Tunelovanie dátových prenosov protokolom DNS

Patrik Skaloš

11. října 2022

Obsah

1 Úvod

1.1 Protokol DNS

DNS (*Domain Name System*) je protokol používaný na preklad doménového mena (napr. *www.example.com*) na IP adresu použiteľnú na prístup k obsahu uloženého pod touto doménou. Klient, ak chce k tomtu obsahu pristúpiť, musí najprv poslať požiadavku (*query*) DNS serveru, ktorý na požiadavku odpovie IP adresou, ak má v svojej databázi záznam so žiadaným doménovým menom. V opačnom prípade je potrebné požiadavku poslať DNS serveru vyššej úrovne.

1.2 Tunelovanie dát cez DNS

Tunelovanie dát protokolom DNS (ďalej DNS tunelovanie) je metóda enkapsulácie dát do DNS datagramu (paketu).

Klientské zariadenie môže mať ten najprísnejší firewall, no ak má povolené datagramy (pakety) DNS, využitím DNS tunelovania je možné medzi ním a serverom vytvoriť komunikačný kanál, keďže bude komunikácia vedená čisto cez požiadavaky a odpovede podľa protokolu DNS.

Príklad otázky v DNS požiadavke obsahujúcej nezakódované dáta - užívateľské meno *username* a heslo *password*: **username.password.example.com**

Poznámka: Podľa štandardu *RFC1035*[1] musí doménový štítok (*angl. label* - časť domény ohraničená bodkou, začiatkom alebo koncom domény) začínať písmenom, končiť písmenom alebo číslom a obsahovať iba písmená, čísla a pomlčku. To je jeden z dôvodov, prečo sa dáta enkapsulované v URL požiadavke enkódujú do formátu obsahujúceho iba povolené znaky.

1.3 Prečo DNS tunelovanie funguje

Ak chce klient používať protokol DNS, nemôže ho, samozrejme, zakázať. Jedinou ochranou pred DNS tunelovaním potom ostáva rozbaľovať DNS datagramy (pakety) a filtrovať ich podľa požiadavky. Môže však byť veľmi náročné určiť, či je požiadavka dôveryhodná alebo nie.

[1] RFC1035: https://www.ietf.org/rfc/rfc1035.txt

2 Impementácia

2.1 Klient

Klient (po spracovaní argumentov) prečíta a uloží všetky vstupné dáta zo štandardného vstupu alebo súboru, podľa argumentov spustenia. Tieto dáta následne zakóduje do formátu *base64* a je pripravený na ich odoslanie.

Na prenos dát klient používa iba protokol *UDP* (User Datagram Protocol), ktorý je známy svojou nespoľahlivosťou. V tomto protokole odosielateľ jednoducho pošle svoje dáta, pričom ich doručenie nie je zaistené a o doručení odosielateľ žiadne potvrdenie nedostane. Keďže je pre nás dôležité, aby serveru dáta prišli v poriadku (a všetky), bolo potrebné implementovať lepší mechanizmus prenosu dát. Tento mechanizmus je popísaný v podsekcii Komunikácia: Prenos dát medzi klientom a serverom.

Poznámka: Formát Base64 je sada 64 znakov (ktoré sa vojdú do 6 bitov) určená hlavne na zakódovanie dát, ktoré by mali obsahovať iba tlačiteľné znaky (*angl. printable characters*). Pozostáva z 26 malých a 26 veľkých písmen abecedy, desiatich cifier a znakov `+` a `/`.

2.2 Server

Server po spracovaní argumentov začína naslúchať na porte 53 (predvolený port pre DNS komunikáciu) a kým nie je program zastavený, v slučke prijíma datagramy od klienta. Po prijatí všetkých dát (zakódovaných do formátu *base64*) ich dekóduje a zapíše do výstupného súboru, ktorý je tvorený povinným parametrom *DST_FILEPATH* (ktorý značí priečinok, do ktorého sa dáta uložia) a relatívnou cestou prijatou od klienta. Proces komunikácie je podrobnejšie popísaný v nasledujúcej sekcii.

