a máquina `ns1.intnet.` como o servidor autoritativo para o domínio `intnet.`, com email de contato `admin@intnet`.
- O serial é `2018111000` — este é um número que deve identificar a versão do arquivo de zonas, e incrementado a cada atualização. O formato escolhido para esse *serial* foi `ANO-MES-DIA-VERSAO`, de forma que versões do arquivo em datas futuras terão sempre um valor superior ao de versões antigas.
- Os servidores DNS responsáveis pelo domínio são `ns1.intnet` e `ns2.intnet.`.
- Os servidores de e-mail (MX — *mail exchange*) do domínio são `mx1.intnet.` e `mx2.intnet.`. Ambos são servidores fictícios, que não implantaremos neste curso, e mostrados aqui apenas para demonstrar a sintaxe desse tipo de entrada no arquivo de zonas.
- Configuram-se registros A (*address*) para as máquinas mencionadas.
- Configura-se um registro CNAME (*canonical name*) para a máquina `ns1.intnet.`, `fw.intnet.`. Com isso, os usuário poderão referir-se a essa máquina também pelo seu "apelido".

5. Vamos para o arquivo de zona reversa. Crie o arquivo novo `/etc/nsd/zones/10.0.42.zone` com o seguinte conteúdo:

```
$TTL 86400 ; (1 day)
$ORIGIN 42.0.10.in-addr.arpa.

@       IN   SOA   ns1.intnet.  admin.intnet. (
                   2018111000   ;serial (YYYYMMDDnn)
                   14400        ;refresh (4 hours)
                   1800         ;retry (30 minutes)
                   1209600      ;expire (2 weeks)
                   3600         ;negative cache TTL (1 hour)
                   )

@       IN   NS                 ns1.intnet.
@       IN   NS                 ns2.intnet.

@       IN   MX    10           mx1.intnet.
@       IN   MX    20           mx2.intnet.

1       IN   PTR                ns1.intnet.
2       IN   PTR                ns2.intnet.

91      IN   PTR                mx1.intnet.
92      IN   PTR                mx2.intnet.
```

Nada muito diferente neste arquivo em relação ao anterior — note apenas que os registros inseridos aqui são do tipo PTR (*pointer*), fazendo o mapeamento reverso entre endereços IP e nomes de domínio. Note, ainda, que o *ORIGIN* do arquivo é definido como `42.0.10.in-addr.arpa.`, já que somos o servidor autoritativo para a faixa 10.0.42.0/24.

6. Vamos verificar se a sintaxe dos arquivos de configuração não possui erros:

```
# nsd-checkconf /etc/nsd/nsd.conf
```

Agora, basta reiniciar o serviço e verificar sua correta operação:

```
# nsd-control start
```

```
# tail /var/log/nsd.log
[2018-11-13 17:17:26.763] nsd[2012]: notice: nsd starting (NSD 4.1.14)
[2018-11-13 17:17:26.780] nsd[2014]: notice: nsd started (NSD 4.1.14), pid 2013
```

```
# ss -tunlp | grep 8053
udp    UNCONN     0      0      10.0.42.1:8053                  *:*
users:(("nsd",pid=648,fd=5),("nsd",pid=647,fd=5),("nsd",pid=646,fd=5))
udp    UNCONN     0      0      127.0.0.1:8053                  *:*
users:(("nsd",pid=648,fd=4),("nsd",pid=647,fd=4),("nsd",pid=646,fd=4))
tcp    LISTEN     0      128    10.0.42.1:8053                  *:*
users:(("nsd",pid=648,fd=7),("nsd",pid=647,fd=7),("nsd",pid=646,fd=7))
tcp    LISTEN     0      128    127.0.0.1:8053                  *:*
users:(("nsd",pid=648,fd=6),("nsd",pid=647,fd=6),("nsd",pid=646,fd=6))
```

7. Vamos colocar o servidor à prova: teste a resolução direta de nomes usando a ferramenta `dig`. Lembre-se que o NSD está operando na porta 8053/UDP, e não na porta padrão:

```
# dig @127.0.0.1 -p 8053 fw.intnet +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 fw.intnet +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64874
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
fw.intnet.              86400   IN      CNAME   ns1.intnet.
ns1.intnet.             86400   IN      A       10.0.42.1

;; AUTHORITY SECTION:
intnet.                 86400   IN      NS      ns1.intnet.
intnet.                 86400   IN      NS      ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 09:19:30 -02 2018
;; MSG SIZE  rcvd: 120
```

Excelente! Todas as seções de resposta parecem corretas, tanto ANSWER quando AUTHORITY. Vamos verificar a resolução reversa:

```
# dig @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4272
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
2.42.0.10.in-addr.arpa. 86400   IN      PTR     ns2.intnet.

;; AUTHORITY SECTION:
42.0.10.in-addr.arpa.   86400   IN      NS      ns1.intnet.
42.0.10.in-addr.arpa.   86400   IN      NS      ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 09:19:47 -02 2018
;; MSG SIZE  rcvd: 107
```

8. Agora que o NSD está configurado, vamos implementar o **Unbound**. Primeiro, pare o *daemon* se estiver rodando:

```
# systemctl stop unbound
```

