PSAwise2324Team10Aufgabe03

DNS Server

We decided to create a new VM that will function as a designated DNS Server, VM4. We chose BIND9 to manage our DNS Server because it is very commonly used.

To install BIND9, run `sudo apt update` and `sudo apt install bind9`

Setting a DNS Server

To change which DNS server your machine uses, you need to specify the correct DNS server. The DNS settings are read from /etc/resolv.conf. However, this resolv.conf is dynamically overwritten by default. To make it static, you have to remove the symlink. Delete the original resolv.conf file with `sudo rm /etc/resolv.conf`., then create a new file of the same name and give it the contents you want. Finally restart resolved using `sudo systemctl restart systemd-resolved`.

```
nameserver 192.168.10.4
domain psa-team10.cit.tum.de
search psa-team10.cit.tum.de
```

DNS Firewall

Make sure that the firewall accepts udp sport and dport 53 in both input and output chain. This is necessary so that it can both accept queries/send responses to clients as well as send queries/accept responses from the DNS servers it forwards to. The client VMs only need input sport 53 and output dport 53 rules.

Configuration

To apply any of the following changes, you need to run `sudo systemctl restart bind9`.

Keep in mind that all configuration files should have permission 640.

1) Primary Server

Open /etc/bind/named.conf.local with a text editor of your choice. Here you can add new zones to your DNS server. Insert the following block in order to add a new primary zone for the subnet:

```
zone "psa-team10.cit.tum.de" {
        type primary;
        file "/etc/bind/db.psa-team10.cit.tum.de";
};
```

The name of our primary zone is psa-team10.cit.tum.de, since that is our domain name.

Create the corresponding file /etc/bind/db.psa-team10.cit.tum.de from an existing zone file with `sudo cp /etc/bind/db.local /etc/bind/db.psa-team10.cit.tum.de`.
You have to replace the original domain name "localhost" with your domain name. Make sure not to forget the "." at the end of the name. At the bottom, you can add new records for this zone. The name of the machine that the IP address belongs to belongs in the first column, the IP address belongs in the last. Since we are only using IPv4 addresses, we only have type A records (= IPv4 records). For the sake of convenience, we also created a few shorter aliases (CNAMEs) for our VMs.

```
;
; BIND data file for psa-team10.cit.tum.de
;
$ORIGIN psa-team10.cit.tum.de
$TTL    86400
@               IN      SOA     dns1.psa-team10.cit.tum.de. psa.in.tum.de. (
                                      5         ; Serial
                                  21600         ; Refresh
                                   3600         ; Retry
                                 604800         ; Expire
                                  86400 )       ; Negative Cache TTL

                IN      NS      dns1.psa-team10.cit.tum.de.
dns1            IN      A       192.168.10.4

vmpsateam10-01  IN      A       192.168.10.1
vmpsateam10-02  IN      A       192.168.10.2
vmpsateam10-03  IN      A       192.168.10.3
vmpsateam10-04  IN      A       192.168.10.4

vm10-1          IN      CNAME   vmpsateam10-01
vm10-2          IN      CNAME   vmpsateam10-02
vm10-3          IN      CNAME   vmpsateam10-03
vm10-4          IN      CNAME   vmpsateam10-04
```

2) Reverse Zone

Open /etc/bind/named.conf.local again and add this new zone:

```
zone "10.168.192.in-addr.arpa" {
        type primary;
        file "/etc/bind/db.192.168.10";
};
```

Note that the first three octets in the zone name are in reverse order.

Same as with the primary server, you now have to create the database file. In this case, it is named db.192.168.10 to specify that this is the zone file for all addresses with this subnet mask. The file should have these contents:

```
  GNU nano
7.2                        db.192.168.10                                  ;
; BIND reverse data file for 192.168.10.xxx
;
$ORIGIN 10.168.192.in-addr.arpa.
$TTL    604800
```

```
@       IN      SOA     dns1.psa-team10.cit.tum.de. root.psa-
team10.cit.tum.de. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

        IN      NS      dns1.psa-team10.cit.tum.de.

4       IN      PTR     vmpsateam10-04.
1       IN      PTR     vmpsateam10-01.
2       IN      PTR     vmpsateam10-02.
3       IN      PTR     vmpsateam10-03.
```

This is similar to the database file of the ordinary zone, only that the machine names and IP addresses are reversed. The numbers in the first column of each record represent the last octet of the matching IP address, which is assumed to begin with 192.168.10.