2.3 Komunikácia

2.3.1 Enkódovanie dát do DNS otázky

Podľa *RFC1035*, doména v DNS otázke môže obsahovať až štyri štítky, z ktorých dva posledné štítky (doménové meno a doména najvyššej úrovne - *angl. extension*) sú definované parametrom (klienta aj serveru) *BASE_HOST*. Ostávajú nám teda dva štítky pre naše dáta, pričom ďalej *RFC1035* stanovuje, že štítok môže mať maximálnu dĺžku 63 bajtov. Do jedného DNS datagramu nám teda vojde 126 bajtov. Keďže dáta kódujeme do formátu *base64*, ktorý dáta nafúkne na $\frac{4}{3}$ pôvodnej veľkosti, do datagramu sa vojde približne 94 bajtov užívateľských dát.

Dva príklady enkódovania dát do *base64* a do doménového mena pri parametri *BASE_HOST = xskalo01.com*:

- – Uživateľské dáta: `Ahoj`
  – Base64: `QWhvag`
  – Doména v DNS otázke: `QWhvag.xskalo01.com`
- – Uživateľské dáta: `0123456789012345678901234567890123456789012345678901234567890123456789`
  – Base64: `MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODk`
  – Doména v DNS otázke: `MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY .3ODk.xskalo01.com`

2.3.2 Prenos dát medzi klientom a serverom

Do prvej správy jednej komunikácie sa vždy zakóduje iba relatívna cesta udávajúca, kam má server uložiť dáta, ktoré nasledujú. Predpokladáme, teda, že táto cesta (zakódovaná) bude kratšia ako 126 bajtov, čo v niektorých prípadoch nemusí byť dosť, no nám to postačí.

Do nasledujúcich datagramov sa vkladajú zakódované dáta - vždy maximálne 126 bajtov (znakov). Server tieto dáta postupne ukladá do RAM a dekóduje ich až na konci komunikácie.

Posledná správa, značiaca koniec komunikácie, neobsahuje žiadne zo zakódovaných dát. Doména v DNS otázke poslednej požiadavky je *a.a.BASE_HOST*, kde *BASE_HOST* je parameter spustenia. Takáto otázka je ľahko rozoznateľná otázok s dátami, keďže má dva doménové štítky (nepočítajúc *BASE_HOST*) a prvý z nich nie je plný, čo by sa u otázky so zakódovanými dátami nestalo. Tento spôsob nie je efektívny a zvolili sme si ho iba kvôli jednoduchosti implementácie.

Ako sme už naznačili, UDP nie je spoľahlivý komunikačný protokol. Z toho dôvodu server na každú prijatú správu klientovi odpovedá za účelom potvrdenia prijatia rovnakým datagramom, ako prijal, s jedinou zmenou - v hlavičke DNS požiadavke nastaví príznak *odpoveď* (*angl. response*). Časový limit na obdržanie potvrdenia o prijatí je nastavený na 100 ms. V prípade, že klient neobdrží potvrdenie o prijatí, komunikácia musí byť ukončená a začatá odznovu. Aby sa klient nezasekol pri odosielaní dát nenaslúchajúcemu serveru, použili sme premennú *MAX_TRIES*, ktorá je štandardne

nastavená na 3 a definuje počet pokusov na odosielanie jedného súboru a zároveň počet pokusov na odoslanie datagramu ukončujúceho komunikáciu.

Poradie komunikácie v prípade, že klientovi nie je doručené potvrdenie o prijatí:

1. Klient odošle datagram serveru a nedostane odpoveď
2. Klient sa pokúsi predčasne uzavrieť komunikáciu odoslaním požiadavky bez dát (ukončujúca požiadavka)
3. Ak sa uzavretie komunikácie nepodarilo (server nepotvrdil prijatie, resp. potvrdenie nebolo klientovi doručené) a o uzavretie komunikácie sa program pokúsil menej ako *MAX_TRIES* krát, pokračuje krokom 2.
4. Ak sa uzavretie komunikácie nepodarilo ani na posledný pokus, kanál medzi klientom a serverom je nefunkčný, prenos dát zlyhal a program klienta je ukončený s chybovou hláškou.
5. Ak sa predčasné uzavretie komunikácie podarilo, môžeme sa znovu pokúsiť odoslať dáta serveru. Začína sa znovu, od prvej požiadavky obsahujúcej cieľovú cestu. V prípade, že prenos dát zlyhal už *MAX_TRIES* krát, program klienta je ukončený s chybovou hláškou.

