Lokální DNS a DNSSEC

Datum zpracování: 23. 11. 2023 - 03.12.2023

Zpracovali: Kevin Daněk, Big Shock! Apple 0.7L

Zadání

1. Zapojte lokální síť s DHCP.
2. Nakonfigurujte router a připojené počítače tak, aby DNS počítačům v síti přiděloval jména s koncovkou **.lan**.
3. Příkazem **dig +dnssec** zkontrolujte záznamy lokálních počítačů a serveru **www.tul.cz**
4. Porovnejte s výsledkem při zapojení přímo do sítě (s veřejnou adresou 147.230.x.x)
5. Do routeru v terminálovém připojení doinstalujte DNS **unbound** a nakonfigurujte předávání DNSSEC záznamů do sítě.
6. Doplňte **ubound** o IANA klíč umožňující ověřování lokálních DNSSEC záznamů.
7. V průběhu práce pořizujte screenshoty a záznamy použitých příkazů; použijte je v elaborátu a okomentujte postup.

Úlohu zpracovávejte ve 2 - 5 členných týmech.

Elaborát zpracujte do šablony v záhlaví kurzu, odevzdávejte ve formátu PDF.

Do abecedně seřazeného seznamu řešitelů na úvodním listu uveďte reálné složení týmu!

Simulace lokální sítě

Stejně jako minulé cvičení, i toto zpracovávám s kontejnery kvůli chabému technickému stavu učeby A305, takže je opět potřeba si trochu předpřipravit prostředí. Abych se ze všech těch příkazů nezbláznil, připravil jsem si `docker-compose.yaml` soubor, který slouží k vytvoření několika kontejnerů naráz a jejich nastavení. Kolektivně se všemu, co compose vytvoří, říká **stack** (Jeden compose file vytváří jeden stack). Přes `docker compose up` jsem následně celý stack spustil a všechny tři kontejnery se mi vytvořili a spustili.

```
PS C:\Users\HP\Práce\Škola\PBE> docker compose up -d
[+] Building 0.0s (0/0)                                         docker:default
[+] Running 4/4
 ✓ Network pbe_7_dns_dnssec_pbe  Cre...                                  0.1s
 ✓ Container client_2            Started                                0.3s
 ✓ Container dhcp                Started                                0.4s
 ✓ Container client_1            Started                                0.4s
```

Nastavení sítě mezi kontejnery

Kromě kontejnerů jsem také nechal v rámci stacku vytvořit vlastní síť, která má interní název pbe (reálně je její název ještě prefixován názvem stacku, tudíž reálné jméno je `pbe_7_dns_dnssec_pbe`). Pro jednodušší správu jsem síti nastavil subnet 172.20.0.0/16. V tomto případě by byl v pohodě i prefix /24, ale už se mi to nechtělo měnit.

```
PS C:\Users\HP> docker network list
  NETWORK ID     NAME                   DRIVER    SCOPE
  1c17db14cef8   bridge                 bridge    local
  fc808389db8e   host                   host      local
  53ad48d9a028   none                   null      local
+ 84e4f330b071   pbe_7_dns_dnssec_pbe   bridge    local
```

Nastavení dnsmasq

Protože nepotřebuju v rámci cvičení pracovat s celým routerem, ale pouze s DHCP a DNS, rozhodl jsem se sáhnout po dnsmasq, který mi v rámci docker sítě bude tuhle roli zastávat. Obrazů s dnsmasq existuje spousta, ovšem já jsem si zvolil obraz od uživatele `drpsychick`, který je plně konfigurovatelný skrz proměnné prostředí (`environment variables`), takže mi odpadá nutnost ke kontejneru dělat `bind mount` nebo `volume` jenom kvůli jednomu souboru.

```yaml
dhcp:
  container_name: "dhcp"
  image: "drpsychick/dnsmasq:latest"
  privileged: true
  environment:
    - DMQ_DHCP_DNS=dhcp-option=6,8.8.8.8,8.8.4.4
    - DMQ_DHCP_GATEWAY=dhcp-option=3,172.20.0.1
    - DMQ_DHCP_RANGES=dhcp-range=172.20.0.10,172.20.0.100,24h
    - DMQ_DNS_SERVER=server=8.8.8.8\nserver=8.8.4.4
```