Assim como no caso do NSD, o Unbound pode ser controlado via `unbound-control` — gere as chaves de controle:

```
# unbound-control-setup
setup in directory /etc/unbound
unbound_server.key exists
unbound_control.key exists
create unbound_server.pem (self signed certificate)
create unbound_control.pem (signed client certificate)
Signature ok
subject=CN = unbound-control
Getting CA Private Key
Setup success. Certificates created.
```

Faça um backup do arquivo de configuração original `/etc/unbound/unbound.conf`:

```
# mv /etc/unbound/unbound.conf /etc/unbound/unbound.conf.orig
```

Em seguida, crie o arquivo novo `/etc/unbound/unbound.conf` com o seguinte conteúdo:

```
server:
  interface: 127.0.0.1
  interface: 10.0.42.1
  interface: 192.168.42.1
  port: 53

  access-control: 127.0.0.0/8 allow
  access-control: 10.0.42.0/24 allow
  access-control: 192.168.42.0/24 allow

  cache-min-ttl: 300
  cache-max-ttl: 14400

  local-zone: "intnet" nodefault
  domain-insecure: "intnet"

  local-zone: "10.in-addr.arpa." nodefault
  domain-insecure: "10.in-addr.arpa."

  verbosity: 1
  prefetch: yes
  hide-version: yes
  hide-identity: yes
  use-caps-for-id: yes
  rrset-roundrobin: yes
  minimal-responses: yes
  do-not-query-localhost: no

stub-zone:
  name: "intnet"
  stub-addr: 127.0.0.1@8053

stub-zone:
  name: "42.0.10.in-addr.arpa."
  stub-addr: 127.0.0.1@8053

forward-zone:
  name: "."
  forward-addr: 8.8.8.8
  forward-addr: 8.8.4.4

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

Abaixo, destacamos e explicamos algumas das diretivas mais relevantes do arquivo acima:

- `server/interface`: define os endereços de rede nos quais o Unbound irá escutar — atenderemos requisições de *localhost* e das sub-redes DMZ e Intranet.
- `server/port`: o Unbound irá escutar na porta padrão, 53/UDP.
- `server/access-control`: define que as sub-redes declaradas (*localhost*, DMZ e Intranet, novamente) terão permissão para utilizar os serviços de resolução de nomes.
- `server/local-zone` e `server/domain-insecure`: define opções para as zonais locais `intnet.` e `42.0.10.in-addr.arpa.`, e que a cadeia de confiança DNSSEC não será verificada para esses domínios.
- `server/hide-version` e `server/hide-identity`: controles que objetivam mascarar o software rodando na máquina local, dificultando a tarefa de *banner grabbing* e identificação de *exploits* por parte de eventuais atacantes.
- `stub-zone` e `stub-zone/stub-addr`: declara zonas autoritativas locais que devem ser consultadas para os domínios especificados, para as quais o registro público DNS não será usado. Usaremos o servidor NSD rodando em *localhost*, consultando a porta 8053/UDP.
- `forward-zone` e `forward-zone/forward-addr`: define servidores nos quais o Unbound irá buscar recursão caso não consiga resolver um nome. Como está sendo declarado o nome de zona `"."`, todas as pesquisas serão redirecionadas para os servidores indicados em `forward-addr`.

9. Vamos verificar o funcionamento do *daemon*. Inicie o Unbound usando o `unbound-control`:

```
# unbound-control start
```

Verifique seu funcionamento usando o `ss`:

```
# ss -unlp | grep :53
UNCONN     0      0      192.168.42.1:53                  *:*
users:(("unbound",pid=2084,fd=7))
UNCONN     0      0      10.0.42.1:53                     *:*
users:(("unbound",pid=2084,fd=5))
UNCONN     0      0      127.0.0.1:53                     *:*
users:(("unbound",pid=2084,fd=3))
```

Reconfigure o DNS *system-wide*, no arquivo `/etc/resolv.conf`.

```
# nano /etc/resolv.conf
(...)
```

```
# cat /etc/resolv.conf
domain intnet.
search intnet.
nameserver 10.0.42.1
nameserver 10.0.42.2
```

Devido ao fato de estarmos usando DHCP na interface de saída (`enp0s3`), o arquivo `/etc/resolv.conf` poderá ser sobrescrito sempre que as interfaces de rede forem reiniciadas, ou após o *reboot* da máquina. Para evitar isso, marque o arquivo como imutável:

```
# chattr +i /etc/resolv.conf
```

10. Feito! Vamos testar a resolução de domínios internos `unbound` com a ferramenta `dig`.

```
# dig fw.intnet +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> fw.intnet +noquestion +noadditional
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63613
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
fw.intnet.              86400   IN      CNAME   ns1.intnet.
ns1.intnet.             86400   IN      A       10.0.42.1

;; Query time: 0 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 09:39:55 -02 2018
;; MSG SIZE  rcvd: 72
```

E quanto a nomes de domínio externos, usando recursão? Vejamos:

```
# dig openbsd.org +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> openbsd.org +noquestion +noadditional
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5694
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
openbsd.org.            21599   IN      A       129.128.5.194

