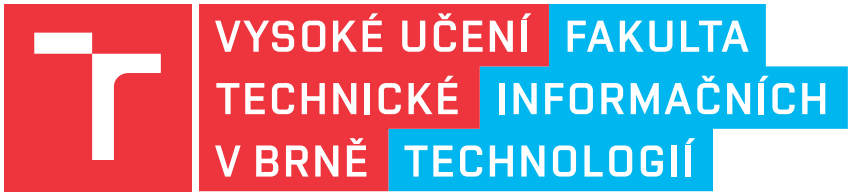

Dokumentace k projektu
Přenos souboru skrz skrytý kanál
Síťové aplikace a správa sítí

26. října 2021

Michal Pyšík (xpysik00)

Obsah

1 Úvod

Cílem projektu bylo implementovat aplikaci typu klient-server, která umožní skrytý přenos souborů za použití pouze ICMP Echo-request/reply zpráv. Klient daný soubor zašifruje a pošle serveru, který ho dešifruje a uloží do složky, ze které byl spuštěn. Bylo třeba definovat tajný protokol, který slouží primárně k vzájemné identifikaci klienta/serveru, jejich komunikaci a ověření, že daný soubor dorazil v pořádku.

1.1 Komunikace přes ICMP protokol

ICMP (*Internet Control Message Protocol*) je protokol používaný především k posílání různých chybových hlášení, nebo testům dostupnosti zařízení v síti, obvykle přiřazován k protokolům síťové vrstvy L3. Mezi zařízeními komuninikující tímto protokolem není zřízeno ustálené spojení, v kontrastu s protokoly jako je například TCP.

Při využívání datové části paketu pro přenost dat lze ICMP komunikaci využít jako skrytý kanál, tato technika se označuje názvem ICMP tunneling [6]. Bez znalosti domluveného způsobu komunikace, případně i způsobu enkrypce dat, je smysluplné monitorování této komunikace třetí osobu téměř nemožné.

2 Návrh aplikace

Tato sekce se zaměřuje na klíčové části programu spíše z abstraktního pohledu, bez konkrétních implementačních detailů popsaných v kapitole 3.

2.1 Zpracování argumentů

Program začíná zpracováním argumentů příkazové řádky. Pokud chce uživatel program spustit jako klient, musí zadat cestu k odesílanému souboru a IP adresu cílového serveru, místo které má také možnost zadat doménové jméno, které bude na adresu automaticky přeloženo. Dále je zde přepínač pro spuštění programu v roli serveru, a přepínač který vypíše nápovědu včetně základních informací o programu a správného použití.

2.2 Skrytý protokol

Před popisem fungování klienta a serveru je třeba popsat skrytý protokol, kterým komunikují. Ten je složen ze 3 částí, které společně obsazují vždy prvních 8 bytů datové části ICMP paketu. Předpokládá se, že žádný paket nebude ztracen, protokol se však umí vypořádat s datovými pakety, které dorazí poškozené.

První položkou je podpis odesílatele, kde `0xaaaa` je podpis klienta, `0xbbbb` je standardní podpis serveru, a `0xcccc` je podpis serveru v případě, že oznamuje klientovi obdržení poškozeného paketu.

Druhou položkou je unikátní identifikátor klienta, který mu je přiřazen serverem. Celkový počet identifikátorů udává maximální počet klientů aktivních v daném časovém okamžiku, tedy identifikátor neidentifikuje klienta ve smyslu jeho zařízení, ale pouze konkrétní aktivní přenosu souboru.

Třetí položkou je sekvenční číslo, které označuje pořadí paketu v rámci paketů obsahující část přenášeného souboru. První souborový paket má pořadí 1, jelikož pořadí 0 je rezervováno pro označení nové žádosti o zaslání souboru.

Struktura skrytého protokolu je výrazně ovlivněna faktem, že jsem původně místo datové části využíval identifikátor v IP hlavičce [4], společně s identifikátorem a sekvenčním číslem v ICMP hlavičce, to jsem se však později rozhodl pozměnit.

2.3 Klient

Funkce klienta začíná vytvořením soketu pro komunikaci se serverem, klient posílá vždy pouze ICMP zprávy typu `Echo Request` (kód 8). Poté je třeba zjistit celkovou velikost odesílaného souboru (v bytech), z čehož lze také vypočítat celkový počet potřebných paketů.

První zprávou kterou klient serveru zašle je požadavek o zaslání souboru, který se identifikuje sekvenčním i identifikačním číslem nabývající hodnoty 0. Tato zpráva obsahuje velikost odesílaného souboru, následovanou jeho názvem. Pokud server odpoví stejnou zprávou pouze s pozměněným podpisem a identifikátorem klienta (přidělený identifikátor, který je třeba si uložit a dále se ním identifikovat), zahájí klient přenos souboru.