Nastavení klientů

Pro účely testování jsem si vytvořil dva klienty, kteří, stejně jako v minulém cvičení, vycházejí z obrazu distribuce ubuntu. Abych nemusel pokaždé instalovat všechno, co potřebuju, vytvořil jsem pro klienty vlastní dockerfile, který obsahuje instrukce, co se má před jeho spuštěním nainstalovat.

```
FROM ubuntu
RUN apt update
RUN apt install -y net-tools dnsutils nano iputils-ping iproute2 dhcpcd5
CMD /bin/bash
```

Vyžádání DHCP u klientů

Všechny kontejnery podléhají adresování uvnitř docker sítě, a bohužel, automatické přidělování IP adres nejde v rámci sítě vypnout. Přemýšlel jsem, jak tohle nejlépe vyřešit a došel jsem k řešení, které mi dává největší smysl, a taky je asi nejjednodušší. DNSMASQ má nastavený rozsah pro přidělování 172.20.0.10 až 172.20.0.100, a automatické přidělování adres přiděluje od té nejnižší možné (172.20.0.2 je první volná adresa). Proto jsem DNSMASQ kontejneru nastavil statickou IP adresu (172.20.0.5) a u spuštěných klientů stačí spustit následující sekvenci příkazů:

```
root@7a78b8d0e1b1:/# ifconfig eth0 0.0.0.0
root@7a78b8d0e1b1:/# dhcpcd
root@7a78b8d0e1b1:/# ifconfig eth0 | grep inet
        inet 172.20.0.13  netmask 255.255.0.0  broadcast 172.20.255.255
```

Ta na rozhraní `eth0` (které je tam společně se smyčkou jediné) smaže přidělené IP adresy, vyžádá si novou IP adresu od DHCP serveru a vypíše novou adresu do konzole. To, že DHCP bylo vyžádáno, lze ověřit i výpisem v DNSMASQ kontejneru, kde jsou požadavky na DHCP logovány:

```
dnsmasq-dhcp[43]: DHCPDISCOVER(eth0) 02:42:ac:14:00:03
dnsmasq-dhcp[43]: DHCPOFFER(eth0) 172.20.0.14 02:42:ac:14:00:03
dnsmasq-dhcp[43]: DHCPREQUEST(eth0) 172.20.0.14 02:42:ac:14:00:03
dnsmasq-dhcp[43]: DHCPACK(eth0) 172.20.0.14 02:42:ac:14:00:03 6273cd83df93
```

Samozřejmě tohle lze automatizovat. U klientských kontejnerů jsem přidal direktivu depends_on, která definuje závislosti mezi kontejnery. V tomto případě závisí klienti na DHCP službě, kterou poskytuje DNSMASQ. V této direktivě lze nastavit, v jakém stavu závisejícího kontejneru se mají ty závislé spustit (např. při jeho startu, či pokud vyžaduje “zdravý” stav). Zde jsem zjistil, že stačí klienty spustit hned po startu DNSMASQ a funguje vše jak má. Následně jsem už jenom vytvořil jednoduchý skript z příkazů, které jsou výše:

```
RUN touch /entrypoint.sh
RUN echo "#!/bin/bash" >> /entrypoint.sh
RUN echo "ifconfig eth0 0.0.0.0" >> /entrypoint.sh
RUN echo "dhcpcd" >> /entrypoint.sh

CMD /bin/bash /entrypoint.sh; /bin/bash
```

Nastavení dnsmasq jako hlavního DNS pro klienty

Protože budeme chtít ve cvičení použít i funkcionality DNS serveru, které v sobě dnsmasq má, bylo potřeba klienty nasměrovat k používání právě DNSMASQ a né toho DNS, které implicitně používá docker síť. Záznamy o tom, jaké jmenné servery má klient použít, jsou uloženy v souboru `/etc/resolv.conf`, ovšem připadalo mi stupidní pokaždé tento soubor manuálně přepisovat, když routery standardně umí "naordinovat", které DNS použít.

