Projekt do předmětu ISA:

Filtrující DNS resolver

Login: xdanca01
Jméno: Petr
Příjmení: Dančák
Rok: 2020/21

Obsah

1. Úvod do problematiky

Cílem projektu je implementace DNS resolveru, který bude poslouchat na zadaném portu, přeposílat dotazy na zadaný server a filtrovat požadavky, které budou pod/doménou nějaké z domén ve specifikovaném souboru. Toto řešení by mělo pomoci s filtrací nežádoucích doménových jmen a to například k blokování webů s reklamním sdělením, tedy nemusí se vždy jednat o škodlivý obsah.

1.1. DNS

Je to základní kámen pro přístup k internetu. Uživatel sám nemusí znát ip adresy, ale stačí mu zadat doménové jméno, které je serverem s doménovými jmény přeloženo na ip adresu, kterou vrátí tazateli a ten pak dale ví, s jakým serverem má komunikovat. Toto vše probíhá automaticky a v některých případech může znamenat hrozbu pro uživatele, proto by mohl každý uživatel znát co je to DNS filtering a jaké mohou nastat problémy s DNS obecně.

1.2. Problémy s DNS

Problémů s tímto mechanismem je mnoho a to jak s bezpečností, tak s funkčností. Servery s doménovými jmény nemusí fungovat, nebo mohou obsahovat zastaralá data, proto je nutné tato data aktualizovat. Dalším z problémů je bezpečnost přístupu ke stránkám. Existují totiž stránky, které obsahují škodlivý obsah, nebo je jednoduše chceme blokovat, protože obsahují obsah, který je nelegální nebo nežádoucí, proto existují dns filtery, které jsou často zabudované do mechanismu operačního systému a pracují se seznamy doménových jmen, podle kterých pak následně filtrují dotazy.

2. Návrh a implementace

Projekt se skládá z vícero jazyků, které můžeme rozdělit do 2 skupin. Hlavní část, tedy DNS resolver je naprogramován v jazyce c++ a tvoří ho jediný soubor main.cpp. Druhou část tvoří jednotlivé testy napsané ve skriptovacím jazyce bash, které se vyskytují v souboru test.sh.

2.1. Zpracování argumentů

Zpracování argumentů probíhá ve funkci main a k implementaci jsou použity knihovny ***stdlib.h*** a ***string.h*** Předané argumenty se postupně parsují konstrukcí if, else, která využívá funkcí ***strcmp()*** a ***strtol()***. Tyto funkce se používají pro porovnání očekávaných argumentů s předanými a pokud se některý argument neshoduje, tak dojde k vrácení chybového kódu, který je v případě parsing 3, a ohlášení chyby na standardní chybový výstup.

2.2. Server - start

Hlavní část serveru se také vyskytuje ve funkci main, ve které server začíná otevřením socketu funkcí ***socket()*** s parametry ***AF_*INET** (makro pro IPv4), ***SOCK_DGRAM*** (makro pro UDP), ***17*** (označení protokolu UDP) a jeho následovným nastavením na odposlech i odesílání (funkcí ***setsockopt()***) a nastavením portu, který je defaultně při nezadání argumentu port nastaven na hodnotu 53 a musíme ho zároveň převést z host byte order do network byte orderu funkcí ***htons()***. Dále socketu nastavíme, ať poslouchá na všech rozhraních hodnotou ***INADDR_ANY***, kterou musíme převést z lokálního byte orderu do síťového byte order funkcí ***htonl()***. Přiřazení portu, rozhraní na kterém má poslouchat a jestli se jedná o IPv4, nebo IPv6 se přiřadí funkcí ***bind()***. Po nastavení socketu si deklaruji určitý počet vláken, který je daný makrem ***THREADS***, který jsem si defaultně nastavil na 100. Jakmile server vstoupí do nekonečného loopu, tak čeká na příchozí komunikaci.

2.3. Server – navázání komunikace

Jakmile server obdrží nějaká data, tak si uloží adresu s portem odesílatele a zkontroluje typ požadavku. Server podporuje pouze typ požadavku ***A*** a na všechny ostatní odpovídá přímo odesílateli erorem typu ***NOTIMP***, který je popsaný v RFC a vrátí se na začátek loopu, kde opět čeká na navázání komunikace. Jestliže server dostane požadavek typu ***A***, tak zkontroluje obsah dat, jestli doména, kterou máme přeložit se nevyskytuje v souboru s filtrovanými doménami, a nebo jestli není poddoménou některé z domén v souboru. Pokud je dotaz filtrovaný, tak se opět odpovídá zpět odesílateli a to s erorem typu ***REFUSED*** a pokud dojde k internímu erroru, tak se odpovídá erorem typu ***SERVFAIL***. Jinak se vytvoří struktura pomocí funkce ***malloc()*** a ta se vyplní potřebnými parametry pro zpracování v jiné funkci. Jakmile se struktura naplní, tak se vytvoří nové vlákno, které začne funkcí ***send_next()*** a dostane vytvořenou strukturu.

