VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí

Tunelování datových přenosů přes DNS dotazy

14. listopadu 2022 Simona Češková (xcesko00)

Obsah

1 Popis mechanismu pro tunelování datových přenosů prostřednictvím DNS dotazů

1.1 Co je to DNS

Hlavní právě, kterou DNS vykonává, je poskytnout informace pro zadané doménové jméno. Tyto informace jsou nejčastěji IP adresa. Dále umí poskytnout i pačně jméno pro které je IP adresa zaregistrovaná. S těmito dotazy se obrací zařízení na lokální DNS server, který má uložený adresy kořenových serverů ve své konfiguraci síťových parametrů. Obrací se na tyto kořenové servery s dotazem.

Získávání informací z takového systému probíhá rekurzí. Program zajišťující překlad (Resolver) postupuje od kořene stromu postupně směrem dolů. Dokud autoritativní záznam o hledané doméně nenajde. Na autoritativní DNS jej odkazují postupně další jednotlivé DNS servery pro jednotlivé části jména. [5]

1.2 Tunelování DNS

DNS protokol lze zneužít díky tomu, že není omezován a většinou ani kontrolován. To znamená, že jakékoliv zařízení v rámci sítě jej může používat. K DNS tunelování je nutné mít přístup k vnitřnímu DNS serveru s připojením k internetu. [3]

Tunelování díky DNS dotazů se může stát tehdy, pokud je počítač infikovaný nějakým `malwarem` a díky tomu útočník posílá dotazy (`firewall` je nezastaví) na DNS server s kterým je ve spojení. Skoro nelze dohledat zařízení útočníka, protože není přímo spojený s uživatelem, který má na svém zařízení tunelovací program. [2]

2 Popis návrhu a implementace klientské a serverové aplikace

2.1 Komunikační protokol mezi klientem a serverem

Komunikace mezi klientem a serverem probíhá na portu 53, protože DNS využívá právě port 53. Probíhá tak, že klient pošle dotaz na server, který čeká právě na dotaz. Jakmile na server přijde dotaz, tak pošle nazpět odpověď. Odpoveď je původní zpráva od klienta s pozměněnými parametry. Klient mezitím čeká na tuto odpověď, než začne posílat další dotaz. Tento proces se opakuje, dokud klient nedá serveru vědět, že chce spojení ukončit a zavře vytvořený socket.

2.2 Způsob kódování dat a informací

2.2.1 Způsob čtení dat

Funkce fread dovoluje čtení i ne textovým souborům.

2.2.2 Metoda kódování

Jako způsob kódování jsem zvolila `base32` systém. Vybrala jsem ho, protože má značné výhody např. oproti base64 kódovacímu systému. Mezi hlavní positiva, podle kterých jsem se rozhodovala patří: [4]

- je univerzální, podporuje systémy, které nerozlišují malá a velká písmena
- nesmí obsahovat `\`, proto může být použit i jako název souboru, což se velmi hodí vzhledem k tomu, že klient má parametr `DST_FILEPATH`, který obsahuje cestu pod kterou se data uloží na serveru

Jako algoritmus pro zakódování do `base32` jsem využila postup z [6], který jsem využila ve svých dvou funkcích `decodeCalculate()` a `encodeCalculate()`.

2.2.3 Způsob kódování

Výše zmíněnou funkci `encodeCalculate()` jsem zavolala s argumentem, který obsahuje `string` v kterém jsou uložená nějaká data s určitou délkou. Po zakódování se tato délka dat zvětšila na maximální velikost, kterou může obsahovat jedna část v `query`. Tato moje velikost je 56 bytů, protože 5 bytů zabere označení délky tohoto `stringu` (zápis je v hexadecimální soustavě - např. `\0x38`). Což po sečtení se rovná 61 bytům. Obecná maximální délka je 63 bytů, ale ve svém řešení jsem ji zkrátila o 2 byty kvůli větší přehlednosti a celistvosti.