;; Query time: 420 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 09:40:33 -02 2018
;; MSG SIZE  rcvd: 56
```

11. Ainda não acabou! Lembre-se que nossas regras de firewall configuradas no começo desta sessão estão extremamente restritivas, então temos que criar exceções para que a consulta de nomes funcione. Vamos ver os tipos de acesso que precisamos liberar:

    1. O NSD será consultado apenas pelo Unbound em *localhost* (não é necessário criar regras neste caso), e pelo servidor DNS secundário em 10.0.42.2. Como nesse caso será feita transferência de zonas entre os servidores, é necessário liberar tráfego nos protocolos TCP e UDP na porta 8053.
    2. O Unbound local será consultado por todas as máquinas da DMZ e Intranet, na porta 53/UDP.
    3. Os clientes da Intranet devem conseguir consultar o Unbound rodando no servidor DNS secundário, porta 53/UDP. Não é necessário criar uma regra para a DMZ, neste caso, pois as máquinas estão na mesma sub-rede e portanto seu tráfego não passa pelo firewall (na *chain* FORWARD).

    Vamos ao requisito (a):

    ```
    # iptables -A INPUT -s 10.0.42.2/32 -p tcp -m tcp --dport 8053 -j ACCEPT
    ```

    ```
    # iptables -A INPUT -s 10.0.42.2/32 -p udp -m udp --dport 8053 -j ACCEPT
    ```

    Agora, o requisito (b):

    ```
    # iptables -A INPUT -s 10.0.42.0/24,192.168.42.0/24 -p udp -m udp --dport 53 -j
    ACCEPT
    ```

    E, finalmente, o critério (c) estabelecido inicialmente:

    ```
    # iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.2/32 -p udp -m udp --dport 53
    -j ACCEPT
    ```

    Salve as regras na configuração do firewall local:

    ```
    # /etc/init.d/netfilter-persistent save
    [....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
    persistent/plugins.d/15-ip4tables save
    run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
    done.
    ```

5) Configuração do DNSSEC

Vamos agora configurar o DNSSEC em nosso servidor autoritativo, de forma a produzir consultas assinadas digitalmente. Evidentemente, como estamos configurando um domínio fictício não

iremos conseguir propagar a assinatura pela cadeia de confiança raiz, mas esta atividade servirá como um exemplo caso você queira fazê-lo em sua organização posteriormente.

1. Primeiro, vamos instalar as ferramentas de suporte para gestão/geração de chaves DNSSEC:

```
# apt-get install ldnsutils haveged
```

2. Vamos criar as chaves de assinatura de zona (ZSK, ou *Zone Signing Key*) e chave (KSK, ou *Key Signing Key*). O KSK é um par de chaves público/privada usado para gerar uma assinatura digital para o ZSK. O ZSK, por sua vez, é um par de chaves público/privada para gerar uma assinatura digital conhecida como RRSIG (*Resource Record Signature*) para cada um dos RRSETs (*Resource Record Sets*) da zona. O website da Cloudflare (https://www.cloudflare.com/dns/dnssec/how-dnssec-works/) possui uma excelente explicação sobre o sistema DNSSEC que pode ser usada para consulta.

   Primeiro, entre no diretório `/etc/nsd`:

```
# cd /etc/nsd/
```

   Agora, vamos gerar o ZSK/KSK e armazenar seus nomes em uma variável temporária. Os algoritmos escolhidos estão de acordo com as melhores práticas recomendadas na RFC 6944 (https://tools.ietf.org/html/rfc6944).

```
# export ZSK=$( ldns-keygen -a RSASHA512 -b 2048 intnet )
```

```
# export KSK=$( ldns-keygen -k -a RSASHA512 -b 2048 intnet )
```

   Podemos remover o registro DS (*Delegation of Signing*) auto-gerado, já que iremos regerá-lo futuramente sob demanda.

```
# rm Kintnet.*.ds
```

   Vejamos como ficaram as chaves ZSK/KSK para o domínio:

```
# ls -1 Kintnet.*
Kintnet.+010+14936.key
Kintnet.+010+14936.private
Kintnet.+010+42867.key
Kintnet.+010+42867.private
```

   Para identificar qual dessas chaves é a ZSK e qual é a KSK, consulte as variáveis do *shell*:

```
# echo $ZSK
Kintnet.+010+14936
```

```
# echo $KSK
Kintnet.+010+42867
```

3. Agora, vamos assinar a zona `intnet.zone` com o comando `ldns-signzone`:

```
# ldns-signzone -n -p -s $(head -n 1000 /dev/urandom | sha1sum | cut -b 1-16)
/etc/nsd/zones/intnet.zone $ZSK $KSK
```

O que ocorreu? Verifique o conteúdo do diretório `/etc/nsd/zones`:

```
# ls -1 /etc/nsd/zones
10.0.42.zone
intnet.zone
intnet.zone.signed
```

Vamos configurar o NSD para usar a zona assinada. Para isso, basta substituir o caminho de busca do arquivo de zona direta `/etc/nsd/zones/intnet.zone` por sua contraparte assinada:

```
# sed -i 's/intnet\.zone/intnet\.zone\.signed/' /etc/nsd/nsd.conf
```

```
# grep -B3 intnet.zone.signed /etc/nsd/nsd.conf
zone:
  name: "intnet"
  include-pattern: "inslave"
  zonefile: "intnet.zone.signed"
```

Para informar ao NSD que o arquivo de zonas mudou, basta usar o programa `nsd-control` como mostrado a seguir:

```
# nsd-control reconfig
reconfig start, read /etc/nsd/nsd.conf
ok
```

```
# nsd-control reload intnet
ok
```

4. Vamos testar se nossa configuração surtiu efeito. Primeiro, pesquise pelos registros DNSKEY do domínio no NSD em *localhost*, verificando as chaves ZSK e KSK:

```
# dig -p 8053 @localhost DNSKEY intnet. +multiline +norec +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> -p 8053 @localhost DNSKEY intnet. +multiline
+norec +noquestion
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18768
;; flags: qr aa; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
intnet.                 86400 IN DNSKEY 256 3 10 (
                                AwEAAerseiw1VUd7L90iBA4IgztUXpkw/k/6kKIHx5Qm
                                Cm1HvtLTT/dSFk9JOHVBoQ2Rpk0LQBc+WZsnqz6dQZJn
                                b9FkGzsAh39BnP8gOA1d+SfaGGkceEteQfgi3rKz5dnk
                                EWgOonWk2vGOJ5oPDvcy4aUQqfTACY9Tk0qeWZT7enfy
                                gdrg4AjaTKBdJ4kruBj2z3FKXBolDYO3HTPmosKe94Xr
                                muiLQa3uHFvFvu6k/X0mH1kCS+lb91R/OF60mG40iW4/
                                z8aLNSKPUuwJNddvZ0I14F7SwN7YGaVpoY11DAsxmbS5
                                1lGxp6Fc0fWQpAfT4WG9/mbyHj29kZnP/Oxktsk=
                                ) ; ZSK; alg = RSASHA512; key id = 14936
intnet.                 86400 IN DNSKEY 257 3 10 (
                                AwEAAdIo8BoUEY6ab3YBbs24FEj8Vt9aRByAFyKJk0sH
                                BiNIy9I1vXZIGB+5wlIKPzT4ZVd3oYO3MDExxDYZ1HVS
                                D1magCRBzLF2oeTCWY3kLz/9ls23RV9r5IMY68aPHMg0
                                tDBJ8IZTC9SbO+os3L9jazwpoL22Akl34YN6iIC6kXpQ
                                gN3TtNorzD8DIHkBvJ9NrpKYJjzt0g8oTyg0La3xTNcO
                                6q51Q4eVBnlcXbbJ6gquSFzAnoJ9qzq5JUnbvAB9I4yv
                                Spo3RGCX3lZFmsDvsDfnwkXShyeEPdGTr6m/mbqPrceW
                                R53QLOWnZq1KrczPwTlKjgtzwg+F7ol8YDdQOwU=
                                ) ; KSK; alg = RSASHA512; key id = 42867

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 11:13:56 -02 2018
;; MSG SIZE  rcvd: 587
```

Perfeito! Aparentemente, nossas chaves foram registradas e estão sendo utilizadas com sucesso. O Unbound, no entanto, ainda possui algumas entradas na *cache* desatualizadas, prévias à configuração do DNSSEC, que foram mantidas pelas consultas anteriores. Para limpá-las, basta usar o comando `unbound-control flush_zone`:

```
# unbound-control flush_zone intnet.
ok removed 4 rrsets, 2 messages and 0 key entries
```

Agora sim, faça uma consulta via Unbound usando DNSSEC:

```
# dig fw.intnet +dnssec +multiline +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> fw.intnet +dnssec +multiline +noquestion
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47390
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
fw.intnet.              86355 IN CNAME ns1.intnet.
fw.intnet.              86355 IN RRSIG CNAME 10 2 86400 (
                                20181210130559 20181112130559 14936 intnet.
                                u2DBcjAp91qYSYvspczvWiW8QuIX24aAoFTzfM8hJdUQ
                                3VcjekyJeQQfNPIM4+rE9ZdTywADS4nKCFl/XqkHwFST
                                kitVm9So17I4GBJ+sgpKkpoKEwh6IfbUB563Mmfj9jIQ
                                SvuR0dP0ot9QV+DXWCFLD02tvXTOBSltBT9Nogx1h926
                                5lkSZ0F+5x0Ss1Law5Rpn8ZKyAAwCYS0UNJIqB8XzIqb
                                eSblG5Q4Si7Qwrccv7Ll6DscBbh1gjmnCAYCE63XF3SN
                                s0N9SgTgfzibgVaVGw+S1SoVYjvxDVd5ZW/tpQxhx4kY
                                Wt/c+541nj7xmFT58/dnoo8t6Meo/JCKgw== )
ns1.intnet.             86297 IN A 10.0.42.1
ns1.intnet.             86297 IN RRSIG A 10 2 86400 (
                                20181210130559 20181112130559 14936 intnet.
                                bmwQPD9b/GwUX2ZvpP6baOzyEXNW/ghcc9rqzZSWwUYn
                                VfX+UqRwzon/2lxJfYO1uJh0v8noO2S5+zb7saroDsCi
                                9/Oaukw+iAvwrZh6VO1dHDr6WjAlS4hi9MxmfJ94NwD7
                                txB5WLG39KM0V/EpH+v4L6ser4v+oYFSer7a6EsZxCcD
                                nfciKtN5L/QUWVMacduxpRM16+MI83bEf/ulC82xIThf
                                2cVVdsDA4xfZvAOcYS/IXUauYXaj9fkE2TlVoyIC1UIM
                                XE6CSbulo2iWc+V+lj5tkP9kctmqit3rfegWXJlQ4es3
                                4fsxR64+Weqre/iTMpVxVss/TLus7g7OiA== )

;; Query time: 0 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 11:17:12 -02 2018
;; MSG SIZE  rcvd: 660
```

Caso fosse desejável exportar a configuração DNS para um *registrar* hierarquicamente superior (fechando a cadeia de verificação DNS), pode-se gerar os registros DS das chaves com o comando abaixo.

