Vysoké učení technické v Brně
Fakulta informačních technologií

Síťové aplikace a správa sítí – projekt
varianta 2

Přenos souboru skrz skrytý kanál

manuál a dokumentace

15. listopadu 2021

Ondřej Ondryáš

Obsah

1 Úvod

Tato dokumentace popisuje použití a implementaci aplikace, která umožňuje přenos zašifrovaných dat pomocí ICMP nebo ICMPv6 datagramů. Aplikace funguje buď v režimu odesílatele („klienta“), nebo v režimu příjemce („serveru“). V režimu odesílatele čte data ze zadaného souboru, šifruje je symetrickou šifrou AES-256-CBC a po částech je vkládá do ICMP(v6) Echo (Request) datagramů, které odesílá na zadanou adresu. Stejná aplikace spuštěná v režimu příjemce na druhé straně zachytává přijaté ICMP(v6) datagramy, dešifruje obsah, který nesou, a ukládá jej do souboru. Aplikace implementuje mechanismy pro zajištění spolehlivosti přenosu.

2 Použitá terminologie

V protokolu IP verze 4 se využívá pomocný protokol ICMP (Internet Control Message Protocol) [1], v protokolu IP verze 6 mluvíme o ICMPv6 (Internet Control Message Protocol for the IP Version 6) [2]. V kontextu zde dokumentované aplikace se tyto verze ICMP liší pouze v drobných aspektech. Pokud není řečeno jinak, principy popisované v této dokumentaci související s ICMP a IPv4 jsou použity obdobně pro ICMPv6 a IPv6.

Pokud je v tomto dokumentu použito spojení **zpráva aplikačního protokolu** nebo **zpráva AP**, je tím myšlen IP datagram s ICMP *Echo* hlavičkou, jejíž pole *Identification*, *Sequence Number* a data jsou nastaveny na hodnoty, které aplikace interpretuje definovaným způsobem.

3 Použití ICMP pro přenos dat

Internet Control Message Protocol (ICMP) je nedílnou součástí protokolu IP [1]. Hlavičky protokolu ICMP jsou obalovány hlavičkami IP a mají přiřazené číslo protokolu[1] [3]. To naznačuje, že by mohlo jít o protokol vyšší vrstvy, nepovažuje se za něj však, protože není zamýšlen pro poskytování transportního média aplikacím na vyšších vrstvách. ICMP se používá pro přenos doplňujících informací mezi stranami komunikujícími pomocí IP. Takovou informací může být např. zpráva o tom, že datagram nemůže být doručen příjemci. Jednotlivé typy zpráv jsou definovány v [2] a [1].

Jedním z nich je typ *Echo*[2]. ICMP zpráva tohoto typu obsahuje mj. dva číslené identifikátory (*Identifier* a *Sequence Number*) a kromě povinných polí hlavičky ICMP může obsahovat libovolné množství jakýchkoliv dat. Tyto zprávy jsou typicky určeny pro diagnostické účely a chování zařízení při jejich přijetí je prosté: zašlou zpět ICMP zprávu typu *Echo Reply* s nezměněnými daty a identifikátory. [1, 2]

Právě ICMP *Echo* zprávy je tedy možné využít jako nosiče pro přenos vlastních dat. V principu stačí pouze rozdělit odesílanou zprávu na vhodně malé části, obalit je ICMP *Echo* hlavičkou a odeslat je. Příjemce poté zachytává všechny přijaté ICMP *Echo* zprávy a ukládá jejich obsah. Protože odeslané ICMP zprávy nemusí dojít v odesílaném pořadí, je nutné do zprávy zahrnout informaci o tom, jakou část původních dat daný datagram reprezentuje. Také je vhodné všechny zprávy označovat sjednaným identifikátorem, aby bylo možné odlišit, které ICMP zprávy jsou součástí přenosu dat. Je vhodné také implementovat mechanismus pro zjištění úspěchu či neúspěchu přenosu a případné vyžádání chybějících dat.

Z popsaných požadavků vyplývá, že pro spolehlivý přenos dat je nutné nad protokolem ICMP v podstatě „ustavit spojení“ na aplikační vrstvě. Konkrétní způsob, jakým je toto provedeno v mé implementaci, je popsán v kapitolách 6.4 a 6.6.

[1] 8 pro ICMP, 58 pro ICMPv6.

[2] V ICMPv6 označován jako *Echo Request*.

