Dokumentace k projektu z ISA

Tunelování datových přenosů přes DNS dotazy

Hung Do
xdohun00@stud.fit.vutbr.cz

29. října 2022

Obsah

1 Úvod

1.1 Domain Name System

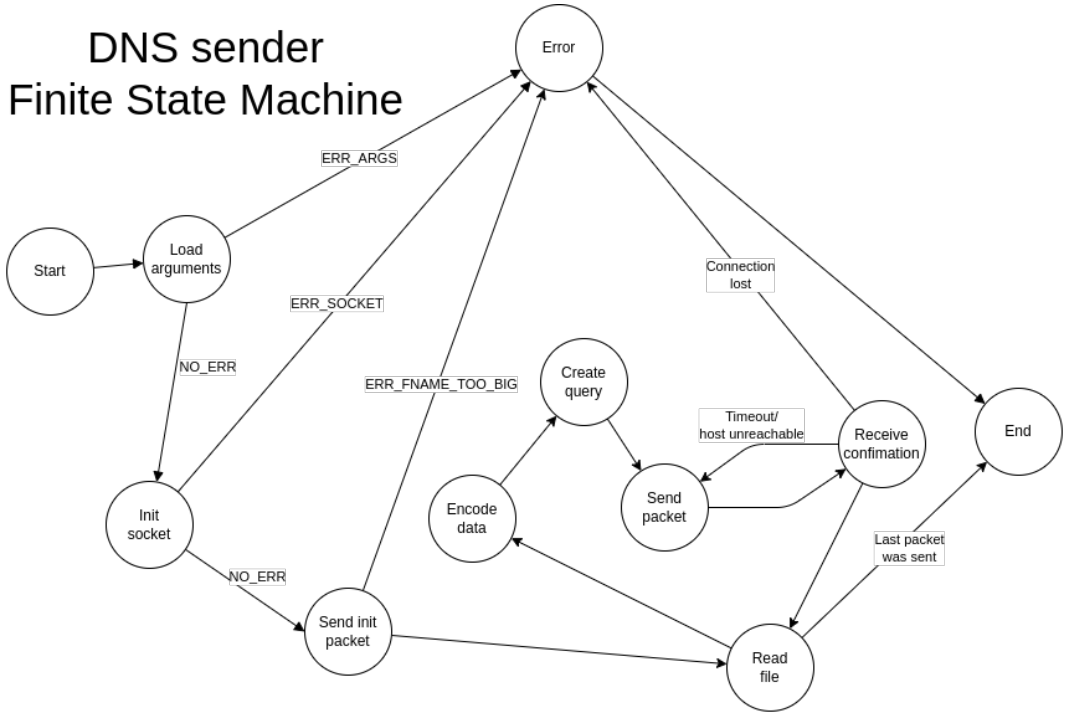
DNS neboli **Domain Name System** slouží pro převod doménových jmen do IP adres uzlů v síti. Jde o decentralizovaný, hierarchický systém doménových jmen, který je uspořádán do stromové struktury. Každý uzel stromu může mít pod sebou seznam podřízených domén. V kořeni tohoto stromu se nachází *tečka*. Pod ní se nachází *domény nejvýšší úrovně* (např. com, cz, sk, apod.). A tato hierarchie pokračuje dále.

Strom se dá rozdělit do jednotlivých zón, které spravují správci daných zón.