Tato část algoritmu se dále opakuje pro další načtená data, dokud nenarazí na na maximální počet odeslaného `DNS name`, který se rovná 255 bytům. tato velikost 255 bytů se ale zkracuje podle délky domény, jejích ukazatelů délky a znaku konce `DNS name`. Pak se `packet` odešle a celý proces se zopakuje od začátku. [1]

2.3 Způsob ukládání souborů na serveru

Můj způsob implementace je, že první odeslaný `packet` nese pouze informaci o tom, kde se mají přenášená data na serveru uložit. Na začátku přenosu se otevře soubor, který zjistíme dekódováním poslaného `DST_FILEPATH` v prvním `packetu`. Poté se pošle odpověď a zahájí se přenost dalších dat. Na každý odeslaný `packet` od klienta, server odpoví a až poté se odesílá další, jakmile klient dostane od serveru odpověď.

Poslední packet který se odešle obsahuje informace o ukončení spojení mezi serverem a clientem.

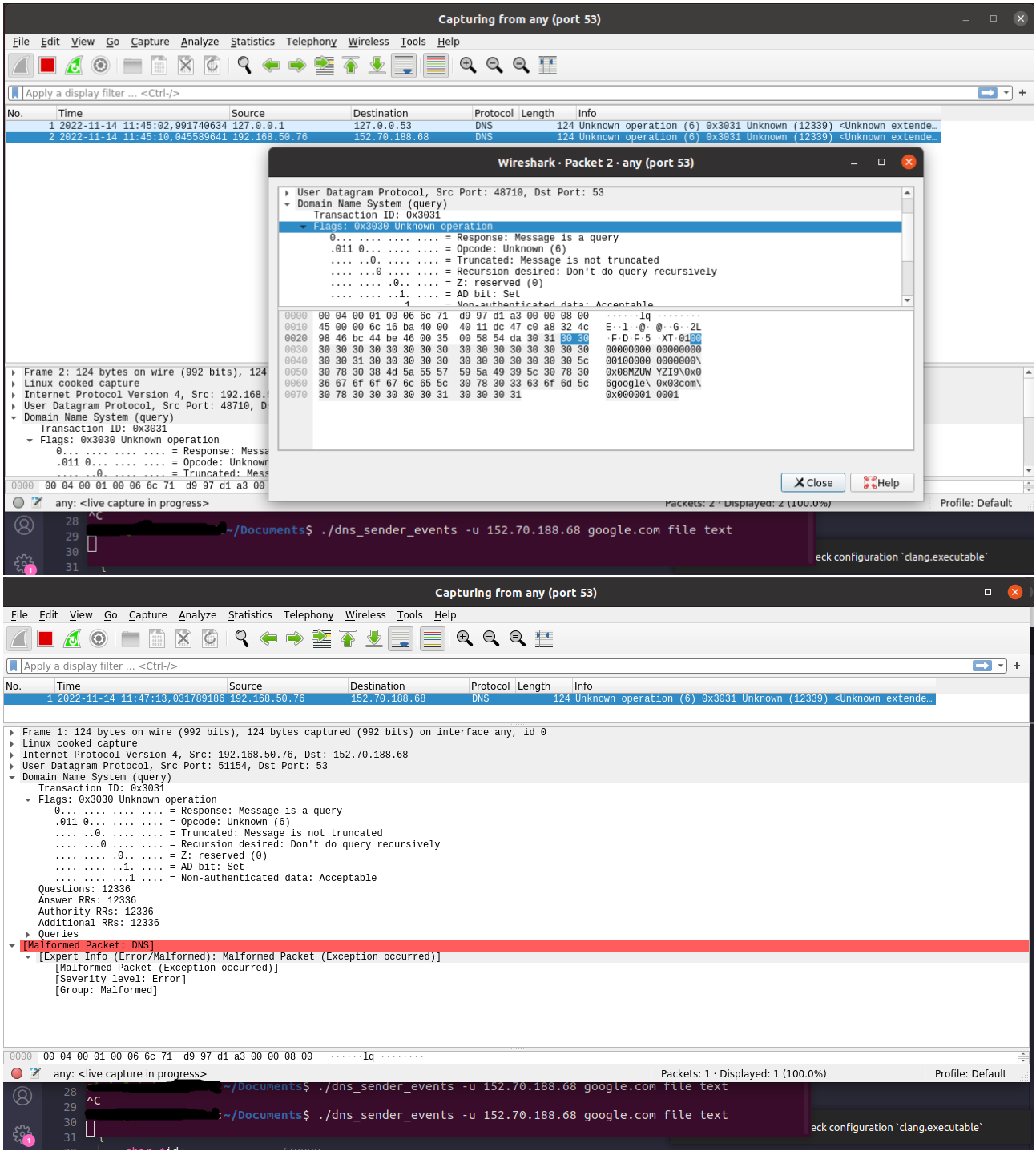
2.4 Možná rozšíření, omezení

Projekt by šlo rozšířit o ošetření posílání velkých souborů, aby se neztrácely packety. To by se dalo díky TCP spojení.

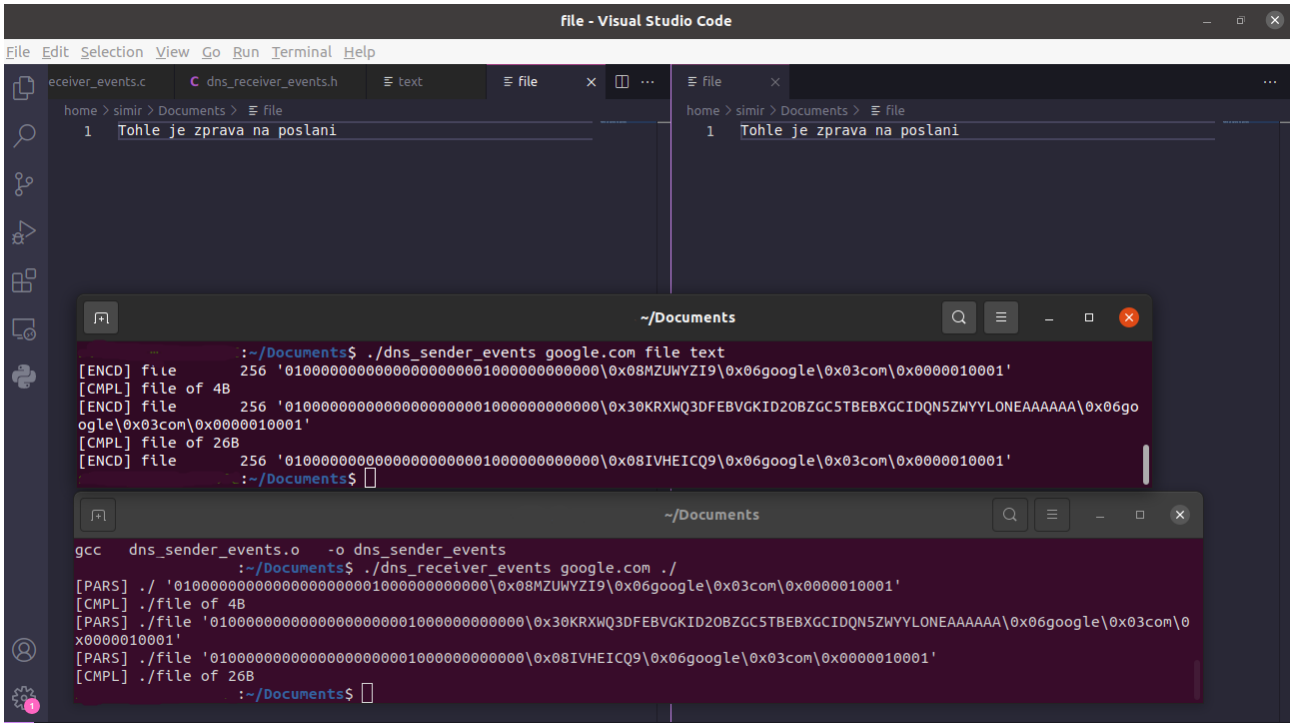
3 Popis testování a měření vytvořeného softwaru

