VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí

Tunelování datových přenosů přes DNS dotazy

9. november 2022 Andrej Pavlovič (xpavlo14)

Obsah

1 Úvod

Tento text dokumentuje prácu do predmetu ISA, ktorá implementuje jak klientskú, tak serverovú časť aplikácie v jazyku C, určenej pre tunelovanie dát pomocou DNS systému. Server je aplikácia načúvajúca na implicitnom porte pre DNS (53) na prichádzajúce spojenia a dáta od klienta a klient nástroj na enkapsuláciu týchto dát do validných DNS paketov.

2 Systém DNS

DNS (Domain Name System) je globálny distribuovaný systém, určený primárne pre prevody doménových mien na ich IP adresy. Realizovaný je pomocou DNS serverov a protokolu DNS.

Tento systém je možné zneužiť útočníkmi okrem iného napríklad na tzv. *data exfiltration attack*, čo je typ útoku využívaný pre exfiltráciu údajov z interných, alebo obmedzených sietí. Z princípu, akým tento systém funguje, je nemožné týmto útokom s určitosťou predísť.

DNS protokol sa používa na tri hlavné účely, podľa čoho sa typ paketov delí na:

- ***otázky*** pre žiadosť o preklad doménového mena, *(klient –>server)*,
- ***odpovede*** pre odpoveď na žiadosť o preklad doménového mena, *(server –>klient)*,
- ***prenos zónových súborov*** pre aktualizovanie dát medzi servermi, *(server –>server)*.

3 Návrh

Aplikácia prenáša dáta pomocou DNS nad **TCP protokolom**, ktorý sa stará o spoľahlivé doručenie na transportnej vrstve. Prenos jedného súboru je uskutočnený vždy v jednej *TCP session* a teda klient posiela v rámci jedného TCP spojenia viacero DNS paketov (pokiaľ nie je vstupný súbor prázdny – vtedy pošle iba jeden a tým je cesta, pod ktorou tento súbor uložiť). Pre prenos dát sú využité *DNS standard query*, teda žiadosti o preklad doménového mena. Dáta sú klientom umiestnené pred doménové meno v tvare *"nejake.data.domenove.meno.com"*. Tieto si potom pomocou DNS serverov nájdu cestu do danej domény, kde ich server extrahuje.

DNS standard query obsahujú vždy práve jednu otázku. Ako identifikačné číslo v DNS hlavičke sa použije identifikačné číslo procesu klienta.

Server klientovi neodosiela žiadne DNS pakety. Po naviazaní TCP spojenia pošle klient serveru cestu, pod ktorou má byť súbor uložený, obsiahnutú v jednom DNS pakete. Maximálna dĺžka prenesenej cesty je teda *249 - <dĺžka bázovej domény>*. Nasleduje prenos súboru, ktorý sa podľa potreby kúskuje do paketov podľa rovnakej maximálnej dĺžky ako cesta, po ktorom sa klient úspešne ukončí a server sa prepne do stavu čakajúceho na ďalšie spojenia. **Server teda funguje iteratívne**.

Ak súbor pod danou cestou existuje, je prepísaný. Aplikácia sa vždy pokúsi vytvoriť adresárovú štruktúru obsiahnutú v ceste, ak táto štruktúra neexistuje. To platí pre cestu ku súboru od klienta a aj adresárovú cestu zadanú na strane serveru. Prijatý súbor je na serveri uložený pod spojenou cestou v tvare <server_path>/<client_path>.

Dáta nie sú šifrované. Na kódovanie sa používa vlastná implementácia **Base16** so skupinou znakov *"A-P"* čo sú validné znaky DNS mena [todo]. Tieto dáta sú následne delené do tzv. *labels* o maximálne 63 znakoch, oddelené bodkou.

4 Implementácia, odovzdané súbory a ich hierarchia

Zdrojové texty projektu pod zložkou `src/`. Tie sa ďalej delia na klientské pod zložkou `src/sender/`, serverové pod zložkou `src/receiver/`. Spoločné moduly potom pod zložkou `src/common/`. V koreňovom

adresári sa nachádza `Makefile` pre preklad projektu, `README` so základnými informáciami o projekte a `manual.pdf` obsahujúci túto dokumentáciu projektu.

