VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí

Čtečka novinek ve formátu Atom a RSS s podporou TLS

12. listopadu 2022

Vladislav Mikheda

Obsah

1 Úvod

Cílem projektu je implementování RRS čtečky,která bude vypisovat uživatelům informace uvedené ve stažených zdrojích.
Čtečka podporuje formáty RSS 2.0 a Atom.
Takže uživatel může vybrat jakou informace o zdrojích chce vidět kromě záhlaví, informace o autorech zdrojích, asociované URL, informace o čase změny či vytvoření záznamu. Čtečka podporuje jak bezpečné spojení s využitím SSL certifikátu tak i obyčejné. Uživatel může i přidat vlastní ssl certifikát pro zaručení TLS/SSL spojení.
Cílem bylo nejen spojení a stahovaní zdrojů a také provést analýzu a vypsat informaci o které požádá uživatel.

Na obrázku č:1 je možné vidět spuštění čtečky pro výpis informace ze zdrojů https://xkcd.com/atom.xml s využitím lokální schránky SSL certifikátu.
Ve výstupu je uvedená informace o názvu zdroje, názvy jednotlivých článků, jejich aktualizace a asociované URL.

```
❯ ./feedreader 'https://xkcd.com/atom.xml' -u -T -C '/etc/ssl/certs'
*** xkcd.com ***
Soil
URL: https://xkcd.com/2695/
Aktualizace: 2022-11-07T00:00:00Z

Königsberg
URL: https://xkcd.com/2694/
Aktualizace: 2022-11-04T00:00:00Z

Wirecutter Recommendation
URL: https://xkcd.com/2693/
Aktualizace: 2022-11-02T00:00:00Z

Interior Decorating
URL: https://xkcd.com/2692/
Aktualizace: 2022-10-31T00:00:00Z
```

Obrázek 1: Příklad spuštění a výstupu

2 RSS

RSS – (en. Really Simple Syndication) je shrnutí webu, je to rodina XML formátů která se využívá k popisu novinek, blogu a td. Uživatel si může nastavit odběr z různých zdrojů a informace kterou zdroj zobrazuje ve formátu RSS může být jednoduše shromážděna, zpracovaná a předaná uživatelům. Obvykle RSS používá zdroje obsah kterých se často mění víc zde[8] a zde[9]. RSS 2.0 obvykle popisuje nové přidané články, jejich krátký obsah a také zahrnuje v sobě odkaz na plnou verze článku. RSS-kanál je to zdroj který poskytuje informace v formátu RSS viz[9]. RSS formát musí odpovídat specifikaci XML 1.0. Formát RSS je strukturovaný, na nejvyšší úrovni RSS dokumentu musí být <rss> prvek který musí obsahovat verzi RSS tohoto dokumentu. Dál povinným prvkem je kanál <chanel> v kanálu se nacházejí další prvky. Povinné prvky kanálu:

- <title> Název kanálu.
- <link> Odkaz na html verzi tohoto dokladu.
- <description> Krátký popis toho co se nachází v tomto kanále.

Pro účely projektu nás zajímá jen <title>. Kanál může obsahovat v sobě prvek item přičemž množství těch prvků může byt jakýkoliv. Prvek item reprezentuje jednotlivý element kanálu, například jednotlivý článek. Všechny vložené prvky <item> nejsou povinné, ale pro účely projektu nás zajímá:

- <title> Název, například název článku.
- <link> Odkaz na celý text tohoto elementu.
- <author> Autor který tento text napsal.
- <pubDate> Datum publikování elementu.

Více o RSS a prvcích můžete dozvědět zde[1].

3 Atom

Atom – je reprezentován jako jednoduchý a rychlejší způsob čtení a publikování informace na internetu viz[5]. Takže je možný se dívat na atom jako na souhrn dvou technologie: první je to web-standart pro popis informaci a druhá je standart pro jejich publikování. Atom jako RSS musí odpovídat specifikaci XML 1.0. Pro publikování se používá protokol AtomPub který je založen na http, více tady[7].

Na začátku dokladu musí být prvek <feed>. Povinné prvky elementu <feed>:

- <title> Název prvku ¡feed¿.
- <id> Unikátní identifikátor prvku ¡feed¿.
- <update> Datum kdy byly prováděny významné změny

Pro účely projektu nás zajímá jen <title>. V prvek <title> jako jednotlivé elementy budou vnořené prvky <entery> co můžou odpovídat jednotlivým článkům. Na rozdíl od RSS, v Atom u prvku <entery> jsou povinné elementy jako prvky ve <feed>. Pro projekt nás zajímají prvky:

- <title> Název, například název článku.
- <link> Odkaz na celý text tohoto elementu nebo související zdroj. Na rozdíl od RSS tento prvek může obsahovat typ
- <author> Autor který tento text napsal. Na rozdíl od RSS obsahuje v sobe vnořené prvky: povinný <name> a nepovinný <email>.