4) Redirect to other teams

For each team whose DNS server a DNS query of the appropriate zone should be directed to, you add an entry to /etc/bind/named.conf.local that looks something like this:

```
zone "psa-team1.cit.tum.de" {
        type forward;
        forwarders { 192.168.1.1; };
        forward only;
};
zone "1.168.192.in-addr.arpa" {
        type forward;
        forwarders { 192.168.1.1; };
        forward only;
};
```

Here, the IP address specified under "forwarders" has to be the IP address of the team's DNS server. If you want to keep your named.conf.local file clean, it is recommended that you put all of these zone definitions in a separate file, e.g. "zones.psa_teams", and simply include it in the named.conf.local file using `include "etc/bind/zones.psa_teams"` at the top of the file.

5) Redirect all others

In order to redirect all DNS queries that don't fall into the zones you defined manually, add the following option in /etc/bind/named.conf.options:

```
acl "trusted" {
    127.0.0.1;
    192.168.0.0/16;
};

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

        dnssec-validation no;
```

```
        allow-query { trusted; };
        allow-transfer { trusted; };

        forwarders { 131.159.254.1; 131.159.254.2; };
        forward only;

        recursion yes;
        empty-zones-enable yes;
        allow-recursion { trusted; };

        listen-on { any; };
        listen-on-v6 {};
};
```

With 131.159.254.1 and 131.159.254.2 being the IP addresses of the internal DNS servers. `forward only` here simply means that the DNS server will not attempt to contact any DNS servers other than the forwarders in case they happen to be down.

6) Secondary Server

To set up a secondary server for zone transfers, first allow transfers in your primary and primary-reverse zones. To do this, extend /etc/bind/named.conf.local with `allow-transfer` and `allow-notify` as follows. (We will create secondary zones for Team 1 and Team 9)

```
zone "psa-team10.cit.tum.de" {
        type primary;
        file "/etc/bind/db.psa-team10.cit.tum.de";
        allow-transfer { 192.168.1.1; 192.168.9.2; };
        also-notify { 192.168.1.1; 192.168.9.2; };
};

zone "10.168.192.in-addr.arpa" {
        type primary;
        file "/etc/bind/db.192.168.10";
        allow-transfer { 192.168.1.1; 192.168.9.2; };
        also-notify { 192.168.1.1; 192.168.9.2; };
};


include "/etc/bind/zones.secondary";
```

The IP addresses are those of the targeted DNS servers. Next, create the file called zones.secondary in the /etc/bind folder and include it in named.conf.local. Add entries like this for every zone you want to do a zone transfer within the file:

```
zone "psa-team1.cit.tum.de" {
        type secondary;
        file "/var/lib/bind/db.team1";
        primaries { 192.168.1.1; };
};

zone "1.168.192.in-addr.arpa" {
        type secondary;
        file "/var/lib/bind/db.team1_rev";
```

```
        primaries { 192.168.1.1; };
};
```

With the attribute file, you specify where the zone files of those DNS servers will be stored. We suggest putting them in /var/lib/bind. They should be named as seen in the configuration above.

Once you've done all this, apply changes by restarting the bind server `sudo systemctl restart bind9`. The zone transfer should happen automatically after the restart and, in the future, whenever there's a change in the zone files.

DHCP Server

To configure a DHCP server, we decided to use Kea. Kea is a DHCP server developed by ISC and replaced the deprecated DHCP server isc-dhcp developed by the same organization.

To install isc-kea, run the usual install command: `sudo apt install kea`.
Afterwards, we are able to configure the DHCP4 server by editing the config file {{code language="bash"}}/etc/kea/kea-dhcp4.conf{{/code}} .