```
# ldns-key2ds -n -1 /etc/nsd/zones/intnet.zone.signed && ldns-key2ds -n -2
/etc/nsd/zones/intnet.zone.signed
intnet. 86400   IN      DS      42867 10 1 5b8c2c011bd011618f7e339667c075587e65ab05
intnet. 86400   IN      DS      42867 10 2
424bcf7d7c09df8be1520c4230680b5c33a63598b4c6185c7884592ad3bce63b
```

6) Automatizando assinatura DNSSEC após alterações

1. É claro, seria muito inconveniente ter que realizar todos os passos que fizemos na atividade (5) a cada alteração no servidor DNS. Vamos automatizar a assinatura de zonas e reinício dos *daemons* relevantes com um *script shell*. Crie o arquivo novo `/root/scripts/signzone-intnet.sh` com o seguinte conteúdo:

```
#!/bin/bash

ZSK="ZSKSTUB"
KSK="KSKSTUB"
ZONE_FILE="/etc/nsd/zones/intnet.zone"

cd /etc/nsd

ldns-signzone -n \
  -p \
  -s $(head -n 1000 /dev/random | sha1sum | cut -b 1-16) \
  $ZONE_FILE \
  $ZSK \
  $KSK

nsd-control reconfig
nsd-control reload intnet
nsd-control reload 42.0.10.in-addr.arpa
unbound-control flush_zone intnet.
```

   Note que é necessário substituir as variáveis `ZSKSTUB` e `KSKSTUB` com os valores da variáveis `$ZSK` e `$KSK` no seu *shell* corrente, como se segue:

```
# sed -i "s/ZSKSTUB/${ZSK}/" /root/scripts/signzone-intnet.sh
```

```
# sed -i "s/KSKSTUB/${KSK}/" /root/scripts/signzone-intnet.sh
```

2. Teste a assinatura de zonas usando o *script*:

```
# bash scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 4 rrsets, 2 messages and 0 key entries
```

Ao atualizar arquivos de zona, não se esqueça que além de adicionar registros A, CNAME ou PTR conforme necessário, é também **imprescindível** incrementar o valor do *serial* do arquivo, próximo do topo no registro SOA. Se isso não for feito, o NSD não irá detectar que o arquivo de zona foi atualizado e suas modificações não serão propagadas.

7) Reconfiguração da VM *debian-template*

Agora que temos um firewall e servidores DNS na rede é interessante que alteremos a configuração padrão da VM `debian-template` para que as novas máquinas, clonadas a partir dela, já estejam corretamente ajustadas para operar em nosso *datacenter* simulado.

1. Ligue a máquina `debian-template` e acesse o terminal como o usuário `root`.

```
# hostname ; whoami
debian-template
root
```

2. Com a máquina `debian-template` ligada, acesse na janela principal do Virtualbox o menu *Settings* > *Network* > *Adapter 1* > *Attached to* e altere a opção para *Host-only Adapter*. O nome da rede *host-only* escolhido deve ser o mesmo alocado para a interface de rede da máquina virtual `ns1` que está conectada à DMZ. Seguindo o exemplo mostrado no início desta sessão, portanto, a rede escolhida seria a `Virtualbox Host-Only Ethernet Adapter #2`.
3. De volta ao terminal da máquina `debian-template`, vamos configurar a rede: edite o arquivo `/etc/network/interfaces` como se segue:

```
# nano /etc/network/interfaces
(...)
```

```
# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo enp0s3

iface lo inet loopback

iface enp0s3 inet static
address 10.0.42.253/24
gateway 10.0.42.1
```

   Para garantir que nenhum endereço IP antigo, primário, se mantenha alocado às interfaces, execute o comando `flush` e em seguida reinicie a rede do sistema:

```
# ip addr flush label 'enp0s*' ; systemctl restart networking
```

Verifique que as interfaces estão com os endereços corretamente alocados:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}'
10.0.42.253/24 enp0s3
```

4. Agora, configure a resolução de nomes no arquivo `/etc/resolv.conf`:

```
# nano /etc/resolv.conf
(...)
```

```
# cat /etc/resolv.conf
domain intnet.
search intnet.
nameserver 10.0.42.1
nameserver 10.0.42.2
```

5. Vamos alterar o *script* `/root/scripts/changehost.sh` para que possamos configurar, de uma vez só, informações como endereço IP e *hostname* na máquina clonada. Apague todo o conteúdo do *script* original e substitua-o pelo seguinte:

```
#!/bin/bash


# exibir uso do script e sair
function usage() {
  echo "  Usage: $0 -h HOSTNAME -i IPADDR -g GATEWAY"
  echo "  Netmask is assumed as /24."
  exit 1
}


# testar sintaxe valida de HOSTNAME
function valid_host() {
  if [[ "$nhost" =~ [^a-z0-9-] ]]; then
    echo "  [*] HOSTNAME must be lowercase alphanumeric: [a-z0-9]*"
    usage
  elif [ ${#nhost} -gt 63 ]; then
    echo "  [*] HOSTNAME must have <63 chars"
    usage
  fi
}