- `<update>` Datum kdy byly prováděný významné změny, také `<entery>` může obsahovat prvek `<published>` ale na rozdíl od `<update>` je nepovinný, takže pro účely projektu byl zvolen druhý.

Více informace o prvku a Atomu je možné dozvědět zde[6].

4 Knihovna libxml2

Pro analýzu `XML` dokladu byla zvolena knihovna `libxml2`. Je to knihovna s otevřeným zdrojovým kódem, pro analýzu `XML` dokladu. Knihovna poskytuje rozhraní pro přístup k jednotlivým prvkům a atributem prvků `XML` dokladu.

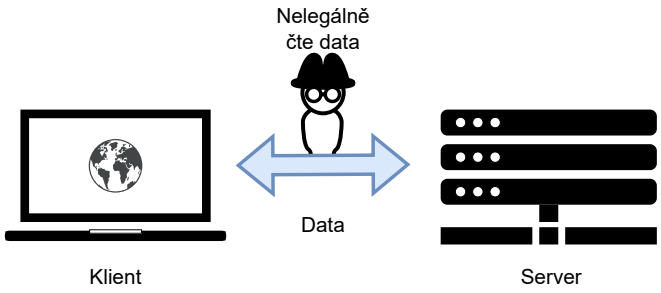
5 SSL/TLS

TLS/SSL protokoly pro bezpečnost předávání dat. Hlavní cíl protokolu je zabezpečit bezpečné předávání dat v nebezpečných sítích. TLS/SSL vložený mezi `transportním` a aplikačním protokolem. V dnešní době:
Aktuální verze protokolu – `TLS 1.3, TLS 1.2 rfc8446`
Už zastaralý a nepoužívaný – `TLS 1.1, TLS 1.0` a všechny verze `SSL`
více zde[4]

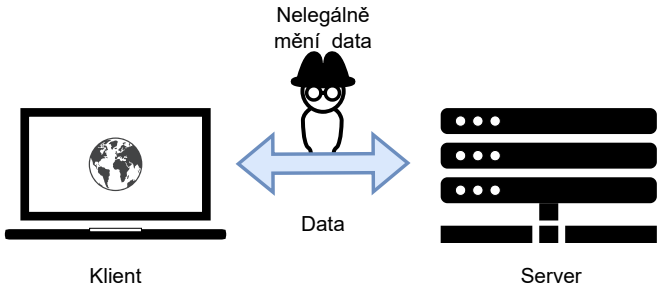
5.1 Bezpečnost předávání dat

Jak už bylo zmíněno nahoře hlavní cíl je bezpečnost dat a zaměření na příští aspekty:

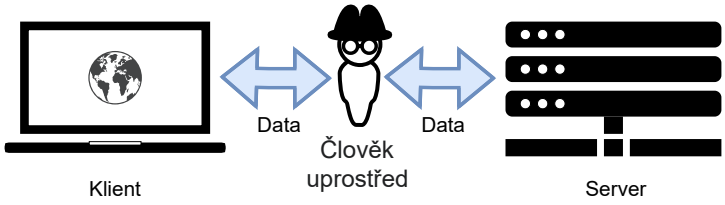
- Soukromí dat. Data mohou být čtené třetí stranou, to je ilustrováno na obrázku č:2. K zachování soukromí dat se používá šifrování.
- Integrita dat. Data mohou být změněny v době předávání od klienta k serveru je ilustrováno na obrázku č:3. Pro zabezpečení integrity jsou použité hašovací funkce(kryptografické).
- Autentizace. Říká se že server je opravdu tím serverem se kterým chcete komunikovat. Může se stát že komunikace bude probíhat přes třetí stranu, například útok „Člověk uprostřed“(en. Man in the middle) je ilustrováno na obrázku č:4. Pro zaručení autentizace se používá digitální podpis a infrastruktury veřejných klíčů (en. Public Key Infrastructure, PKI)