The following box contains our config file:

```
{
   "Dhcp4": {
      "interfaces-config": {
         "interfaces": [
            "enp0s8/192.168.10.1"
         ]
      },
 "option-def":[
            {
                "name":"web-proxy",
                "code": 252,
                "type":"string"
            },
    {
  "name": "rfc3442-classless-static-routes",
                "code": 121,
                "type": "record",
                "array": true,
                "record-types": "uint8,ipv4-address,uint8,ipv4-address"
            }
        ],
      "control-socket": {
         "socket-type": "unix",
         "socket-name": "/tmp/kea4-ctrl-socket"
      },
      "lease-database": {
         "type": "memfile",
         "lfc-interval": 3600
      },
      "expired-leases-processing": {
         "reclaim-timer-wait-time": 10,
         "flush-reclaimed-timer-wait-time": 25,
```

```
        "hold-reclaimed-time": 3600,
        "max-reclaim-leases": 100,
        "max-reclaim-time": 250,
        "unwarned-reclaim-cycles": 5
    },
    "renew-timer": 900,
    "rebind-timer": 1800,
    "valid-lifetime": 3600,
    "option-data": [],
    "subnet4": [
       {
          "id": 1,
          "subnet": "192.168.10.0/24",
          "pools": [
             {
                "pool": "192.168.10.100 - 192.168.10.200"
             }
          ],
          "reservations": [
             {
                "hw-address": "08:00:27:55:93:de",
                "ip-address": "192.168.10.1"
             },
             {
                "hw-address": "08:00:27:44:4c:d9",
                "ip-address": "192.168.10.4"
             },
      {
 "hw-address": "08:00:27:d9:f7:54",
               "ip-address": "192.168.10.3"
             }
          ],
          "option-data": [
             {
                "name": "routers",
                "data": "192.168.10.1"
             },
             {
                "name": "domain-name-servers",
                "data": "192.168.10.4"
             },
             {
                "name": "domain-name",
                "data": "psa-team10.cit.tum.de"
             },
      {
         "name": "web-proxy",
  "code": 252,
         "data": "http://proxy.in.tum.de/wpad.dat",
         "always-send": true
           },
      {
"name": "rfc3442-classless-static-routes",
              "code": 121,
              "data": "16,192.168.0.0,24,192.168.10.1"
             }
```

```
            ]
         }
      ],
      "loggers": [
         {
            "name": "kea-dhcp4",
            "output_options": [
               {
                  "output": "/var/log/kea-dhcp4.log"
               }
            ],
            "severity": "INFO",
            "debuglevel": 0
         }
      ]
   }
}
```

We specified the interface to be `enp0s8` and left the control socket, lease-related and loggers setting as given by default.
The settings specific to our installation are in "subnet". There, we specified our team 10 subnet and set the range for the dynamically offered IP addresses to 192.168.10.100-192.168.10.200.

Our DNS server is located at 192.168.10.4

To make sure the DHCP server is running as intended, we can use a shell script that came with the installation of Kea by running the script as follows: `keactrl <command> [-c keactrl-config-file] [-s server[,server,...]]`

- `keactrl start` starts all servers
- `keactrl stop` stops the servers
- `keactrl reload` reloads the servers
- `keactrl status` prints the status of all Kea DHCP servers

We have also decided to create a systemd service, which starts the Kea servers located at vm1 (192.168.10.1) automatically on boot.

The service file is located at `/etc/systemd/system/kea.service` and contains the following:

```
[Unit]
Description=Start keactrl
StartLimitIntervalSec=0

[Service]
Type=simple
ExecStart=sudo /usr/sbin/keactrl start
Restart=always
RestartSec=1
```

`[Install]`
`WantedBy=multi-user.target`

After saving the new file, make sure to execute the following commands so that our new service is enabled and started:

```
sudo systemctl daemon-reload
sudo systemctl enable kea.service
sudo systemctl start kea.service
```

To test the functionality of our new server, start a new VM for a new client. In our case it is vm03. Make sure the netplan config is only set to allow DHCP4.