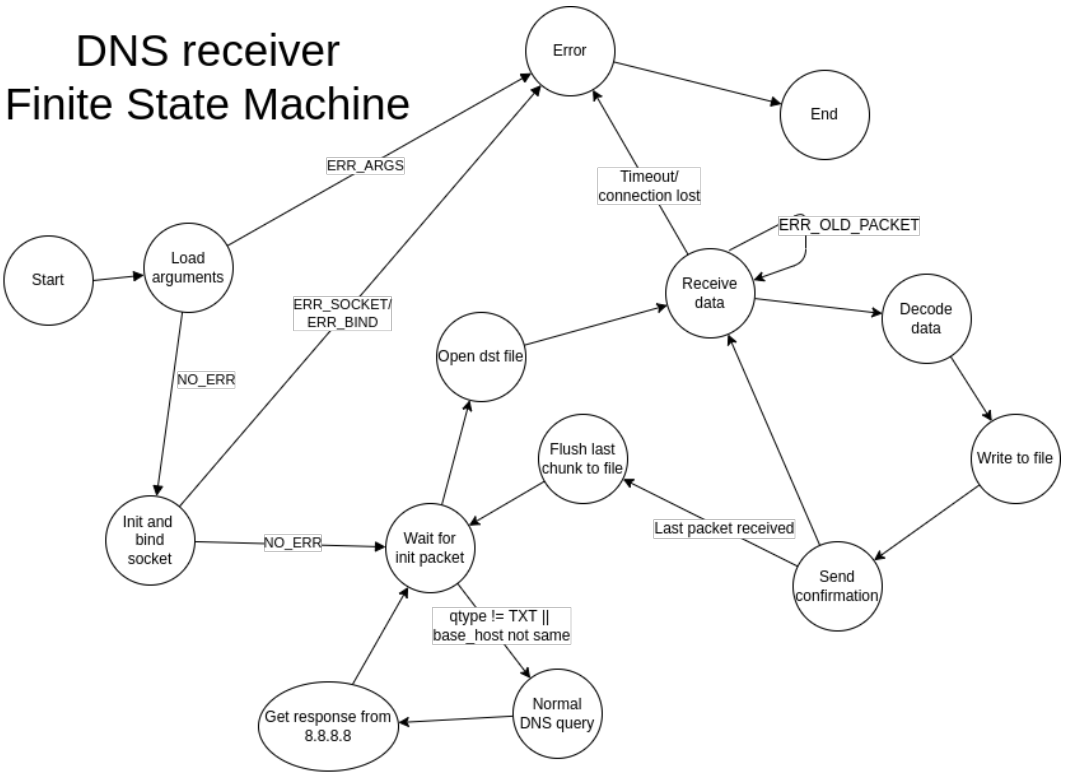
1.2 DNS tunel

DNS tunelování patří mezi známé kyberútoky. Využívá toho, že většina systémů mají povolený protokol DNS. Utočník si vytvoří cílovou doménu, která musí být viditelná externě. Následně pomocí např. trojského koně zakóduje data souborů či jiných programů do DNS dotazů a ten se odešle na útočníkův autorativní DNS server. Tam se data složí do své původní podoby.[2]