Projekt implementuje dve samostatné aplikácie – server (názov spustiteľného súboru `dns_receiver`) a klienta (názov spustiteľného súboru `dns_sender`).

4.1 Chybové stavy

O spracovanie chýb sa stará modul `common/err.h`, ktorý je využívaný funkciami naprieč projektom. Ak nastane chyba, aplikácia informuje užívateľa na štandardný chybový výstup a v závislosti na type chyby sa program buď ukončí, alebo pokračuje.

Tento modul využíva štandarného chybového spracovania v jazyku C za pomoci modulu `errno.h` a ďalších. Užívateľovi by vždy mala byť ponúknutá dostatočne informatívna hláška o chybe.

Pri kritickej chybe program končí s návratovou hodnotou 1.

4.2 Udalosti

V rámci dodržania implementačných rozhraní `src/receiver/dns_receiver_events.h` a `src/sender/dns_sender_events.h` dodaných vyučujúcim bol vytvorený samostatný modul `src/common/events.h`. Tento definuje primárne dátovú štruktúru `event`, ktorá sa potom definuje globálne na klientovi aj na serveri.

5 Použitie

5.1 Preklad

Preklad celého projektu je možný pomocou súboru `Makefile`, klasicky príkazom **`make`** v koreňovom adresári, ktorý invokuje cieľ **`make all`**. `Makefile` definuje nasledujúce použiteľné ciele:

- **`all`** preloží celý projekt. Spustiteľné súbory `dns_sender` a `dns_receiver` prekladá pod adresár `app/`. Nezlinkované medzisúbory prekladu sú ukladané pod adresár `build/`.
- **`sender`** totožný s **`all`** s rozdielom, že prekladá iba klienta,
- **`receiver`** totožný s **`all`** s rozdielom, že prekladá iba server,
- **`clean`** odstránenie celého výsledku prekladu,
- **`clean_build`** odstránenie preložených nespustiteľných medzisúborov.

5.2 Spustenie a ukončenie

5.2.1 Klient

Spustenie:

dns_sender [-u UPSTREAM_DNS_IP] [-s MILLISECONDS] BASE_HOST DST_FILEPATH [SRC_FILEPATH]

Príklad spustenia:

$ dns_sender -u 127.0.0.1 -s 0 example.com receive.txt ./send.txt

Ak sa klient spustí bez vstupného súboru, číta dáta zo štandardného vstupu. Toto chovanie je podľa klasických unixových zvyklostí a teda ak je vstup nastavený na štandardný vstup, no tento je prázdny, program čaká na vstup.

Pri bezproblémovom behu je klient ukončený po úspešnom prenesení súboru. Ak je klient ukončený násilne zo strany užívateľa, systému (toto by nastať nemalo), alebo pri narazení na chybu POČAS prenášania súboru, server sa nedozvie o tom, že klient bol ukončený nekorektne a neúplne prenesený súbor uloží orezaný.

5.2.2 Server

Spustenie:
    dns_receiver BASE_HOST DST_DIRPATH
Príklad spustenia:
    $ dns_receiver example.com ./data
Server ukončí prácu pri zaslaní SIGINT signálu jeho procesu.

6 Testovanie

Testovanie dokázalo korektnú funkcionalitu klienta aj serveru. Testovalo sa pomocou prenášania veľkého setu súborov rôznej veľkosti (od 50 bajtov po 100MB), rôznych formátov s rôznym počtom opakovaní a následným porovnaním prijatého súboru s odoslaným pomocou nástroju **diff**. Server sa testoval novým spustením pred každým odoslaným súborom alebo naopak aj jedným behom. Komunikácia medzi klientom a serverom bola zachytávaná a analyzovaná pomocou programu **wireshark**.

Pre overenie, že odosielané pakety sú valídne DNS otázky, boli využité rôzne DNS servery v sieti. Kontrolovala sa správna syntax paketov opäť programom wireshark a odpovede od týchto serverov.