4 Překlad a spuštění aplikace

Aplikaci je možné přeložit s použitím GNU Make příkazem `make`, příp. `make secret`. Pro překlad je vyžadována knihovna **OpenSSL ve verzi 1.1.1**. Doporučeným překladačem je GCC ve verzi alespoň 9.1.0, s jinými překladači nebyl překlad testován[3]. Aplikace je implementována v **C++17**, z novějších API jazyka C++ využívá například `std::filesystem`.

Před překladem je možné změnit některé aspekty chování aplikace úpravou konstant v souboru `common.h`. Další informace ke konfigurovatelným konstantám jsou uvedeny dále v textu.

Aplikace využívá schránky typu `RAW`, proto je nutné ji spouštět s oprávněním superuživatele[4].

4.1 Režim příjemce (serveru)

Aplikaci v režimu příjemce je možné spustit příkazem:

```
./secret -l
```

Aplikace začne okamžitě zachytávat příchozí ICMP komunikaci. Pokud identifikuje zprávu příslušnou aplikačnímu protokolu, ustaví logické spojení (na aplikační vrstvě) a zahájí přenos.

Název přenášeného souboru určuje odesílatel. Ve výchozím stavu příjemce nezačne přijímat soubor, pokud soubor požadovaného jména v pracovním adresáři už existuje. Toto chování je možné změnit parametrem **`-o`**, který povolí přepis existujícího souboru.

4.2 Režim odesílatele (klienta)

Aplikaci v režimu odesílatele je možné spustit příkazem:

```
./secret -r jméno souboru -s IP nebo hostname
```

Oba parametry `-r` a `-s` jsou v režimu odesílatele povinné. Režimy odesílatele a příjemce se vzájemně vylučují, není možné použít `-r` zároveň s `-l`.

Pokud je zadáno doménové jméno, aplikace jej přeloží na IP adresu[5]. Poté se pokusí ustavit spojení (na aplikační vrstvě) a odesílá data.

4.3 Možnosti společné pro oba režimy

V obou režimech je možné použít následující parametry:

`-k` umožní před zahájením komunikace nastavit klíč pro šifrování dat. Požadovaný klíč zadává uživatel interaktivně do terminálu, zadávané znaky se nezobrazují.

`-v` na standardní výstup vypisuje podrobnější informace o průběhu komunikace.

`-q` zamezí jakémukoliv výstupu na standardní výstup.

[3] Překladač i knihovna v požadované verzi jsou dostupné v poskytnutém referenčním virtuálním stroji. Na serveru `merlin` není dostupná knihovna OpenSSL v požadované verzi, GCC je přítomno, ale ve výchozím stavu v nižší verzi. Pro překlad v tomto prostředí by byly nutné dodatečné kroky.

[4] Proces musí mít efektivní UID rovné 0 nebo nastaven atribut `CAP_NET_RAW`, viz [4].

[5] Detaily rezoluce viz `gethostbyname`: [5].

5 Činnost aplikace, princip komunikace

Zahájení komunikace

Komunikaci zahajuje odesílatel. Po případném přeložení doménového jména na IP adresu je zjištěno, na jakém rozhraní je tato adresa dostupná (více v kap. 6.2). Následně je zjištěna maximální velikost datagramu, kterou je možné od odesilatele přenést k příjemci (více v kap. 6.3).

Ustavení spojení

Pokud je překlad jména a zjištění maximální velikosti datagramu úspěšné, odesílatel se pokusí o „trojcestný handshake“: zašle datagram se zprávou aplikačního protokolu „Hello“ a čeká na příchozí ICMP ***Echo*** zprávu AP typu „Hello Back“ od příjemce. Pokud tato zpráva přijde v definovaném časovém limitu[6], znamená to, že na druhé straně naslouchá příjemce a že *je odesílatel dostupný ze strany příjemce*. Jen v takovém případě si mohou obě strany vyměňovat kontrolní zprávy a na komunikaci můžeme nahlížet jako na *spolehlivou*.

Pokud je handshake neúspěšný, přenos přechází do *jednosměrného režimu*. Odesílatel se i v něm pokusí odeslat data běžným způsobem, nečeká ale na potvrzení o přenosu od příjemce, ten naopak nemůže odesílatele požádat o zaslání dat, která nepřijdou. Úspěšný přenos souboru tedy v tomto případě není garantován, přenos není *spolehlivý*.