3.1 Wireshark

Testování díky wiresharku, podle kterého jsem zjistila, že se mi správně nepřekládají poslané informace v hexadecimální soustavě. Řešením by byla úprava struktury na jiné datové typy, nebo jiný zápis do mnou zvoleného typu `char *`. Zvolila jsem tento typ, protože se s ním lépe pracuje a až později jsem si uvědomila, že to byla špatná volba. Bohužel se mi nepovedlo tento problém opravit, proto jsou přiložené obrázky s chybně sestavenými packetami.



Testování na `ubuntu 20.04`, kde lze vidět, že se správně pošle soubour file.

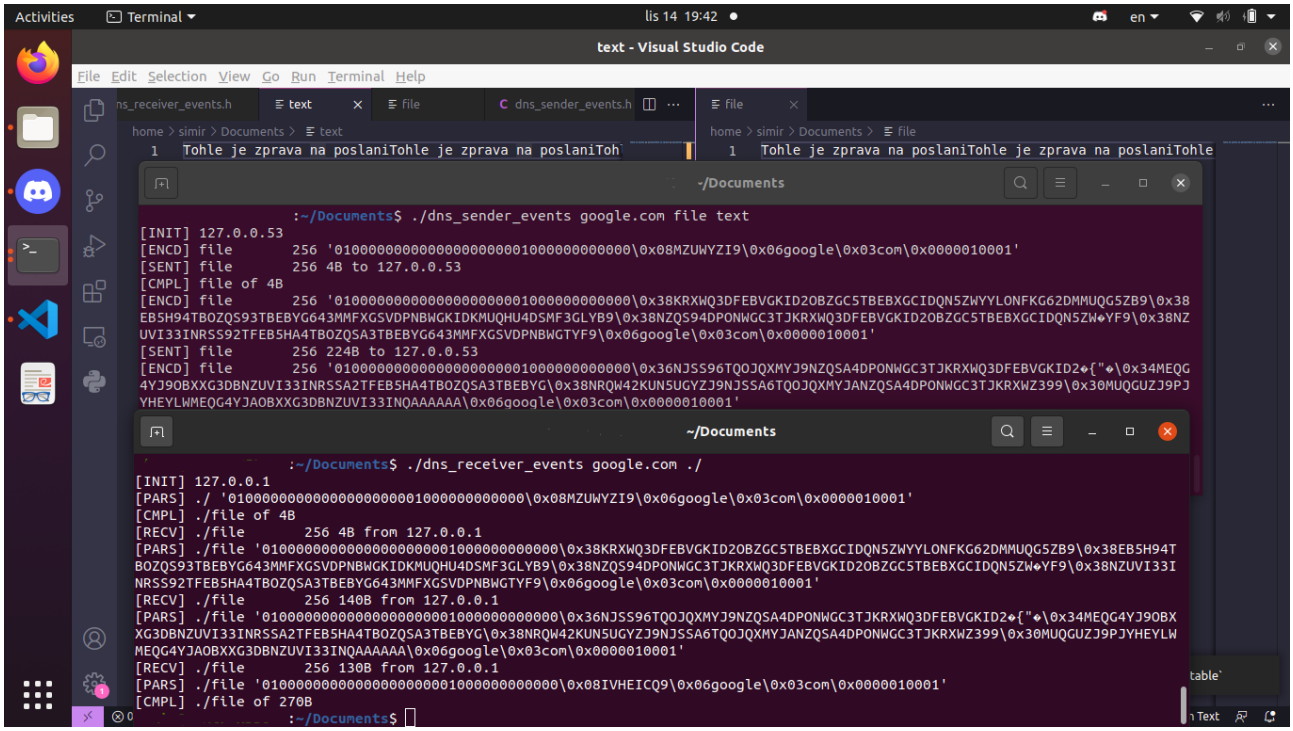

file - Visual Studio Code
File Edit Selection View Go Run Terminal Help
dns_receiver_events.h
text
file
home > simir > Documents > file
1 Tohle je zprava na poslani
~/Documents
:~/Documents$ ./dns_sender_events google.com file text
[ENCD] file 256 '0100000000000000000000010000000000000\0x08MZUWYZI9\0x06google\0x03com\0x0000010001'
[CMPL] file of 4B
[ENCD] file 256 '0100000000000000000000010000000000000\0x30KRXWQ3DFEBVGKID2OBZGC5TBEBXGCIDQN5ZWYYLONEAAAAAA\0x06google\0x03com\0x0000010001'
[CMPL] file of 26B
[ENCD] file 256 '0100000000000000000000010000000000000\0x08IVHEICQ9\0x06google\0x03com\0x0000010001'
:~/Documents$
~/Documents
gcc dns_sender_events.o -o dns_sender_events
:~/Documents$ ./dns_receiver_events google.com ./
[PARS] ./ '0100000000000000000000010000000000000\0x08MZUWYZI9\0x06google\0x03com\0x0000010001'
[CMPL] ./file of 4B
[PARS] ./file '0100000000000000000000010000000000000\0x30KRXWQ3DFEBVGKID2OBZGC5TBEBXGCIDQN5ZWYYLONEAAAAAA\0x06google\0x03com\0x0000010001'
[CMPL] ./file of 26B
[PARS] ./file '0100000000000000000000010000000000000\0x08IVHEICQ9\0x06google\0x03com\0x0000010001'
[CMPL] ./file of 26B
:~/Documents$

3.2 Testování skrz předem vytvořených funkcí k projektu