6.1 Meranie

V rámci testovania bolo vykonané aj meranie rýchlosti implementácie. Toto meranie nebolo nijak rozsiahle, skôr naopak obmedzené. Hlavným cieľom bolo zistiť, či daná implementácia dokáže využiť šírku pásma dnešnej priemernej siete na plno. Inými slovami testovalo sa, či daná implementácia nie je pomalšia ako možnosti siete.

Ukázalo sa, že implementácia je pomerne rýchla. Prenosová rýchlosť z laptopu použitého na implementáciu na školský server eva.fit.vutbr.cz, interne v rámci KolejNetu (z kolejí) bola **$\tilde{2}$32Mbps** (bez spracovávania udalostí z vyučujúcim poskytnutých hlavičkových súborov).

Na otázku, či aplikácia dokáže využiť pásmo na plno sa ťažko odpovedá, no s určitosťou sa dá povedať, že je relatívne rýchla.

Za zmienku by stál aj maximálny objem jedného DNS paketu. Ten je v tejto implementácií daný na

(251 - <dĺžka bázovej domény>) / 2 bajtov; zaokrúhlené nadol.

Toto je maximály počet bajtov, ktorý je možné preniesť v jednej sekcií mena v otázke pomocou jedného DNS paketu obsahujucého jednu otázku. Keďže sa ale pakety prenášajú nad TCP, neznamená to, že každý DNS paket je prenesený ako samostatný paket po sieti. TCP je streamovací protkol, ktorý môže pakety deliť alebo spájať. Predovšetkým toto spájanie napríklad pomocou *Naglevho algoritmu* často túto implementáciu dosť urýchluje.

7 Rozšírenia

Klient bol rozšírený o prepínač `-s` s hodnotou milisekúnd. Táto hodnota udáva, koľko milisekúnd má klient po úspešnom odoslaní súboru pozastaviť proces, pred ukončením spojenia s *FIN* (prípadne *RST*) príznakom. Východzia hodnota tohto parametru je 1 sekunda.

Keďže v tejto implementácií klient po úspešnom odoslaní súboru nečaká, no ihneď končí, v určitých prípadoch by mohlo nastať, že DNS server, s ktorým komunikuje (napríklad ak by tento server zisťoval požiadavky rekurzívne do relatívne hlboko zanorenej domény, cez veľa ne-rekurzívnych DNS serverov), ešte nestihol vybaviť a úspešne preposlať všetky DNS otázky od klienta. Ak by v takom prípade klient predčasne ukončil komunikáciu príznakom FIN, server by sa mohol rozhodnúť, že dané dotazy už nie je potrebné zodpovedať a tak by ich ďalej do siete neposielal.

7.1 Ďaľšie možné rozšírenia

7.1.1 Kódovanie

Pre jednoduchosť implementácie bolo využité kódovanie *Base16*. Toto sa intuitívne naskytuje ako jedno z prvých možných rozšírení. Z naštudovanej literatúry počas vypracovávania projektu by som ako autor odporučil, že jedny z najideálnejších efektívnejších kódovaní v tomto projekte by mohli byť **Base32**, alebo **Base64**.

7.1.2 Šifrovanie

Ďaľšie rozšírenie by mohlo byť prenášané dáta šifrovať, čo by zvýšilo bezpečnosť napríklad pri nelegálnom využití tohoto softwaru na tunelovanie dát (toto sa určite nebude diať :)), ale v závislosti od použitého šifrovania by sa mohla značne navýšiť náročnosť na sieť.

8 Známe nedostatky

Táto práca neobsahuje žiadne autorovi známe nesplnené časti zadania.

9 Prebrané časti kódu

Pri vypracovaní projektu bola použitá kratšia časť kódu z [1] podľa licencií.

Použitá literatúra

[1] Fallahi, F.: DNS Query Code in C with linux sockets. 2015. Dostupné z: https://gist.github.com/fffaraz/9d9170b57791c28ccda9255b48315168

[2] Mockapetris: RFC 1034. 1987. Dostupné z: https://www.ietf.org/rfc/rfc1034.txt