Přenos informací o souboru a dat

Následně odesílatel zašle zprávu AP s informacemi o souboru – počtu bajtů v zašifrované podobě a názvu souboru. Název je přenesen v zašifrované podobě. Příjemce otevře soubor, alokuje v něm určené místo a namapuje jej na paměť.

Poté odesílatel začne přenášet data. Data jsou po blocích čtena ze vstupního souboru, šifrována a každý zašifrovaný blok dat[7] je v podobě datové části ICMP *Echo* datagramu odeslán příjemci. Příjemce přijaté bloky ukládá buď do dočasného souboru, nebo do alokované paměti[8].

Potvrzení přijetí, žádosti o chybějící data

Po odeslání všech zašifrovaných dat zašle odesílatel zprávu AP s informací o celkovém počtu odeslaných datagramů, resp. bloků dat (zpráva AP „All Data Sent“). Příjemce tak může ověřit, jestli úspěšně přijal všechna data, v takovém případě odesílá zprávu AP „OK“. Pokud některé bloky chybí a komunikace je ustavena v obousměrném režimu, příjemce nyní může zaslat zprávu AP „Request Resend“, v jejíž datové části určí číselné identifikátory chybějících bloků dat. Odesílatel po přijetí této zprávy znovu odešle požadované bloky a novou zprávu AP „All Data Sent“ a příjemce může požádat o další chybějící bloky. Tento proces je opakován, dokud se k příjemci nedostanou všechny bloky dat.

Dešifrování

Příjemce dešifruje přijatá data s použitím určeného klíče a dešifrovaná data uloží do souboru. Přenos je tím ukončen.

[6] Definováno konstantou `MODE_ESTAB_TIMEOUT_MS` v `common.h`

[7] Blokem dat je myšlena skupina 16B bloků, které jsou výstupem šifrovacího algoritmu AES-256-CBC, která naplňuje dříve zjištěnou maximální velikost datagramu.

[8] Nastaveno konstantou `R_USE_TEMP_FILE` v `common.h`.

6 Implementační detaily

V následujících kapitolách jsou popsány některé principy, kterých aplikace využívá, a organizace zdrojového kódu.

6.1 Struktura kódu

Aplikace je členěna do následujících modulů:

main vstupní bod aplikace, parsování argumentů.

sender logika odesílatele.

receiver logika příjemce.

encryption třída `Crypto` poskytující metody pro šifrování a dešifrování dat.

interface_find třída `InterfaceFinder` poskytující metodu pro zjištění rozhraní, na kterém je dostupná daná IP adresa (viz kap. 6.2).

packet_utils pomocné funkce pro sestavování a parsování IP a ICMP datagramů.

utils pomocné funkce.

V hlavičkovém souboru `secure_string.h` je definice bezpečného alokátoru pro textové řetězce, který při dealokaci vynuluje paměť. Toto je použito pro uložení šifrovacího klíče. Implementace je převzata z [6].

Hlavičkový soubor `common.h` obsahuje konfigurovatelné konstanty, které ovlivňují chování aplikace.

6.2 Zjištění rozhraní a dostupnosti protějšku

Aplikace prochází všechna dostupná síťová rozhraní a pokusí se odeslat ICMP *Echo* zprávu v IP datagramu se zdrojovou IP adresou daného rozhraní (a cílovou adresou zadanou uživatelem). Pokud poté zaznamená odpovídající ICMP *Echo Reply*, předpokládá se, že je tímto rozhraním protistrana dosažitelná. Toto je využito pro IPv4 komunikaci, kdy je BSD schránka nastavena tak, aby zaznamenávala komunikaci pouze z tohoto rozhraní.

Při použití IPv6 adres se tento proces provede také, ale jeho výsledek není použit. IPv6 schránky neumožňují přidružení k rozhraní a při odesílání IPv6 datagramů není možné poskytovat vlastní hlavičky protokolu IPv6. Znalost zdrojové adresy tedy není nutná. [7]

Odesílané ICMP *Echo* datagramy mají *Identification* pole nastaveno na PID aplikace, *Sequence Number* pole nastaveno na 0, datová část obsahuje několik bajtů dat bez významu. Odeslání je považováno za úspěšné, pokud odesílatel zaznamená po odeslání libovolnou ICMP zprávu přijatou z cílové adresy.

Tato funkcionalita je implementována v modulu `interface_find.cpp`.