```

```
# testar sintaxe valida de IPADDR/GATEWAY
function valid_ip() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    OIFS=$IFS
    IFS='.'
    ip=($ip)
    IFS=$OIFS
    [[ ${ip[0]} -le 255 && \
       ${ip[1]} -le 255 && \
       ${ip[2]} -le 255 && \
       ${ip[3]} -le 255 ]]
        stat=$?
  fi

  if [ $stat -ne 0 ] ; then
    echo "  [*] Invalid syntax for $2"
    usage
  fi
}


while getopts ":g:h:i:" opt; do
  case "$opt" in
    g)
      ngw=${OPTARG}
      ;;
    h)
      nhost=${OPTARG}
      ;;
    i)
      nip=${OPTARG}
      ;;
    *)
      usage
      ;;
  esac
done

# testar se parametros foram informados
[ -z $ngw ]   && { echo "  [*] No gateway?"; usage; }
[ -z $nhost ] && { echo "  [*] No hostname?"; usage; }
[ -z $nip ]   && { echo "  [*] No ipaddr?"; usage; }

# testar sintaxe de parametros
valid_ip $nip "IPADDR"
valid_ip $ngw "GATEWAY"
valid_host $nhost

```

```
# alterar endereco ip/gateway
iff="/etc/network/interfaces"
cip="$( egrep '^address ' $iff | awk -F'[ /]' '{print $2}' )"
cgw="$( egrep '^gateway ' $iff | awk '{print $NF}' )"
sed -i "s|${cip}|${nip}|g" $iff
sed -i "s|${cgw}|${ngw}|g" $iff
ip addr flush label 'enp0s*'

# alterar hostname local
chost="$( hostname -s )"
sed -i "s/${chost}/${nhost}/g" /etc/hosts
sed -i "s/${chost}/${nhost}/g" /etc/hostname

invoke-rc.d hostname.sh restart
invoke-rc.d networking restart
hostnamectl set-hostname $nhost

# re-gerar chaves SSH
rm -f /etc/ssh/ssh_host_* 2> /dev/null
dpkg-reconfigure openssh-server &> /dev/null
```

Em relação ao *script* original, algumas diferenças: adicionou-se um *loop* para leitura de opções a partir da linha de comando (*hostname*, endereço IP e *gateway* a serem utilizados pela nova máquina), verificação de sintaxe dessas opções e efetiva alteração do endereço IP e *gateway* no arquivo `/etc/network/interfaces`.

Feito isso, desligue a máquina `debian-template`.

8) Criação da VM de DNS secundário

Vamos agora criar a segunda máquina de nosso *datacenter* simulado, a máquina `ns2`. Ela atuará como um servidor DNS secundário sob o endereço IP 10.0.42.2/24, dentre outras funções que serão configuradas em sessões posteriores.

1. Clone a máquina `debian-template` seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha `ns2`.
2. Ligue a máquina `ns2` e, após o *boot*, faça login como o usuário `root`. Usando o script `/root/scripts/changehost.sh`, efetue a configuração automática da máquina:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
whoami
10.0.42.253/24 enp0s3
debian-template
root
```

```
# bash ~/scripts/changehost.sh -h ns2 -i 10.0.42.2 -g 10.0.42.1
```

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
whoami
10.0.42.2/24 enp0s3
ns2
root
```

9) Configuração do DNS secundário

A configuração do NSD como um servidor DNS secundário é bastante similar à do servidor primário, com algumas poucas diferenças na seção `pattern` do arquivo original. Já a configuração do Unbound, por outro lado, é absolutamente idêntica. Vamos ao trabalho:

1. Instale os pacotes relevantes:

   ```
   # apt-get install nsd unbound dnsutils
   ```

   Durante a pós-configuração do Unbound o sistema pode demorar um pouco a retornar para o *shell* — seja paciente, em alguns segundos a instalação será concluída.

2. Primeiro, o **NSD**. Queremos copiar o arquivo `/etc/nsd/nsd.conf` da máquina `ns1` e fazer as alterações necessárias — no entanto, o acesso SSH à essa máquina pela DMZ está proibido pelas configurações de firewall que fizemos no começo desta sessão. Assim sendo, vamos fazer a cópia no sentido oposto: da máquina `ns1` para a máquina `ns2`.

   Logue como `root` na máquina `ns1` e copie o arquivo usando o `scp`, como se segue:

   ```
   # hostname ; whoami
   ns1
   root
   ```

   ```
   # scp /etc/nsd/nsd.conf aluno@ns2:~
   The authenticity of host 'ns2 (10.0.42.2)' can't be established.
   ECDSA key fingerprint is SHA256:jxd7SPFgwNSsaMS7ApIEMpdAmxEnWeJ83s/K4h2XV5o.
   Are you sure you want to continue connecting (yes/no)? yes
   Warning: Permanently added 'ns2,10.0.42.2' (ECDSA) to the list of known hosts.
   aluno@ns2's password:
   nsd.conf
   100%  909     2.5MB/s   00:00
   ```

   Perfeito, o arquivo foi copiado para `/home/aluno/nsd.conf` na máquina `ns2` — note que não fizemos o login remoto com o usuário `root` pois esse acesso não é permitido pela política padrão do *daemon* `sshd` no Debian.