2.4. Server – komunikace s DNS serverem

Pro komunikaci s DNS serverem se vždy vytvoří nový socket, který dostane náhodný port, aby jednotlivá vlákna mohla nezávisle komunikovat. Socketu se přiřadí komunikace oběma směry a zároveň hodnota timeoutu, po kterou čekáme na odpověď od DNS serveru. Data pošleme DNS serveru na výchozí port 53 a čekáme na odpověď po dobu timeoutu. Jestliže po tuto dobu nám nepřijde odpověď ze strany serveru, tak odpovíme zpět odesílateli s erorem typu ***SERVFAIL***, uvolníme alokovanou pamět a ukončíme práci vlákna. Jinak vrátíme odesílateli odpověď od DNS serveru.

3. Návod na použití

Pro správné přeložení programu je nutné mít Makefile a složku DNS ve stejném adresáři. Program se přeloží commandem ***make*** a smaže se commandem ***make clean***. Program očekává dva povinné argumenty ***'-s'***, za kterým následuje ip adresa serveru, na který se mají přeposílat dotazy a argument ***'-f'***, za kterým se specifikuje cesta k souboru s doménami, které se mají filtrovat. Třetí argument je volitelný a zadá se pomocí přepínače '-p' a za ním se specifikuje port, na kterém má program komunikovat. Záleží na pořadí argumentů, které je "-s -p -f".

3.1. Příklady spuštění

Překlad programu:

Make

- Přeloží zdrojový kód main.cpp a vytvoří program dns, kterému pak přiřadí práva ke spuštění.

Spuštění serveru:

dns -s <server> [-p <port>] -f <soubor_s_filtry>

- Po úspěšném překladu je možné spustit dns resolver tak jak je naznačeno o řádek výše. Argumenty mají striktně dané pořadí a argument -p je volitelný

Spuštění testů:

make test

- Tento příkaz nejdříve přeloží program a spustí test napsaný v programovacím jazyce bash, kde nejdříve spustí server s patřičnými argumenty a nad ním spouští testy definované v tom samém souboru. Nakonec vypíše, který test selhal nebo proběhl úspěšně a na konci vypíše počet failnutých testů.

Vymazání přeloženého programu:

make clean

- Smaže přeložený program

4. Omezení a připomínky

- DNS Server nepodporuje IPv6. Tedy komunikaci pomocí IPv6 ignoruje a to jak pro příjem, tak pro přeposílání dotazů (parametr -s). Důvodem je moje neschopnost testovat tuto funkčnost, jelikož nemám přístup do sítě pomocí IPv6 a i kdybych si vytvořil lokální síť, tak nemám kam poslat dotaz na rezoluci pomocí IPv6.
- Omezení na počet vláken je 100, ale tato skutečnost se dá změnit nastavením makra pro maximální počet vláken.
- Argument -s přijímá hodnotu doménového jména, ale jméno je přeloženo pomocí funkce gethostbyname, takže pokud funkce gethostbyname není povolená, pak program nepodporuje jako hodnotu argumentu -s doménové jméno.

5. Odevzdané soubory

- main.cpp – Zdrojový kód programu.
- manual.pdf – Tento dokument obsahující základní informace o projektu.
- Makefile – Soubor pro zapouzdření překladu, spouštění testů a smazání přeloženého programu.
- README.md – zkrácená forma dokumentace v jazyce Markdown.
- TEST/test.sh – Bash skript pro spouštění testovací fáze.
- TEST/filter_test.txt – Textový dokument pro testovací účely obsahující filtrované domény.

6. Použitá literatura

- DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION: RFC 1035. *Https://tools.ietf.org* [online]. [cit. 2020-10-12]. Dostupné z: https://tools.ietf.org/html/rfc1035
- Linux manual page: socket(2). *Man7.org* [online]. [cit. 2020-10-12]. Dostupné z: https://man7.org/linux/man-pages/man2/socket.2.html
- Extended DNS Errors. *Tools.ietf.org* [online]. [cit. 2020-10-12]. Dostupné z: https://tools.ietf.org/id/draft-ietf-dnsop-extended-error-05.html