2.4 Nedostatky, obmedzenia a rozšírenia

Nasleduje neúplný zoznam nedostatkov, obmedzení a možných rozšírení našej implementácie (väčšina z nich vyplýva zo zamerania sa na jednoduchosť):

- Kódovanie dát do formátu *base64* znamená, že (len v prípade kódovania binárnych dát) sa v DNS otázke môžu vyskytovať znaky `+` a `/`, ktoré však podľa *RFC1035* v doméne nie sú povolené. Takéto DNS požiadavky sme pri testovaní odosielali aj na bežné DNS servery, ktoré s tým, kupodivu, nemali problém. Z toho dôvodu sme implementáciu neprarábali a naďalej kódujeme do *base64*, no určite by bolo dobrým nápadom kódovať do formátu, ktorý obsahuje iba písmená (veľké aj malé), čo by bez problémov vyhovovalo štandardu. Našim nápadom bolo implementovať vlastný *base48*, ktorý by dáta enkódoval len na malé a veľké písmená abecedy. Riešilo by to viac problémov nášho projektu s *RFC1035*
- Dáta sú síze zakódované, no nie sú zašifrované! Ktokoľvek, kto odchytí naše datagramy by dokázal vcelku ľahko dekódovať dáta, ktoré odosielame.
- Odhalenie toho, že datagramy, ktoré posielame, nie sú reálne DNS požiadavky by bolo vcelku jednoduché, keďže v každej DNS otázke využívame priestor do plnej miery (maximálnych 126 bajtov v doméne) a potvrdenia o prijatí sú nezmyselné.
- Prenos veľkých súborov, alebo aj menších súborov po menej stabilnej môže ľahko zlyhať z dôvodu použitia UDP protokolu a iba jednoduchého mechanizmu zaistenia prenosu.
- Prenos dát je z dôvodu veľkej réžie neefektívny.
- Dĺžka reťazca označujúci relatívnu cestu pre uloženie súboru môže byť dlhá maximálne (približne) 94 bajtov (znakov).
- Celý odosielaný súbor či prijaté dáta sú uložené do RAM, preto naša implementácia nie je vhodná pre prenos veľkých súborov.
- ...

3 Testovanie

Testovanie programov klienta a serveru prebiehali manuálne aj automaticky, na našom zariadení s operačným systémom *Debian 11*, pričom DNS datagramy boli posielané iba cez *localhost*.

3.1 Manuálne testovanie

Programy klienta aj serveru boli manuálne otestované s rôznymi argumentmi a vstupmi s rôznou dĺžkou aj typom (binárne aj textové dáta). Na kontrolu paketov cestujúcich po sieti sme použili program *Wireshark*.

3.2 Testovací skript

Na to, aby sme otestovali prenos čo najviac rôznych kombinácií dát, sme vytvorili jednoduchý skript v jazyku *python*. Skript generoval náhodné ASCII znaky náhodnej dĺžky, zapísal ich do súboru a spustil program klienta pre odoslanie tohto súboru. Po konci behu programu skript skontroluje, či je vstupný súbor identický so súborom vytvoreným serverom.

Reference

[1] P. Mockapetris (1987): Domain Names - Implementation and Specification (RFC 1035). On-line: https://www.ietf.org/rfc/rfc1035.txt

[2] *Silver Moon* (2020): DNS Query Code in C with Linux sockets. On-line: https://www.binarytides.com/dns-query-code-in-c-with-linux-sockets/

[3] *amitds* (2022): UDP Server-Client implementation in C. On-line: https://www.geeksforgeeks.org/udp-server-client-implementation-c/