3. De volta à máquina `ns2` como o usuário `root`, pare o NSD e gere as chaves TLS de controle:

```
# hostname ; whoami
ns2
root
```

```
# systemctl stop nsd
```

```
# nsd-control-setup
setup in directory /etc/nsd
nsd_server.key exists
nsd_control.key exists
create nsd_server.pem (self signed certificate)
create nsd_control.pem (signed client certificate)
Signature ok
subject=CN = nsd-control
Getting CA Private Key
Setup success. Certificates created.
```

Vamos fazer o backup do arquivo de configuração original:

```
# mv /etc/nsd/nsd.conf /etc/nsd/nsd.conf.orig
```

Copie o arquivo `/home/aluno/nsd.conf` para a pasta `/etc/nsd` e ajuste suas permissões:

```
# cp /home/aluno/nsd.conf /etc/nsd
```

```
# chown root. /etc/nsd/nsd.conf
```

```
# ls -ld /etc/nsd/nsd.conf
-rw-r--r-- 1 root root 909 nov 12 12:14 /etc/nsd/nsd.conf
```

Excelente. Temos agora que editar o arquivo com as configurações do servidor secundário, o que faremos através do uso do `sed` nos comandos a seguir:

```
# sed -i 's/^\( *ip-address:\) 10\.0\.42\.1/\1 10\.0\.42\.2/' /etc/nsd/nsd.conf
```

```
# sed -i '/^ *zonesdir:/d' /etc/nsd/nsd.conf
```

```
# sed -i 's/\"inslave\"/\"inmaster\"/' /etc/nsd/nsd.conf
```

```
# sed -i 's/^\( *\)notify:[0-9@. ]*\(.*\)/\1allow-notify: 10\.0\.42\.1 \2/'
/etc/nsd/nsd.conf
```

```
# sed -i 's/^\( *\)provide-xfr:[0-9. ]*\(.*\)/\1request-xfr: AXFR 10\.0\.42\.1@8053
\2/' /etc/nsd/nsd.conf
```

O que esses comandos fizeram, afinal? O comando `diff` pode ser usado para comparar arquivos, evidenciando as diferenças entre eles:

```
# diff -u /home/aluno/nsd.conf /etc/nsd/nsd.conf
--- /home/aluno/nsd.conf        2018-11-12 20:51:45.040000000 -0200
+++ /etc/nsd/nsd.conf   2018-11-12 20:55:34.156000000 -0200
@@ -1,10 +1,9 @@
 server:
   ip-address: 127.0.0.1
-  ip-address: 10.0.42.1
+  ip-address: 10.0.42.2
   do-ip4: yes
   port: 8053
   username: nsd
-  zonesdir: "/etc/nsd/zones"

   logfile: "/var/log/nsd.log"
   pidfile: "/run/nsd/nsd.pid"
@@ -27,17 +26,17 @@
   secret: "i7IoB5VDHVOCW9wvuOGQuFLNu8hzfAblVAbCD1SbPL4="

 pattern:
-  name: "inslave"
-  notify: 10.0.42.2@8053 inkey
-  provide-xfr: 10.0.42.2 inkey
+  name: "inmaster"
+  allow-notify: 10.0.42.1 inkey
+  request-xfr: AXFR 10.0.42.1@8053 inkey

 zone:
   name: "intnet"
-  include-pattern: "inslave"
+  include-pattern: "inmaster"
   zonefile: "intnet.zone.signed"

 zone:
   name: "42.0.10.in-addr.arpa"
-  include-pattern: "inslave"
+  include-pattern: "inmaster"
   zonefile: "10.0.42.zone"
```

Não se esqueça de remover o arquivo `/home/aluno/nsd.conf`:

```
# rm /home/aluno/nsd.conf
```

4. Inicie o *daemon* do NSD usando o `nsd-control`:

```
# nsd-control start
```

Note que não foi necessário criar o diretório de zonas no servidor secundário, `/etc/nsd/zones`, já que as zonas transferidas ficam mantidas no diretório `/var/lib/nsd`:

```
# ls -1 /var/lib/nsd
nsd.db
xfrd.state
```

Vamos testar? Tente efetuar uma resolução direta de nomes no servidor local usando a ferramenta `dig`, também testando o funcionamento do DNSSEC:

```
# dig @127.0.0.1 -p 8053 ns1.intnet +dnssec +noadditional +noquestion +multiline
+noauthority

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 ns1.intnet +dnssec
+noadditional +noquestion +multiline +noauthority
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29660
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 3
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
ns1.intnet.             86400 IN A 10.0.42.1
ns1.intnet.             86400 IN RRSIG A 10 2 86400 (
                              20181210223558 20181112223558 14936 intnet.
                              TdK1Ivm1Ykdk1MBaMQSqNHANV14FQbfdBCgiOznMc7wT
                              vqZPa2TBMTe2GYvcjbWd2WzH4o88pOAzwmCxMnnAxGJ4
                              7IglUx4GlQKTG/hx/5vxpFYgczJLAKpt/HEMfrInYVyP
                              qvCFpCyUudgA7kV8w05+BvBiK4IQAWZCiH8GcC3ERFoC
                              B/ZNE/f8YnMyi2sNqdN8Mhtkgz2vMZnACiLrifnV7hlI
                              3uvqn0fQetnDWLF/xSeHoAE4WMZ6KwMgutUT1H9lQwOg
                              Ci5PemGKnOn2Va4ARzvQeTzBdLHZlVi04X25NHIeKZTj
                              8Sx1vRnBwPfT8qlyI1oaOozuiLSyrII3Mw== )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 20:58:05 -02 2018
;; MSG SIZE  rcvd: 985
```

