Vysoké učení technické v Brně
Fakulta informačních technologií

Síťové aplikace a správa sítí –

Filtrující DNS resolver

25. října 2020 Jakub Sekula

Obsah

1 IPV4 a IPV6

Pro komunikaci mezi klientem a serverem je využit IPV6 socket. Většina operačních systému totiž podporuje příjem ipv4 paketu ipv6 socketem. Tzn. pokud by server bežel na operačním systému, který toto nepodporuje, tak server nebude fungovat. Server nepoužívá pouze 1 socket, ale používá ještě jeden pro komunikaci s DNS serverem. Zda-li bude socket ipv4, nebo ipv6, rozhuduje parametr -s, respketive funkce getaddrinfo(), kterou volám pro zjištění ipv6 adresy serveru a pokud funkce nevrátí ipv6 adresu, tak volám znovu, ale tentokrát pro ipv4.

2 DNS

DNS je hierarchický systém doménových jmen. Ten realizují právě DNS servery spolu s protokolem stejného jména, který využívají k výměně informací. DNS je v praxi překladová služba, která číselnou IP adresu přeloží do podoby zvolené domény. Na IP se ptá DNS serveru[1].

3 DNS packet

Veškerá komunikace v domain protokolu je přenášená pomocí zpáv, které mají následující podobu[2].

```
+---------------------+
|        Header       | hlavička
+---------------------+
|       Question      | dotaz
+---------------------+
|        Answer       | odpověď
+---------------------+
|      Authority      | autorita( autorizovaná odpověď )
+---------------------+
|      Additional     | dodatečné informace
+---------------------+
```

3.1 Hlavička

Hlavička paketu je přítomna vždy. Hlavička zahrnuje pole, které značí, z jakých dalších částí se paket skládá a dále také, jestli je paket dotaz, nebo odpověď a další[2]. Podoba hlavičky je následující:

- ID - 16 bitový identifikátor paketu
- QR - 1 bit značíčí, jestli je paket typu 0 (dotaz), nebo 1 (odpověď)
- Opcode - 4 bity pro označení typu dotazu
- AA - 1bit Odpověď autority, tedy serveru odpovědného serveru
- TC - 1 bit Indikuje zda byl paket zmenšen
- RD - 1 bit Indikuje, zda-li je vyžadována rekurze

- RA - 1 bit Podpora rekurze
- Z - 4 bity Rezerva pro budoucí použití
- RCODE - 4 bity Response code
- QDCOUNT - 16 bitů Specifikujících počet vstupů v dotazu
- ANCOUNT - 16 bitů Specifikujících počet zdrojových záznamů v odpověďi
- NSCOUNT - 16 bitů Počet zdrojových záznamů
- ARCOUNT - 16 bitů Počet zdrojových záznamů v přídavné sekci

3.2 Dotaz

Část paketu používaná pro přenos dotazu a parametrů s ním spojených[2].

- QNAME - 16 bitů Název domény
- QTYPE - 16 bitů Typ dotazu
- QCLASS - 16 bitů Třída dotazu

3.3 Odpověď

Část paketu používaná pro přenos odpověďi a parametrů s ní spojených[2].

- NAME - 16 bitů Název domény spojené se zdrojovým záznamem
- TYPE - 16 bitů Typ dotazu
- CLASS - 16 bitů Specifikuje význam dat v RDATA
- TTL - 16 bitů Čas po jakou dobu má být zaznám cachován před vymazáním
- RDLENGTH - 16 bitů Délka oktetů v RDATA
- RDATA - Proměnná délka oktetových stringů popisujících zdroj

3.4 Authority

Sekce obsahuje zdrojové záznamy odkazující na autoritativní server[2].

3.5 Additional

Obsahuje zdrojové záznamy, které jsou spjaté s dotazem, ale nejsou odpovědí na tento dotaz[2].

4 Návrh a implementace aplikace

4.1 Překlad programu

Zdrojové kódy k programu se nachází ve složce src. V souboru, nad složkou src, se nachází makefile, pomocí kterého lze pracovat s aplikací. Příkazem make se přeloží projekt a vygeneruje se program dns. Příkazem make clean se program dns smaže. Makefile také zná příkaz make docu, který vytvoří doxygen dokumentaci k projektu.

4.2 Ukázka spouštění

Program se spouští následujícím způsobem:

./dns -s 1.1.1.1 -p 8080 -f blacklist

./dns -s 8.8.4.4 -f blacklist.txt

4.3 Parametry programu

Program umí zpracovat 3 parametry. Jsou jimi:

- -s: IP adresa nebo doménové jméno DNS serveru (resolveru), kam se má zaslat dotaz. POZOR !! Pokud nebude zadán DNS server, tak server bude posílat požadavky na parametrem zadaný server a je mu jedno, že se nejedná o DNS ! Tím pádem se klient nikdy nedočká odpovědi.
- -p port: Číslo portu, na kterém bude program očekávat dotazy. Výchozí je port 53. (volitený parametr)
- -f filter_file: Jméno souboru obsahující nežádoucí domény.