Soubor je čten po částech, omezenými maximální velikostí paketu na síti (1500 B). Každá část je poté zašifrována a zaslána serveru, včetně daného pořadového čísla paketu. Před odesláním dalšího paketu je třeba vyčkat na odpověď serveru, a pokud server odpoví zprávou s podpisem označující přijetí poškozeného paketu, je nutné vykonat znovuzaslání tohoto paketu. Po zaslání posledního paketu obsahující data souboru a přijetí potvrzení o jeho korektním doručení od serveru se klient ukončí.

2.4 Server

Také server musí nejprve vytvořit vhodný soket, pomocí kterého bude klientovi zasílat ICMP odpovědi typu `Echo Reply` (kód 0). Dále je vytvořeno pole struktur, viz 3.5, kam se vždy dočasně ukládají informace o aktivních přenosech od klientů. Poté server začne zachytávat zprávy od klientů, a vhodně na ně reagovat.

Obdrží-li server nový požadavek, je třeba zkontrolovat, zda je dostupný neobsazený identifikátor, který lze zaslat klientovi v rámci potvrzení zahájení přenosu. Pokud ano, server otevře soubor s odpovídajícím jménem pro zápis v adresáři, odkud je spuštěn. Pokud jsou všechny identifikátory v daný moment obsazené, oznámí server klientovi, že je aktuálně zaneprázdněn.

Když server obdrží paket s nenulovým pořadovým číslem, ověří podle identifikátoru, zda se jedná o validního aktivního klienta odesílající datový paket s očekávaným pořadím. Na základě ověření správnosti ICMP checksum se poté rozhodne, zda paket dorazil v pořádku, tedy danou část lze dešifrovat a zapsat do souboru. Klientovi je zaslána informace o tom, zda má zaslat následující paket, nebo opakovat odeslání aktuálního paketu. Pokud server obdržel poslední očekávaný paket od daného klienta a paket dorazil v pořádku, dokončí zápis souboru, odešle potvrzující zprávu a uvolní daný identifikátor novým příchozím požadavkům od klientů.

Server neustále zpracovává příchozí požadavky až do ukončení uživatelem, jehož správné ošetření je popsáno v sekci 3.6.

3 Popis implementace

Tato sekce je zaměřena na zajímavé implementační detaily včetně použitých technologií a omezení aplikace. Zdrojový kód celého programu napsaný v jazyce C++ se nachází v souboru `secret.cpp`, společně s odpovídajícím hlavičkovým souborem `secret.hpp`.

3.1 Zpracování argumentů

Zpracování argumentů příkazové řádky probíhá pomocí knihovny `getopt` [2]. Při potřebě předkladu doménového jména na IP adresu je volána funkce `hostname_to_ip`, která se kromě samotného překladu stará o rozpoznání různých druhů chyb které mohou při překladu nastat, následného informování uživatele skrze standardní chybový výstup, a ukončení programu s chybovou návratovou hodnotou.

3.2 Strana klienta/serveru

Funkce klienta i serveru probíhá uvnitř jejich vlastních funkcí `client_side` a `server_side`, které jsou volány přímo z funkce `main`. V obou případech je nejprve vytvořen BSD soket [3] typu `SOCK_RAW` operující nad protokolem `IPROTO_ICMP`. Příchozí pakety jsou ukládány do statického bufferu, na jehož odpovídající části ukazují ukazetele struktur ICMP hlavičky a tajného protokolu, pomocí čehož jsou příchozí zprávy filtrovány a zpracovány. Způsob komunikace klienta se serverem se je popsán v sekcích 2.2, 2.3 a 2.4.

3.3 Odesílání ICMP paketů

K odesílání paketů přes vybraný soket slouží funkce `send_ICMP_packet`. Té jsou kromě odesílajícího soketu, samotných dat a adresy příjemce předávány jako parametry všechny atributy skrytého protokolu a typ ICMP zprávy. Do paketu je také vložena ICMP checksum vypočítaná funkcí `ICMP_checksum`, což je klíčové pro kontrolu správnosti příchozích paketů ze strany serveru. Je třeba dávat pozor na to, že velikost bufferu pro odesílání nezahrnuje velikost IP hlavičky, která je generována automaticky, to však neplatí pro příchozí pakety.