V nastavení DNSMASQ existuje spousta možností pro DHCP server, a jedna z nich (s číslem 6) se jmenuje dns-server, která všem klientům, kteří mají DHCP lease předá informace o DNS serverech. Stačí tedy nastavit IP adresu DNSMASQ kontejneru jako primární DNS server pro klienty a je hotovo.

```
 container_name: "dhcp"
 image: "drpsychick/dnsmasq:latest"
 privileged: true
 environment:
+      - DMQ_DHCP_DNS=dhcp-option=option:dns-server,172.20.0.5,8.8.8.8,8.8.4.4
```

To, že se nové DNS servery propsaly ke klientům, lze ověřit vypsáním obsahu souboru `/etc/resolv.conf` po vyžádání DHCP:

```
root@b1f98a2ebfbf:/# cat /etc/resolv.conf
# Generated by resolvconf
nameserver 172.20.0.5
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Koncovka .lan

Nyní je čas se pustit do samostatného vypracování úkolů. První je zprovoznit mechanismus, kdy jsou v DNS záznamech uloženy záznamy pro jednotlivé stanice, které mají na routeru DHCP lease. Kromě toho mají být v doméně `.lan`.

DNSMASQ takové nastavení podporuje, tudíž jsem při brouzdání dokumentace našel patřičné proměnné a ty nastavil. Pak jsem následně ještě našel jejich ekvivalent v proměnných prostředí obrazu DNSMASQ a nasázel to vše do compose filu:

```
dhcp:
  container_name: "dhcp"
  image: "drpsychick/dnsmasq:latest"
  privileged: true
  environment:
+   - DMQ_DNS_DOMAIN=domain=lan
+   - DMQ_DNS_FLAGS=expand-hosts\ndomain-needed\nselfmx\ndns-loop-detect
+   - DMQ_DNS_LOCAL=local=/lan/
+   - DMQ_DNS_RESOLV=no-resolv
+   - DMQ_DNS_SERVER=server=8.8.8.8\nserver=8.8.4.4
```

Co jednotlivé možnosti dělají?

- **domain** nastaví doménu, která se bude automaticky přidávat k “nekvalifikovaným názvům”, tj. názvům, které nemají doménu.
- **flags** jsou příznaky pro DNSMASQ, které nějakým způsobem modifikují jeho chování
  - **expand-hosts** společně s nastavením **domain** vytvoří kvalifikované názvy hostů z /etc/hosts, které jsou poté přidány do DNS záznamů
  - **domain-needed** zajistí, že požadavky na přeložení nekvalifikovaných názvů (bez domény) nejsou přeposílány dále
  - **selfmx** vytvoří MX záznamy pro všechny lokální stanice ukazující na sebe sama
  - **dns-loop-detect** zamezí zacyklení dotazování
- **local** nastaví lokální doménu
- **no-resolv** určuje, že pro získání vyšších DNS serverů nemá DNSMASQ používat /etc/resolv.conf
- **server** určuje, jaké DNS servery vyššího řádu má DNSMASQ použít

To, jestli nyní vše funguje jak má, ověříme jednoduchým pingem. V době zkoušení měly kontejnery jména b16f7664f6b3 (klient 1) a 7a78b8d0e1b1 (klient 2). Když zkusíme pingnout na klienta 2 pomocí jeho síťového jména a koncovky .lan:

```
root@b16f7664f6b3:/# ping -c 4 7a78b8d0e1b1.lan
PING 7a78b8d0e1b1.lan (172.20.0.13) 56(84) bytes of data.
64 bytes from 7a78b8d0e1b1.lan (172.20.0.13): icmp_seq=1 ttl=64 time=3.07 ms
64 bytes from 7a78b8d0e1b1.lan (172.20.0.13): icmp_seq=2 ttl=64 time=0.156 ms
64 bytes from 7a78b8d0e1b1.lan (172.20.0.13): icmp_seq=3 ttl=64 time=0.335 ms
64 bytes from 7a78b8d0e1b1.lan (172.20.0.13): icmp_seq=4 ttl=64 time=0.223 ms

--- 7a78b8d0e1b1.lan ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.156/0.945/3.067/1.226 ms
```

Tak vše funguje tak, jak má. Ve výpisu lze dokonce vidět, A protože jsme nastavovali výchozí doménu pro nekvalifikovaná jména hostů, mělo by také fungovat, pokud na klienta 1 (b16f7664f6b3) pingneme bez koncovky .lan, resp. měla by se automaticky přidat za jeho jméno:

```
root@7a78b8d0e1b1:/# ping -c 4 b16f7664f6b3
PING b16f7664f6b3.lan (172.20.0.14) 56(84) bytes of data.
64 bytes from b16f7664f6b3.lan (172.20.0.14): icmp_seq=1 ttl=64 time=1.24 ms
64 bytes from b16f7664f6b3.lan (172.20.0.14): icmp_seq=2 ttl=64 time=0.384 ms
64 bytes from b16f7664f6b3.lan (172.20.0.14): icmp_seq=3 ttl=64 time=0.123 ms
64 bytes from b16f7664f6b3.lan (172.20.0.14): icmp_seq=4 ttl=64 time=0.438 ms

--- b16f7664f6b3.lan ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.123/0.546/1.241/0.418 ms
```

Bomba. Vše funguje, jak má.

Záznamy v lokální síti

Pokud se podíváme do záznamů na lokálním DNS, můžeme vidět, že DNSMASQ má v sobě NS (nameserver) záznamy pro všech 13 kořenových DNS serverů a zároveň jeden RRSIG záznam. RRSIG (RRSet Signature) je DNSSEC podpis množiny záznamů stejného typu (v tomto případě NS záznamy). `+norecurse` možnost slouží k tomu, aby mi záznamy vrátil přímo DNSMASQ server, protože bez ní jsem nedostal žádnou odpověď.

```
root@16de589c28a5:/# dig +dnssec +norecurse +noall +answer
.                       39350   IN      NS      e.root-servers.net.
.                       39350   IN      NS      h.root-servers.net.
.                       39350   IN      NS      l.root-servers.net.
.                       39350   IN      NS      i.root-servers.net.
.                       39350   IN      NS      a.root-servers.net.
.                       39350   IN      NS      d.root-servers.net.
.                       39350   IN      NS      c.root-servers.net.
.                       39350   IN      NS      b.root-servers.net.
.                       39350   IN      NS      j.root-servers.net.
.                       39350   IN      NS      k.root-servers.net.
.                       39350   IN      NS      g.root-servers.net.
.                       39350   IN      NS      m.root-servers.net.
.                       39350   IN      NS      f.root-servers.net.
.                       39350   IN      RRSIG   NS 8 0 518400 20231215170000
20231202160000 46780 . FtJeNzHnmMPThtmPrx8/OGm3tlNFj9B+KL1Q1LHSvBOwQ9Ybi4j8kYyF
1cXxUzjXPArEycfO2IpLSkDTpzuxMyKm/1LMXQuqfGLG9i4AArkwY+c3 7KF/eh6KsiRKIGFr8iFNim
/TOL0NaqgMx95FC1Q4PTB8r/5CWe7pvT7F
DpEJD3ZTx2TjcwzA4Cvs2aJJBFFZM7dsSifYBqE0JJV5YN3q8y4OxK0+
J1WKmC708bi2SsPgBWP7C23pm47d5DO+TOQaEuVerHn54sv+Cm1fGVVw
kPCMF/TBIRGF3niIjvtVNiUNAAELL/3Bt1E9Hj6vV89v91nx+OHaifJW AFpaKA==
```

Záznamy v síti TUL

Pro ověření záznamů přímo ze sítě jsem se přes SSH přihlásil na stanici A0320 a prověřil záznamy:

```
kevin.danek@a0320:~$ dig +dnssec +noall +answer
.                       7081    IN      NS      c.root-servers.net.
.                       7081    IN      NS      i.root-servers.net.
.                       7081    IN      NS      l.root-servers.net.
.                       7081    IN      NS      e.root-servers.net.
.                       7081    IN      NS      j.root-servers.net.
.                       7081    IN      NS      m.root-servers.net.
.                       7081    IN      NS      b.root-servers.net.
.                       7081    IN      NS      a.root-servers.net.
.                       7081    IN      NS      f.root-servers.net.
.                       7081    IN      NS      k.root-servers.net.
.                       7081    IN      NS      g.root-servers.net.
.                       7081    IN      NS      d.root-servers.net.
.                       7081    IN      NS      h.root-servers.net.
```

```
kevin.danek@a0320:~$ dig +dnssec +noall +answer www.tul.cz
www.tul.cz.             1098    IN      CNAME   novy.tul.cz.
novy.tul.cz.            1098    IN      A       147.230.18.195
```

Ověřování lokálních DNSSEC záznamů

Další úlohou by mělo být přídání Unbound DNS serveru pro ověřování DNSSEC záznamů, ovšem když se podíváme na záznamy z lokální sítě, tak tam jsou RRSIG záznamy, které, jak jsem již psal, jsou výsledkem fungování DNSSEC. Ačkoliv bych mohl ještě přidat Unbound DNS jako upstream DNS pro DNSMASQ, jaksi to postrádá smysl, když už validace DNSSEC záznamů probíhá v DNSMASQ.