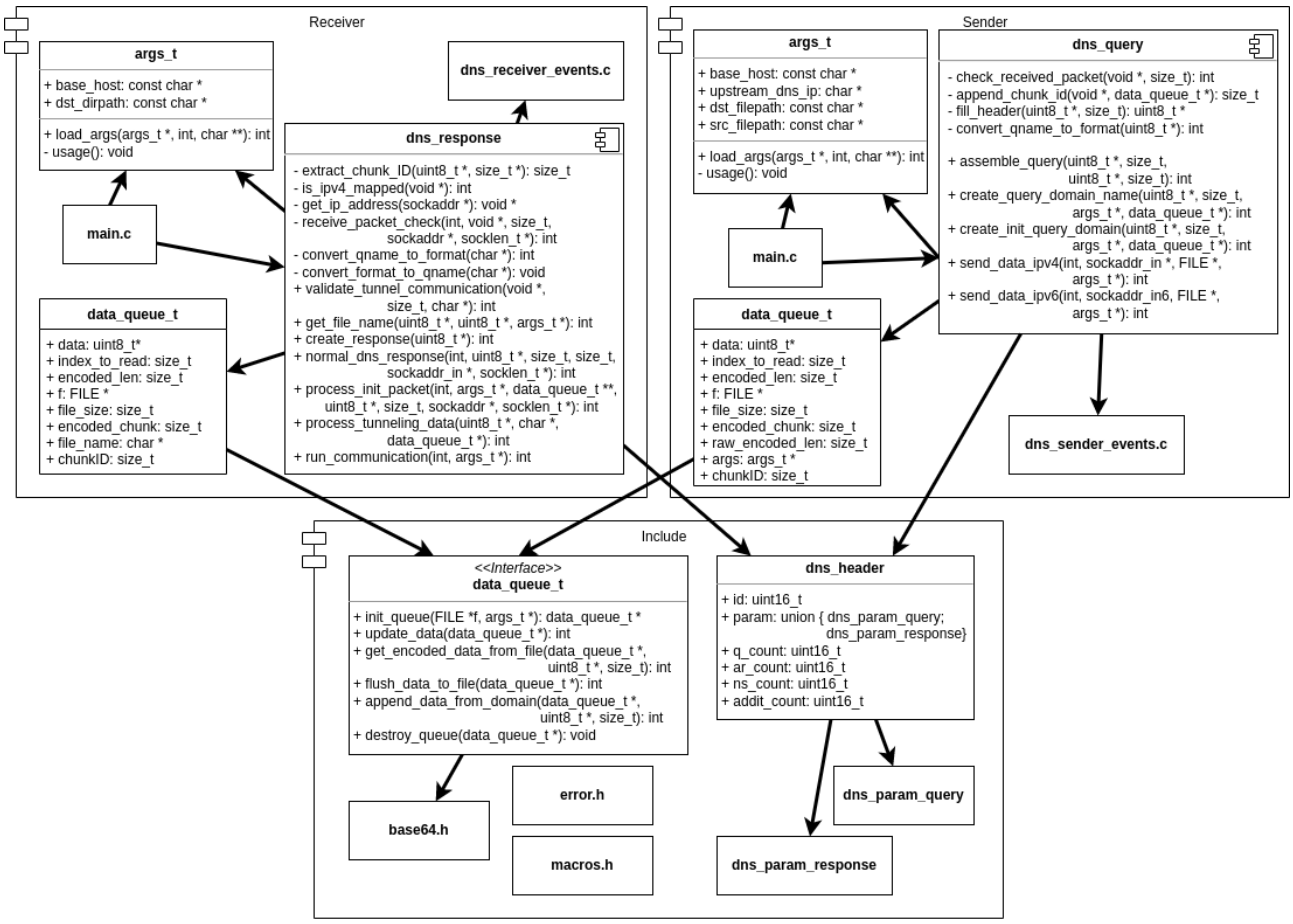
2 Návrh implementace



Obrázek 1: Konečný automat programu **dns_sender**



Obrázek 2: Konečný automat programu **dns_receiver**



Obrázek 3: Propojení jednotlivých modulů v projektu

3 Implementace

Implementace projektu je realizována v programovacím jazyce C za použití systémových knihoven `<arpa/inet.h>`, `<netinet/in.h>` a `<sys/socket.h>`. Pro algoritmus kódování base64 byly použité zdrojové soubory od **The Apache Group**[1].

3.1 DNS Sender

Implementace `dns_sender` se nachází ve složce `sender/`. Program je rozdělen do čtyř částí:

- `main` - startující bod programu; vytváří a nastavuje soket
- `dns_query` - načítá, kóduje, transformuje a odesílá data na server
- `arguments` - zpracovává argumenty programu
- `data_queue` - definice struktury pro uchování zakódovaných dat a dalších pomocných informací

V souboru `main.c` se vytváří a nastavují sokety pro komunikaci. Po nastavení soketů se zavolá funkce `send_data` z `dns_query.h`, která již zpracovává, transformuje a odesílá data na server. Jádrem programu je soubor `dns_query.c`, který obsahuje funkce potřebné pro tvorbu DNS dotazů.

Komunikace začíná inicializačním paketem, který definuje jméno cílového souboru *DST_FILEPATH* a první *chunkID* komunikace. Po úspěšném navázání kontaktu ve odesílají DNS dotazy s daty. Paketu je nejprve nastavena výchozí DNS hlavička, poté se vygeneruje doménové jméno. Doménové jméno je ve formátu:

```
[chunkID].[collection_of_labels].[base_host]
```

Jednotlivé *labels* jsou naplňovány zakódovanými daty získané ze souboru *SRC_FILEPATH* (viz. sekce 5). Po odeslání a přijmutí posledního paketu vypíše program ukončující log a ukončí úspěšně program.