6.3 Zjištění maximální velikosti datagramu

Zjišťování maximální velikosti odesílaného datagramu je provedeno modifikovaným binárním vyhledáváním. V prvním kroku se aplikace pokusí do schránky zapsat ICMP *Echo* zprávu nakonfigurované maximální velikosti[9]. Pokud schránka odmítne takový datagram odeslat nebo pokud na něj nepřijde ICMP *Echo Reply* odpověď, testovaná velikost je nastavena jako pravá hranice binárního vyhledávání, je

[9] Definována konstantou `STARTING_MPS` v `common.h`.

zmenšena na polovinu a proces je opakován. Pokud je nalezena hodnota maximální velikosti, při které je přenos úspěšný, je tato nastavena jako levá hranice vyhledávání a proces pokračuje s hodnotou průměru aktuální levé a pravé hranice.

Proces je ukončen ve chvíli, kdy je potvrzená hodnota rovna maximální, nebo když by případné zvýšení velikosti bylo menší než nakonfigurovaná hranice[10].

Odesílané ICMP *Echo* datagramy mají *Identification* pole nastaveno na PID aplikace, *Sequence Number* pole roste od 0 s jednotkovým krokem s každým odeslaným datagramem, datová část je vyplněna nulami do testované délky.

Tato funkcionalita je implementována v modulu `packet_utils.cpp`.

6.4 Ustavení spojení a zajištění spolehlivého přenosu

Aplikace na aplikační vrstvě zajišťuje ustavení logického spojení mezi odesílatelem a příjemcem. Všechny odeslané datagramy, které jsou pro přenos významné, mají pole *Identification* ICMP *Echo* hlavičky nastavené na pseudonáhodnou hodnotu vygenerovanou před začátkem komunikace odesílatelem. Všechny datagramy odesílané odesílatelem **při přenosu dat**, včetně závěrečného „All Data Sent“ datagramu, mají tuto hodnotu **o 1 vyšší**. Pokud nedojde ke kolizi vygenerovaných pseudonáhodných hodnot, je díky této identifikaci zajištěno oddělení aplikačního přenosu od jiných ICMP přenosů v síti.

Logické spojení je ustaveno příjmem „Hello“ zprávy AP příjemcem. Tato zpráva je odlišena specifickou hodnotou pole dat[11] a pole *Sequence Number* v hlavičce (nula). Při rozpoznání takového datagramu příjemce zaznamená pole *Identification* z ICMP hlavičky a dále přijímá pouze datagramy se stejnou hodnotou tohoto pole (příp. o 1 vyšší, viz výše).

Pokud se vydaří celý trojcestný handshake, tj. odesílatel zachytí zprávu AP „Hello Back“ od příjemce a příjemce zachytí stejnou zprávu od odesílatele (viz kap. 5), na obou stranách je spojení označeno jako obousměrné. Spojení pak můžeme v jistém směru označit za *spolehlivé* (*reliable* [8, kap. 3.4]): dochází k potvrzení úspěšného či neúspěšného přenosu, aplikační protokol pak zajišťuje také správné pořadí zápisu přijatých dat a možnost zažádat o dodatečné zaslání chybějících dat. V opačném případě je označeno jako jednosměrné, což ovlivňuje logiku přenosu (nedochází k potvrzovací komunikaci z příjemce na odesílatele).

6.5 Šifrování dat

Data jsou šifrována symetrickou šifrou AES-256-CBC s využitím knihovny OpenSSL. Vstupem pro šifrování je kromě dat také *šifrovací klíč* (*encryption key*) a *inicializační vektor* (*initialization vector*, IV). Klíč a IV jsou odvozeny pomocí derivační funkce `PKCS5_PBKDF2_HMAC_SHA1`, kterou poskytuje knihovna OpenSSL [9]. Vstupem této funkce je řetězec označený jako *heslo* a pole bajtů – *sůl*. Jako heslo je použit předdefinovaný řetězec[12] (login autora) nebo řetězec poskytnutý uživatelem (pro uživatele označen jako *klíč* pro lepší pochopení významu). Jako sůl je použit 4B identifikátor spojení (hodnota pole *Identification* ICMP hlaviček). To zajišťuje, že jsou data v každém přenosu zašifrována jiným způsobem.

Šifrování je implementováno v modulu `encryption.cpp`.