3.4 Šifrování souborů

Pro enkrypci přenášených souborů byla požita šifra AES (*Advanced Encryption Standard*), implementována knihovnou `OpenSSL` [5]. Klient daný soubor zašifruje funkcí `encrypt_data`, server ho poté dešifruje symetricky obrácenou funkcí `decrypt_data`. Tyto funkce operují nad datovými bloky o velikosti 16 bytů, tudíž klient musí případně při šifrování konce souboru použít daný počet libovolných bytů na tzv. ”padding”, server si poté z celkové velikosti souboru vypočítá, kolik takových bytů bylo použito a po dešifrování je zahodí.

3.5 Obsluha více klientů současně

Server uchovává informace o aktivních klientech (přenosech) ve statickém poli struktur typu `Client_request`. Její složky jsou název souboru, ukazatel na otevřený soubor, očekávaná velikost souboru, očekávaný počet paketů, očekávané sekvenční číslo následujícího paketu, počet přijatých paketů a informace o tom, zda je daný slot obsazen. Index konkrétní struktury v poli odpovídá přidělenému identifikátoru daného klienta (přenosu), přičemž maximální počet souběžných přenosů je dán konstantou `MAX_CLIENT_ID`, jejíž výchozí hodnota je 32.

3.6 Ukončení běžícího programu

Pomocí knihovny `signal.h` [1] je ošetřeno ukončení programu signálem `SIGINT` (CTRL+C). Pro server je to nutností, jelikož ze své podstaty má běžět v neustálé smyčce, ale je také vhodné mít možnost ukončit klienta čekajícího na odpověď od nedostupného serveru. Tyto obsluhy jsou implenetovány funkcemi `handle_client_sigint` a `handle_server_sigint`, které se kromě výpisu zprávy uživateli (pouze při zadaném parametru `-v`) a návratové hodnoty starají také o uvolnění alokované paměti.

3.7 Omezení aplikace

Program pracuje pouze s protokolem IPv4, podpora IPv6 není implementována. Dále se pro správnou funkci programu předpokládá, že každý zaslaný paket dorazí k cílovému zařízení (nedojde ke ztrátě paketů).

4 Použití programu

Pozor, pro správné fungování klienta i serveru je třeba program spustit s oprávněními superuživatele `root`. Program při absenci verbose režimu na standardní výstup nic nevypisuje (pokud uživatel nezvolil výpis nápovědy), případný výpis chybových hlášení na standardní chybový výstup však tímto režimem podmíněn není.

4.1 Podporované argumenty

- `-r` *<soubor>* Relativní nebo absolutní cesta k odesílanému souboru.
- `-s` *<ip|hostname>* IPv4 adresa nebo doménové jméno cílového serveru.
- `-l` Spuštění programu v režimu odposlechu (v roli serveru).

- `-v` Povolení výpisu informací o stavu programu a přenosu souboru na standardní výstup (verbose).
- `-h` Výpis nápovědy, včetně popisu programu a jeho správného použití (help).

4.2 Spuštění klienta

- `secret -r <soubor> -s <ip|hostname> [-v]`

Příklady:

- `secret -r poznamky.txt -s 127.0.0.1`
- `secret -r ../obrazky/delfin.png -s merlin.fit.vutbr.cz -v`

4.3 Spuštění serveru

- `secret -l [-v]`

4.4 Výpis nápovědy

- `secret -h`

Reference

[1] CGF: Pubs.opengroup.org: *signal.h - signals*. [online], 2000, [vid. 2021-10-13].
Dostupné z: https://pubs.opengroup.org/onlinepubs/009695399/basedefs/signal.h.html

[2] Hall, J.: Short option parsing using getopt in C. [online], 19. srpen 2021, [vid. 2021-10-15].
Dostupné z: https://pubs.opengroup.org/onlinepubs/009695399/basedefs/signal.h.html

[3] Moon, S.: How to Code Raw Sockets in C on Linux. [online], 26. červenec 2020, [vid. 2021-10-16].
Dostupné z: https://www.binarytides.com/raw-sockets-c-code-linux/

[4] Reusch, C.: A short story about the IP ID Field. [online], 29. srpen 2015, [vid. 2021-10-13].
Dostupné z: https://crnetpackets.com/2015/08/29/a-short-story-about-the-ip-id-field/

[5] Rijmen V., B. P., Osselaers A.: OpenBSD manual page server: *AES_encrypt(3)*. [online], 28. srpen 2019, [vid. 2021-10-16].
Dostupné z: https://man.openbsd.org/AES_encrypt.3

[6] Stødle, D.: Ping Tunnel: *For those times when everything else is blocked.* [online], rev. 5. září 2011, [vid. 2021-10-13].
Dostupné z: http://www.cs.uit.no/~daniels/PingTunnel/