3.2 DNS Receiver

Zdrojové soubory k `dns_receiver` jsou k nalezení ve složce `receiver/`. Obdobně jako `dns_sender` je rozdělen do čtyř částí:

- `main` - startující bod programu; vytváří a nastavuje soket
- `dns_response` - přijímá dotazy, dešifruje obsah paketů a skládá z dat cílový soubor
- `arguments` - zpracovává argumenty programu
- `data_queue` - definice struktury pro uchování zakódovaných dat a dalších pomocných informací

Příjemce dat je čeká na inicializační paket, ze kterého dešífruje jméno cílového souboru. Do něj se ukládají dešifrovaná data příjimaných dotazů. Cyklus se načítá do doby, než server obdrží dotaz na doménu, která je kratší než nejdelší povolené jméno[2]. Po dokončení program čeká na další spojení není-li definováno jinak (viz. sekce 4).

3.3 Data Queue struktura

Tato struktura je sdílená v obou programech a její obsah je definována každým programem jinak. Jedná se o pomocnou strukturu, která obsahuje pomocná data (např. jméno cílového souboru či hodnota *chunkID*) a do které si program ukládá zakódovaná data. Jednotlivé struktury jsou definované v souborech `sender/send_queue.h` resp. `receiver/recv_queue.h`.

[1] Zdrojové soubory jsou k nalezení **zde**.

[2] Formát je definován zde: `https://datatracker.ietf.org/doc/html/rfc1035#section-2.3.4`

`dns_sender` ze struktury odebírá data a vkládá je do jednotlivých *labels*. Pokud již z fronty odebral všechny data, struktura automaticky načte nová data ze zdrojového souboru a zakódovaná data vloží opět do fronty.

`dns_receiver` naopak strukturu naplňuje přijímanými daty z DNS dotazů. Pokud se fronta naplní, struktura dešifruje obsah fronty a data „spláchne“ do cílového souboru.

3.4 Chybové kódy

Chybové kódy jsou definované v souboru `include/error.h`.

- `NO_ERR` (0) - nenastala žádná chyba
- `ERR_ARGS` (1) - uživatel nezadal všechny povinné argumenty programu
- `ERR_SOCKET` (2) - nepovedlo se vytvořit soket
- `ERR_IP_FORMAT` (3) - špatný format IP adresy (IPv4 i IPv6)
- `ERR_CONNECT` (4) - nepovedlo se vytvořit spojení mezi klientem a serverem
- `ERR_NO_FILE` (5) - zdrojový souboru nebyl nalezen
- `ERR_BIND` (6) - nepodařilo se připojit soket k portu
- `ERR_FNAME_TOO_BIG` (254) - jméno cílového souboru je moc dlouhý a není možno vytvořit inicializační paket
- `ERR_OTHER` (253) - předem nepředvídatelná chyba (např. nepovedená alokace paměti)

3.5 Zajímavé pasáže kódu

Pro zefektivnění přenosu dat, `dns_sender` se snaží odeslat co nejvíce dat skrze jeden DNS dotaz. Pomocí jednoduché funkce `MIN(a,b)`, která vrací nižší hodnotu ze 2, dokáže zjistit, kolik bajtů může do dotazu ještě vložit, aby nepřesáhla maximální povolenou velikost *label* a přitom neobsadilo místo vyhrazenou pro `BASE_HOST`.

```
/* dns_query.c */
int create_query_domain_name(uint8_t *buffer, size_t buffer_size,
        struct args_t *args, struct data_queue_t *q) {
    // ...

    // reset buffer
    memset(buffer, 0, buffer_size);
    static uint8_t label[LABEL_SIZE] = { 0, };
    // secondary buffer for pointer arithmetics
    uint8_t *buffer_ptr = buffer + 1;

    // append chunk ID
    size_t chunk_size = append_chunk_id(buffer_ptr, q);
    buffer_ptr += chunk_size;

    size_t len, total = 0;
    size_t available_size = buffer_size - strlen(args->base_host)
                                - 2 - buffer[0] - chunk_size;

    // cycle to fill labels
    while ((len = get_encoded_data_from_file(q, label,
                    MIN(LABEL_SIZE,available_size - total))) == LABEL_SIZE) {
                //  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
        // move data from label into destination buffer
        memcpy(buffer_ptr + total, label, len);
```

```
        total += len;
        q->raw_encoded_len += len;
        q->encoded_chunk += len;
        // append separator
        buffer_ptr[total++] = '.';
    }
    // append last label
    memcpy(buffer_ptr + total, label, len);
    total += len;
    q->raw_encoded_len += len;
    q->encoded_chunk += len;

    // add separator and append domain
    buffer_ptr[total] = '.';
    memcpy(buffer + total + 2 + chunk_size,
           args->base_host, strlen(args->base_host));

    // ...
}
```