Parametry zpracovává funkce `getArguments()`

4.4 Tělo programu

Aplikace dns běží jako server. Po spuštění programu se vytvoří fukcí `socket()` komunikační soket, přiřadí se mu adresa pomocí funkce `bind()` a následně se v cyklu čeká na přijetí paketu na adrese localhost na portu zadaném parametrem programu `-p`.

Jakmile je zachycen paket na dané adrese a portu, tak se hlavní aplikační proces rozdělí funkcí `fork()`. Zatímco child proces získavá doménové jméno z paketu a další informace, hlavní proces se vrátí a čeká na přijetí dalšího paketu.

4.5 Child proces

Jakmile je paket přijat, zpracovává se v child procesu. V tomto procesu program získá dotazované doménové jméno QTYPE z paketu, aby si ověřil, zda se opravu jedná o dotaz typu A.

Dále program prochází zadaný soubor domén a hledá, zda-li dotazovaná doména není na tomto seznamu nebo jestli není poddoménou na tomto seznamu.

Případnou zakázanou doménu vrací klientovi jako původní paket s nastaveným `rcode` flagem na 5 a flagem `qr` na 1. Obdobně je to pro případ jiného typu dotazu než je typ A, respketive s `rcode` nastaveným na 4.

V případě, že tyto kontroly proběhnou v pořádku, je paket přesměrován na DNS server zadaný parametrem programu `-s`. Parametr -s se předává funkci `getDnsIp()`, která vrací ip adresu DNS serveru. Po získání ip adresy je paket funkcí `sendto()` poslán na port 53 DNS serveru. Příchod odpověďi je očekáván funkcí `recvfrom()` a jakmile tento paket dorazí, je poslán nazpět klientovi.

4.6 Návratové kódy

- 9 - Chyba soubor neexistuje
- 10 - Chyba vstupních parametrů
- 11 - Chyba získávání ip adresy DNS serveru
- 12 - Chyba vytvoření soketu
- 13 - Chyba přidělování adresy soketu

5 Testování

Korektní funkčnost aplikace jsem testoval pomocí nástroje nslookup a dig. Testy fungovaly následovně. V jednom okně terminálu jsem pustil svůj program a ve 2. okně terminálu jsem pustil nslookup takto: `"nslookup -port=8080 -type=a doména localhost"` a takto dig: `"dig @localhost -p 8080 doména"` respektive `"dig @ip6-localhost -p 8080 doména"` a pak jsem porovnával ip adresu, která přisla zpět nslookupua digu s online resolv nástrojem DNS Checker. Případně jsem zadal ip adresu do prohlížeče. Takto jsem iteroval přes různé domény. Pro testování jsem také napsal skript test.py. Skript se pouští příkazem python3 test.py

5.1 Testování typu dotazu

Korektnost ošetření dns dotazu typu A jsem testoval tak, že jsem pustil nslookup bez parametru -type=a. Tím pádem se pošle první ipv4 paket a následně i ipv6 paket. V tomto případě nebyl QTYPE z hlavičky paketu nastaven na 1 ale na 28. V situaci kdy přišel ipv6 paket, dns program reagoval korektně a poslal zpět paket s nastaveným `rcode` a `qr` a dále neposílal paket na dns resolver. Toto jsem kontroloval programem Wireshark, který celou komunikaci odchytával.

5.2 Testovaní blacklistových domén

Testování blacklistových domén probíhalo tak, že jsem si do souboru zapsal několik domén. Doménami byly kupříkladu facebook.com, docs.google.com a wis.fit.vutbr.cz a na bežící program dns jsem pomocí nslookup posílal domény, které jsem čekal, že projdou, tedy například face.com, google.com, google.cz, vutbr.cz, docs.google.cz apod. Následně jsem posílal domény, u kterých jsem čekal, že neprojdou, tedy domény jako google.com, wis.wis.fit.vutbr.cz atd. Celou síťovou komunikaci jsem sledoval programem Wireshark. A pro každou zaslanou doménu na server, jsem kontroloval odchozí paket na klienta a díval se na `rcode`a `qr` flagy v hlavičče.

Reference

[1] BEST-HOSTING s.r.o., i.-h.: Co je to DNS server? 2020. Dostupné z: https://best-hosting.cz/cs/napoveda/co-je-to-dns-server

[2] Mockapetris, P.: Domain names - implementation and specification. Nov 1987: str. 1–55, doi:10.17487/rfc1035. Dostupné z: https://tools.ietf.org/html/rfc1035