Excelente. E quanto à resolução reversa?

```
# dig @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44932
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
2.42.0.10.in-addr.arpa. 86400   IN      PTR     ns2.intnet.

;; AUTHORITY SECTION:
42.0.10.in-addr.arpa.   86400   IN      NS      ns1.intnet.
42.0.10.in-addr.arpa.   86400   IN      NS      ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 20:58:17 -02 2018
;; MSG SIZE  rcvd: 107
```

5. Agora, vamos configurar o **Unbound**. Como feito anteriormente, logue como `root` na máquina `ns1` e copie desta vez o arquivo `/etc/unbound/unbound.conf` usando o `scp`, como se segue:

```
# hostname ; whoami
ns1
root
```

```
# scp /etc/unbound/unbound.conf aluno@ns2:~
aluno@ns2's password:
unbound.conf
100%  820     2.4MB/s   00:00
```

   O arquivo foi copiado para `/home/aluno/unbound.conf` na máquina `ns2`, como objetivado.

6. De volta a `ns2` como o usuário `root`, pare o *daemon* do Unbound e configure as chaves de controle:

```
# hostname ; whoami
ns2
root
```

```
# systemctl stop unbound
```

```
# unbound-control-setup
setup in directory /etc/unbound
unbound_server.key exists
unbound_control.key exists
create unbound_server.pem (self signed certificate)
create unbound_control.pem (signed client certificate)
Signature ok
subject=CN = unbound-control
Getting CA Private Key
Setup success. Certificates created.
```

Faça o backup do arquivo de configuração original:

```
# mv /etc/unbound/unbound.conf /etc/unbound/unbound.conf.orig
```

Copie o arquivo `/home/aluno/unbound.conf` para a pasta `/etc/unbound` e ajuste suas permissões:

```
# cp /home/aluno/unbound.conf /etc/unbound
```

```
# chown root. /etc/unbound/unbound.conf
```

```
# ls -ld /etc/unbound/unbound.conf
-rw-r--r-- 1 root root 820 nov 12 22:15 /etc/unbound/unbound.conf
```

Edite o arquivo com as configurações do servidor secundário via `sed`, como se segue:

```
# sed -i 's/^\( *interface: 10\.0\.42\.\)1/\12/' /etc/unbound/unbound.conf
```

```
# sed -i '/^ *interface: 192.*/d' /etc/unbound/unbound.conf
```

Use o comando `diff` para verificar as diferenças entre o arquivo original e as alterações feitas com o `sed`:

```
# diff -u /home/aluno/unbound.conf /etc/unbound/unbound.conf
--- /home/aluno/unbound.conf    2018-11-12 22:13:47.276000000 -0200
+++ /etc/unbound/unbound.conf   2018-11-12 22:24:48.564000000 -0200
@@ -1,7 +1,6 @@
 server:
   interface: 127.0.0.1
-  interface: 10.0.42.1
-  interface: 192.168.42.1
+  interface: 10.0.42.2
   port: 53

   access-control: 127.0.0.0/8 allow
```

Remova o arquivo de configuração do Unbound no diretório `/home/aluno`:

```
# rm /home/aluno/unbound.conf
```

7. Vamos proceder aos testes. Inicie o Unbound usando o comando `unbound-control`:

```
# unbound-control start
```

Teste uma resolução direta qualquer no servidor Unbound local, solicitando DNSSEC:

```
# dig @127.0.0.1 mx1.intnet +dnssec +multiline +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 mx1.intnet +dnssec +multiline
+noquestion +noadditional
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53007
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
mx1.intnet.             86388 IN A 10.0.42.91
mx1.intnet.             86388 IN RRSIG A 10 2 86400 (
                                20181210223558 20181112223558 14936 intnet.
                                YPluG8EXq3WwlKIlMpeXc+C1C3AZhC+P1ej+6I5ax9Aw
                                xQ+Zp6lwMXz64e7nPN2mjNa1PcAVuUNc8gE77U5dohhI
                                Al++b4+1mPHLzN6EY4UuSsWqkE5NacSaUngmhV52Mrwj
                                cwXtmJgJm9vhSTLKnUSYhSwFWZlC2FxI96kTQhtzJuDd
                                KI7SVpYsNZPGiKwj5r9F8Wr/sKpCc0PgVnseewyPmNGY
                                5pJEV3Ox6XgzQk0qjatoO4IxSj3CqX6ghU+MykcU2L+n
                                56LfqqrV+R6TzzVst9/bFC7UgSXjF4v9eDKvPc86mcGn
                                J9giG3w+07D2+jv7Ap6RfQWnx1zBy5CWFA== )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Nov 12 22:45:11 -02 2018
;; MSG SIZE  rcvd: 349
```