Obrázek 2: Nelegální čtení dat



Obrázek 3: Nelegálně mění data



Obrázek 4: Člověk uprostřed

5.2 TLS spojení

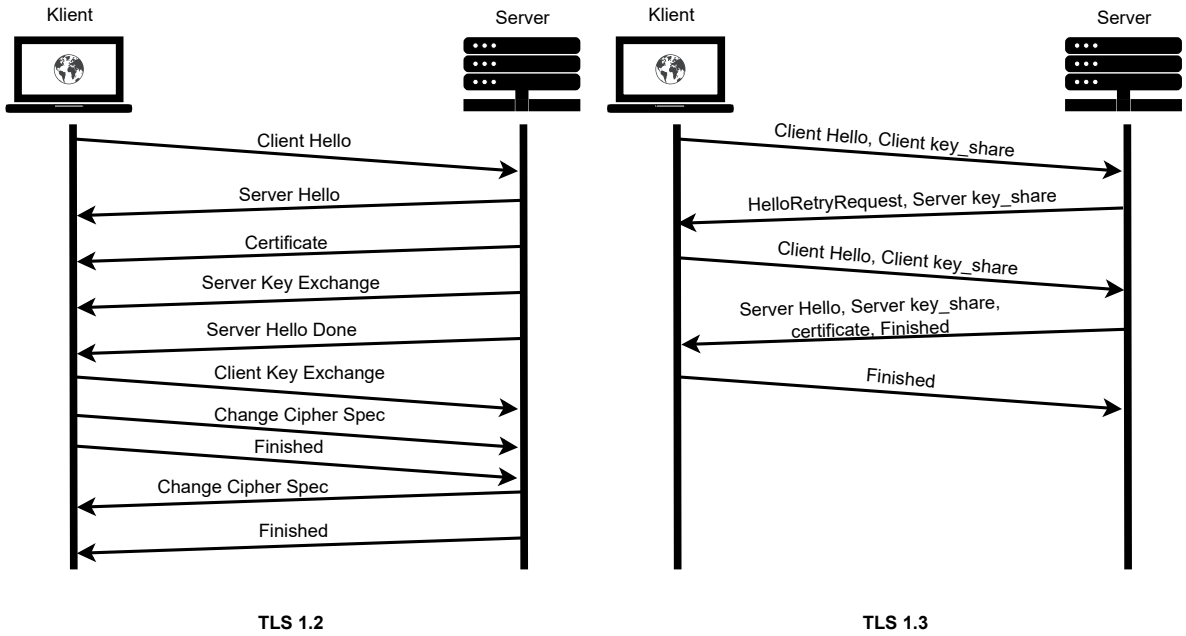
Proč musí být inicializováno TLS spojení:

1. Server a klient musejí se domluvit na tom které kryptografické technologie budou navzájem využívat pro:
    - Algoritmus pro výměnu klíčů. Například: Diffieho-Hellmanova výměna klíčů.
    - Algoritmus pro digitální podpis. Například: RSA.
    - Algoritmus pro symetrické šifrování. Například: AES.
    - Hašovací funkce pro autentizační kód (en. message authentication code). Napriklad: SHA-256.
2. Autentizace serveru.(Někdy musí být potřebná také autentizace klientu, pro naši účely nepoužíváme).
3. Výměna klíčů.

Inicializace spojení TLS 1.3:

1. Klient pošle zprávu „Client Hello“ spolu pošle informaci pro výměnu klíče(en. Pre-shared key)
2. Ne vždy(když server nepodporuje šifrovací algoritmy které sdílí klient), server odešle dotaz na opakované odesílání zprávy „Client Hello“ a přidává algoritmy které podporuje.
3. Ne vždy(když server nepodporuje šifrovací algoritmy které sdílel klient), klient znovu pošle zprávu „Client Hello“ a do informace pro výměnu klíčů generovanou pomocí náboru šifru které server podporuje(když klient je také podporuje)
4. Server na základě informace klienta generuje vlastní informace pro výměnu klíčů(en. Pre-shared key). Pošle serveru „Server Hello“ a generovanou informace, spolu s certifikátem. A také pošle zprávu „Finished“.
5. Klient ověří certifikát serveru, bude vygenerován tajný klíč, a server pošle zprávu „Fineshed“

Příští data už budou předaná po zabezpečenému spojení. Když server a klient nepoužívají ani jeden stejný algoritmus pro šifrování, to prostě zruší připojení viz[3]. Na obrázku č:5. je možné vidět rozdíl mezi inicializaci připojení `TLS 1.2` a `TLS 1.3`



Obrázek 5: Inicializace spojení TLS 1.2 a TLS 1.3

6 Knihovna `OpenSSL`

Pro implementace projektu byla využitá knihovna `OpenSSL`. OpenSSL je to knihovna s otevřeným kódem která zahrnuje v sobě sadu nástrojů pro `TLS` a také kryptografické funkce a podporuje mnoho různých kryptografických algoritmů více zde[10].
Je možné říct že pro naši účely knihovna poskytuje vysokoúrovňové rozhraní pro nezabezpečené a zabezpečené připojeni s využitím `TLS`. Knihovna byla vybrána z důvodu doporučení pro projektový úkol. Také po přečtení doporučené stránky viz[2] bylo zjištěno, že knihovna představuje sadu funkcí pro připojovaní k serveru,komunikace se severem, ověření certifikátu, připojování s použitím `TLS`, při tom skrývá vnitřní implementace. Například při inicializaci `TLS` spojení knihovna samostatně provádí všechny kroky které byly popsané v sekci 5.2. Pro účely projektu byla využita jen malost toho co `openssl` v sobě zahrnuje.