4 Základní informace o programu

Program `dns_receiver` podporuje komunikaci jak pomocí IPv4, tak i IPv6 protokolů. Ke komunikaci bylo použitý protokol UDP. Ztráta paketu je ošetřena pomocí nastavení soketu odesílatele. Ten paket může poslat maximálně 5x po 3 sekundových intervalech. V případě, že i pátý paket nebyl úspěšně doručen, je program terminován.

Poté, co příjemce dat vytvoří spojení s odesílatelem, je mu nastavený *timeout* pro čekání na odeslaná data. Pokud příjemce nedostane do 15 sekund žádný paket a neobdržel poslední ukončující paket, prohlásí spojení za ztracené a ukončí program. `dns_receiver` umí též odpovědět na klasické DNS dotazy. Pokud přijatý dotaz nesplňuje dohodnutý format tunelovacího paketu (viz. sekce 3), dotaz je automaticky přesměrován na veřejný Google DNS server na adrese **8.8.8.8**. Odpověď z Google serveru je poté zase přesměrovaná zpátky odesílateli.

V souboru `include/macros.h` se nachází makro `CONTINUOUS_RUNNING`, skrze který je možno definovat mód programu `dns_receiver`. Pokud je hodnota makra nastavena na 0, program se ukončí, jakmile úspěšně dokoná přenos programu/vyřízení DNS dotazu. V opačném případě program běží, dokud nenastane chyba (ztráta spojení) či je násilně ukončen (např. SIGSEGV). Po změně hodnoty je potřeba znovu přeložit program (`make clean && make`).

5 Návod na použití

Pro vytvoření komunikace mezi oběma programy je zapotřebí, aby strana simulující server měla **sudo oprávnění** a volný **DNS port (port 53)**. Pokud je port již zabraný, je možný ho uvolnit pomocí programu `fuser`[1]. Druhá možnost je přerušení procesu vyžadující daný port.

```
[root@fedora ~]# ss -naup | grep ":53 "
UNCONN 0      0                        127.0.0.53%lo:53         0.0.0.0:*
    users:(("systemd-resolve",pid=945,fd=17))
[root@fedora ~]# systemctl stop systemd-resolved.service
[root@fedora ~]# ss -naup | grep ":53 "
[root@fedora ~]#
```

[1] viz. manuálové stránky: `https://man7.org/linux/man-pages/man1/fuser.1.html`

Poté, co máme zajištěný port 53, je možné komunikaci rozjet.
Na serveru rozjedeme program `dns_receiver`. Parametry programu `dns_receiver` jsou:

- `BASE_HOST` - definuje útočníkovu (cílovou) doménu, na kterou se odesílají data
- `DST_DIRPATH` - definuje cílovou složku, do které se mají vytvářet soubory s přijímanými daty. Pokud složka neexistuje, je programem vytvořena.

Na straně klienta rozjedeme program `dns_sender`. Parametry programu `dns_sender` jsou:

- `-u UPSTREAM_IP` - definuje, na jakou IP adresu má program přesměrovat DNS dotazy. Pokud toto nastavení není definované, program načte adresu ze souboru `/etv/resolve.conf`.
- `BASE_HOST` - definuje cílovou doménu (musí být stejná jako zadaná ve `dns_receiver`).
- `DST_FILEPATH` - definuje jméno cílového souboru (jak se bude jmenovat soubor uložený na útočníkově straně).
- `SRC_FILEPATH` (volitelný) - definuje cestu ke zdrojovému souboru. Pokud zdrojový soubor nebyl zadán, program naslouchá ze `STDIN`.