Čtyři funkce pro server/reciever a klient/sender, které byly přiloženy v zadání projektu jsem přidala do svého řešení. Bohužel, mi ale jejich popisy nepřišly jednoznačné, protože jsem se rozhodla je přiřadit do té části svého řešení, kam mně seděly nejvíce. Při volání funkce `dns_sender_on_chunk_encode` jsem do parametru `char *encodedData` vložila celý enkódovaný `string` dat, které se odešlou na server.

3.3 Posílání větších souborů

Při posílá větších souborů jsem při testování zjistila, že funkce `encodeCalculate()` pravděpodobně dělá menší chyby při enkódování. Nehledě na to ale, jsou takto větší soubory rozděleny do více poslaných packet, které na server přijdou, proto celý proces poslání packetů proběhl úspěšně.



3.4 Testování na virtuálce

Projekt byl otestován i na virtuálce, kde se přeloží a spustí. V hlavičkových souborech je nastavený port na 53, tak jak by měl být pro DNS komunikaci. Ale pro testování na svém počítači se to tak nedá a tak jsem ho nahradila dočasně portem 8080 pouze pro tyto účely.

Literatura

[1] Mertens, X.: DNS Query Length. [online], rev. 20. dubna 2017, [vid. 2022-11-11].
URL https://isc.sans.edu/diary/DNS+Query+Length+Because+Size+Does+Matter/22326

[2] Networks, P. A.: DNS Tunneling – co je to? [online], [vid. 2022-11-11].
URL https://nextgenfw.cz/2020/03/10/dns-tunneling-co-je-to/

[3] Whalebone: Skrytá hrozba v podobě tunelování DNS. [online], rev. 17. května 2022, [vid. 2022-11-11].
URL https://www.whalebone.io/post/skryta-hrozba-v-podobe-tunelovani-dns

[4] WIKIPEDIA: Base32. [online], rev. 10. říjen 2022, [vid. 2022-11-11].
URL https://en.wikipedia.org/wiki/Base32

[5] WIKIPEDIA: Domain Name System. [online], rev. 9. září 2022, [vid. 2022-11-11].
URL https://cs.wikipedia.org/wiki/Domain_Name_System#DNS_v_praxi

[6] Yang, D. H.: Base32 Encoding Algorithm. [online], rev. 2022, [vid. 2022-11-11].
URL http://www.herongyang.com/Encoding/Base32-Encoding-Algorithm.html