```
root@16de589c28a5:/# dig DNSKEY www.tul.cz +noall +answer +authority
www.tul.cz.             608     IN      CNAME   novy.tul.cz.
tul.cz.                 964     IN      SOA     bubo.tul.cz.
pavel\.satrapa.tul.cz. 2021054787 7200 3600 2419200 1200
root@16de589c28a5:/# dig DNSKEY www.tul.cz +noall +answer +authority +multi
www.tul.cz.             259 IN CNAME novy.tul.cz.
tul.cz.                 1072 IN SOA bubo.tul.cz. pavel\.satrapa.tul.cz. (
                                2021054787 ; serial
                                7200       ; refresh (2 hours)
                                3600       ; retry (1 hour)
                                2419200    ; expire (4 weeks)
                                1200       ; minimum (20 minutes)
                                )
```

Závěr

V tomto cvičení jsem vytvořil lokální síť s fungujícím DHCP serverem pomocí DNSMASQ v Dockeru. S využitím Docker compose jsem nakonfiguroval DNSMASQ tak, aby obsluhoval dva testovací klienty ve vlastní síti. Zároveň mi DNSMASQ posloužil jako DNS server pro lokální doménu a pro ověřování DNSSEC záznamů. Díky použití dockeru byla práce svižná a neměla žádný overhead ve formě práce s virtuály, navíc jsem si do této testovací sítě mohl přinést cokoliv jsem potřeboval.

Příloha 1: docker-compose.yaml

```
version: '3'
name: pbe_7_dns_dnssec
services:
  dnsmasq:
      container_name: "dnsmasq"
      image: "drpsychick/dnsmasq:latest"
      privileged: true
      networks:
      pbe:
      ipv4_address: 172.20.0.5
      environment:
      - DMQ_DHCP_DNS=dhcp-option=option:dns-server,172.20.0.5,8.8.8.8,8.8.4.4
      - DMQ_DHCP_GATEWAY=dhcp-option=3,172.20.0.1
      - DMQ_DHCP_RANGES=dhcp-range=172.20.0.10,172.20.0.100,24h
      - DMQ_DNS_DOMAIN=domain=lan
      - DMQ_DNS_FLAGS=expand-hosts\ndomain-needed\nselfmx\ndns-loop-detect
      - DMQ_DNS_LOCAL=local=/lan/
      - DMQ_DNS_RESOLV=no-resolv
      - DMQ_DNS_SERVER=server=172.20.0.6\nserver=8.8.8.8\nserver=8.8.4.4

  client_1:
      container_name: "client_1"
      build:
      context: .
      dockerfile: client.dockerfile
      tty: true
      privileged: true
      stdin_open: true
      depends_on:
      - dnsmasq
      networks:
      - pbe

  client_2:
      container_name: "client_2"
      build:
      context: .
      dockerfile: client.dockerfile
      tty: true
      privileged: true
      stdin_open: true
      depends_on:
      - dnsmasq
      networks:
      - pbe

networks:
  pbe:
      ipam:
      config:
```

```
  - subnet: "172.20.0.0/16"
```

Příloha 2: client.dockerfile

```
FROM ubuntu
RUN apt update
RUN apt install -y net-tools dnsutils nano iputils-ping iproute2 dhcpcd5

# Vytvoření entrypoint scriptu
RUN touch /entrypoint.sh
RUN echo "#!/bin/bash" >> /entrypoint.sh
RUN echo "ifconfig eth0 0.0.0.0" >> /entrypoint.sh
RUN echo "dhcpcd" >> /entrypoint.sh

CMD /bin/bash /entrypoint.sh; /bin/bash
```