Install the `dhclient` package, which gives us the ability to manually send DHCP discovery messages. First, run `dhclient -r` to drop already existing leases and then run `dhclient -v -d -1 enp0s8` to generate DHCP requests from the interface `enp0s8`.
Reject all DHCP offers on our testing vm that are not from our network, by appending the line `reject 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24, 192.168.4.0/24, 192.168.5.0/24, 192.168.6.0/24, 192.168.7.0/24, 192.168.8.0/24, 192.168.9.0/24;` to `/etc/dhcp/dhclient.conf`.
To enable the use of a web proxy on our dhcp client, make sure to also append the following lines to the same config file as above:

```
option web-proxy code 252 = text;
request web-proxy;
```

Finally, we need to adjust the firewalls employed on the vm hosting the DHCP server such that offers only get send to hosts in our designated subnet.

We do this by adding the rule `udp dport {67,68} ip saddr != 192.168.10.0/24 drop` to the input chain and the following rules to the output chain:
`udp dport {67,68} drop`
`udp sport {67, 68} drop`

Testing

Testing of DNS Server

The test file can be found on the DNS server (VM4).

```
#!/bin/bash

failed_tests=0

fail() {
    ((failed_tests++))
    echo "FAIL $@"
```

```
}

ok() {
    echo "OK $@"
}


try_nslookup() {
    timeout 0.5 nslookup $1 &> /dev/null 2>&1
    if [ $? -eq 0 ]; then
        ok "$1"
    else
        fail "$1"
    fi
}

echo "---FORWARD---"

echo "team 1"
try_nslookup "vm1.psa-team1.cit.tum.de"
echo "team 6"
try_nslookup "vmpsateam06-02.psa-team06.cit.tum.de."
echo "team 10 (US)"
d=".psa-team10.cit.tum.de"
try_nslookup "vmpsateam10-01$d"
try_nslookup "vmpsateam10-02$d"
try_nslookup "vm10-1"


echo ""
echo "---REVERSE---"
for team in {1..10}; do
    echo "Team $team"
    try_nslookup "192.168.$team.1"
    try_nslookup "192.168.$team.2"
done


echo "Tests failed: $failed_tests"
```

Testing of DHCP Server

Test on the DHCP server, which is located at vm1.

```
#!/bin/bash

echo "Test DHCP server"

# Test if DHCP4 server is active
res=$(keactrl status | head -n 1)
if [ -z "${res##*active*}" ]; then
    echo "DHCP4 server is running."
else
    echo "DHCP4 server is inactive."
fi
```

Test on the DHCP client, which is located at vm3.

Install package `dhcpcd` and execute the following script located in the root folder in the file `test_PSA_03.sh`:

```
#!/bin/bash

echo "Check DHCP server functionality."

echo "Send DHCP request to our server at 192.168.10.1."

DHCP_SERVER_IP="192.168.10.1"

TARGET_SUBNET="10"

# Maximum number of attempts
MAX_ATTEMPTS=10

# Sleep duration between attempts (in seconds)
SLEEP_DURATION=2

dhclient -r enp0s8

OBTAINED_IP=""

while [ -z "$OBTAINED_IP" ]
do
 echo "Attempt DHCP request"
 dhclient -1 -4 -s $DHCP_SERVER_IP enp0s8
 OBTAINED_IP=$(ip -4 addr show dev enp0s8 | awk '/inet / {print $2}' | cut -
d'/' -f1 | head -1)
 THIRD_OCTET=$(echo "$OBTAINED_IP" | awk -F'.' '{print $3}')

 if [ "$THIRD_OCTET" = "$TARGET_SUBNET" ]; then
  echo "Success! Obtained IP address is $OBTAINED_IP and is in our subnet."
  break
 else
  echo "Obtained IP address $OBTAINED_IP is not in our subnet. Wait for
another offer."
  dhclient -r enp0s8
  OBTAINED_IP=""
 fi
done
```