[10] Definovaná konstantou `MPS_SEARCH_THRESHOLD` v `common.h`.
[11] `01100110` hexadecimálně (tj. data mají délku 4 B)
[12] Definovaný konstantou `DEFAULT_PASSWORD` v `common.h`.

6.6 Aplikační protokol

Aplikační protokol využívá následujících typů zpráv:

Hello Seq = 0, Data: 4 B: `01100110` (hexadecimálně).
Odesláno odesílatelem pro ustavení logického spojení.

Hello Back Seq = 0, Data: 4 B: `10011001` (hexadecimálně).
Odesláno příjemcem po přijetí „Hello“ a poté znovu odesílatelem pro potvrzení obousměrného přenosu.

File Info Seq = 1, Data: 8 B: celé číslo nesoucí počet přenášených bajtů (délku zašifrovaného souboru); libovolný počet bajtů: název souboru (UTF-8).
Odesláno odesílatelem po ustavení logického spojení.

Data* Seq = 2 a inkrementuje se s každým zaslaným blokem dat, Data: 8 B: pozice daného bloku dat v šifrovaném souboru; zbytek datagramu vyplňují (zašifrovaná) data.

All Data Sent* Seq = 65535, Data: 2 B: číslo Seq posledního odeslaného datagramu typu Data.
Odesláno odesílatelem po odeslání všech dat (i po opětovném odesílání dat, hodnota dat je stejná jako u první zprávy tohoto typu).

OK Seq = 65535, Data: 1 B: `1`.
Odesláno příjemcem po úspěšném přijetí všech dat.

Request Resend Seq = 65535, Data: 1 B: `4`; max. 730 × 2 B: číslo Seq datagramu, který nebyl přijat a je vyžadováno jeho opětovné zaslání.
Odesláno příjemcem po přijetí „All Data Sent“ (nebo vypršení časovače pro přijímání dat), pokud nebyly přijaty všechny bloky dat.

V kódu jsou rezervovány také dva druhy zpráv, které odesílá příjemce jako indikaci chybových stavů, ale v odesílateli nebylo jejich přijímání implementováno:

Error Data: 1 B: `2`.
Odesláno příjemcem po přijetí „File Info“, pokud se nepodaří otevřít soubory pro zápis nebo nastane jiná chyba.

Invalid Key Data: 1 B: `3`.
Odesláno příjemcem po přijetí „File Info“, pokud se nepodaří dešifrovat název souboru.

Reference

1. POSTEL, J. *Internet Control Message Protocol* [Internet Requests for Comments]. RFC Editor, 1981-09. RFC, 792. Dostupné z doi: 10.17487/RFC0792.
2. GUPTA, M.; CONTA, A. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification* [Internet Requests for Comments]. RFC Editor, 2006-03. RFC, 4443. Dostupné z doi: 10.17487/RFC4443.
3. IANA. *Protocol Numbers* [Internet Assigned Numbers Authority]. 2021 [cit. 2021-11-14]. Dostupné z: https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml.
4. KERRISK, Michael. *raw - Linux IPv4 raw sockets* [Linux Programmer's Manual]. 2021 [cit. 2021-11-14]. Dostupné z: https://man7.org/linux/man-pages/man7/raw.7.html.
5. KERRISK, Michael. *gethostbyname* [Linux Programmer's Manual]. 2021 [cit. 2021-11-14]. Dostupné z: https://man7.org/linux/man-pages/man3/gethostbyname.3.html.
6. OPENSSLWIKI CONTRIBUTORS. *EVP Symmetric Encryption and Decryption* [online]. 2019 [cit. 2021-11-14]. Dostupné z: https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption.
7. DEERING, S.; HINDEN, R. *Internet Protocol, Version 6 (IPv6) Specification* [Internet Requests for Comments]. RFC Editor, 2017-07. RFC, 8200. issn 2070-1721. Dostupné z doi: 10.17487/RFC8200.
8. KUROSE, James F.; ROSS, Keith W. *Computer networking: a top-down approach*. 6th ed. Boston: Pearson, 2013. isbn 9780132856201. OCLC: ocn769141382.
9. THE OPENSSL PROJECT AUTHORS. *PKCS5_PBKDF2_HMAC* [online] [cit. 2021-11-14]. Dostupné z: https://www.openssl.org/docs/man1.1.1/man3/PKCS5_PBKDF2_HMAC_SHA1.html.