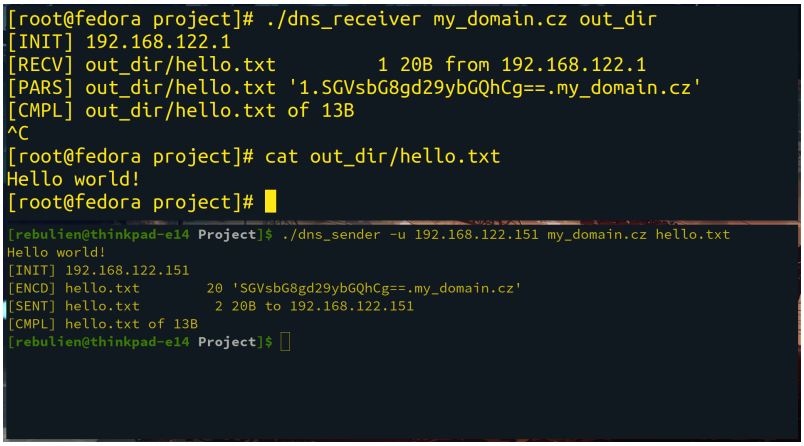
K překladu projektu slouží přiložený souboru `./Makefile` (více informací naleznete v `README.md`). Příklady rozjetí komunikace naleznete v sekci 6.

6 Testování

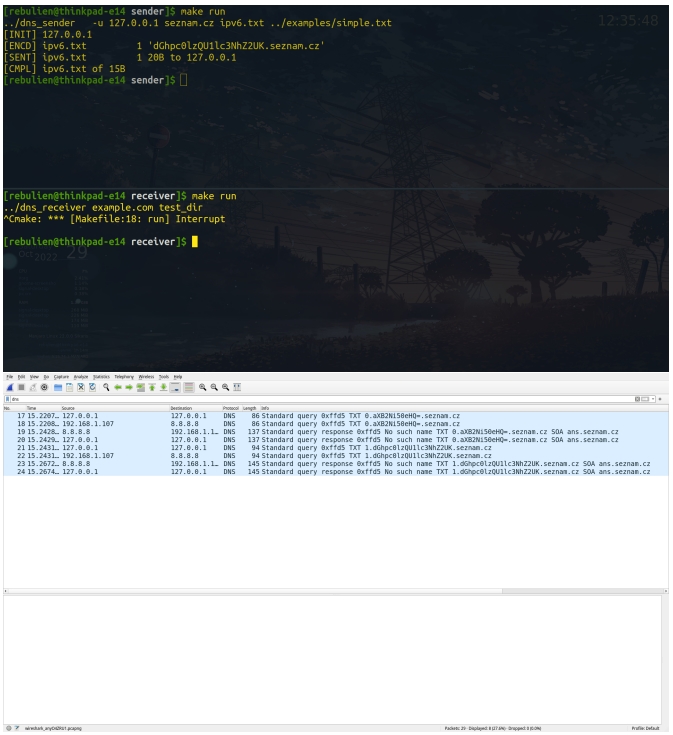
Na testování byly použité 2 systémy:

- `Manjaro Linux x86_64`, s jádrem `5.15.32-3-MANJARO`.
- `Fedora Server x86_64`, s jádrem `5.17.12-100.fc34.x86_64`

Z prvního systému se zapínal program `dns_sender` a odesílaly se z něj data. Druhý systém zprávu přijímal a data skládal do souboru pomocí programu `dns_receiver`.



Obrázek 4: Odesílání dat ze *STDIN*



Obrázek 5: Odesílání normálního DNS dotazu

Obrázek 6: Odesílání a spuštění binárního souboru (FLAC soubor)

Literatura

[1] Fenner, B.; Rudoff, A. M.: *UNIX Network Programming: The sockets networking API*. Addison-Wesley, třetí vydání, 2004, ISBN 0-13-141155-1.

[2] Infoblox: What is DNS tunneling? [online], 2022 [cit. 2022-10-28]. Dostupné z: `https://www.infoblox.com/glossary/dns-tunneling/`

[3] Kurose, J. F.; Ross, K. W.: *Computer networking: A Top-Down Approach*. Pearson, 2017, ISBN 978-0-13-359414-0.