7 Implementace

7.1 Úvod a struktura programu

Pro zpracování projektu byl využit jazyk C++. Program byl implementován s využitím objektové orientovaného přístupu. Bylo zvoleno rozdělení na třídy: `ParserArguments` třída pro analýzu vstupních argumentů, `UrlParser` třída pro analýzu url adresy která buď byla zadaná uživatelem jako argument programu nebo z feed souboru, `Connect` třída pro připojení k serveru a posílání http dotazu, je obalem knihovny `openSSL`, `XMLParser` třída pro analýzu již stažených `RSS/Atom` dokumentů a pro výpis na standardní vystup analyzované informace `FeedReader` je to centrální třída která řídí všechny procesy, `Error` třída pro zpracování varování a chyb a také pro výpis jich na standardní chybový vystup.

7.2 Analyzování argumentů

Pro analýzu argumentů jak bylo výše uvedeno slouží třída `ParserArguments` instance které je vytvořena v metodě `read` třídy `FeedReader`.
Analýza argumentu běží v smyčce kde na začátku bude zjištěno který parametr byl zadán.Určujeme který argument je který pomocí konstrukce `if`, kde hodnoty argumentu se zapisují do jednotlivých proměnných tříd.

7.3 Analýza Url adresy

Hlavní metodou třídy `UrlParser` je metoda `parse` která může být vyvolaná pro analýzu url adresy. Kontrola prochází pomocí regulárního výrazu.
Regulární vyraz je napsán takovým způsobem aby obsahoval v sobě rozdělení do skupin podle jednotlivých součástí url adresy(například, doména a port). Na začátku probíhá kontrola výrazu aby url odpovídala schematu regulárního výrazu pomocí metody `std::regex_match`.
Když adresa je zadaná správně a odpovídá zvolenému regulárnímu výrazu, metoda uloží informace o nalezených shodách skupin v proměnu která reprezentuje specializaci instance třídy std::match_results.
Dál bude zjištěno jaké schéma se používá v url adrese a bude zkontrolován port, je-li port nezadán bude nastaven podle schematu.

7.4 Připojeni k serveru

Připojení k serveru je prováděno s využitím knihovny `OpenSSL`, v projektu byla zabalena a je reprezentována třídou `Connect`.
Na začátku provádíme inicializace knihovny, více zde. Takže musíme vytvořit ukazatel na instanci třídy `bio`. Knihovna `Bio` je použita v `openSSL` jako náhrada knihovny soketu viz[2]. Pak zjistíme který typ připojení budeme používat: obyčejné nebo zabezpečené.
Pro obyčejné připojení využijeme jen jednu funkci `BIO_new_connect` které jako parametry budou předány doménové jméno a port, pak je nutné kontrolovat zda spojení proběhlo dobře a může poslat `http` dotazy na server.
Zabezpečené připojení probíhá složitěji ale bere na sebe více práce jak už bylo popsáno v sekci 5 `openSLL` Na začátku spojení vygenerujeme ukazatel na strukturu ve které bude schovány všechny nutné informace `SSL`. Dál je nutné přidat schránku certifikátu pomocí které `openSSL` ověří certifikát který nám pošle server. V našem případě jsou čtyři varianty:

1. Uživatel nebude nic přidávat, `openSSL` automaticky zvolí standardní adresář pro uchování certifikátu.
2. Uživatel přidá adresář, `openSSL` bude využívat přidaný adresář.
3. Uživatel může zadat konkretní certifikát, `openSSL` bude ověřovat certifikát který pošle server pomocí zadaného certifikátu uživatelem

4. Uživatel zadá adresář a soubor certifikátů najednou, openSSL zkusí ověřit certifikát serveru dvěma způsoby.

Dál je prováděno nastavení parametru pro bio víc zde[2], a připojení k serveru, po ověření úspěchu spojení bude ověřeno zda certifikát který server poslal je opravdu validní, když je to tak může být poslán http dotaz, když ne zamítneme spojení.

7.5 Komunikace s serverem

Pro účely projektu je využit protokol http 1.0. Pro získání RSS/Atom dokladu, server musí být o tom požádán, pro teto účel využijeme http get dotaz. Při dotazu server je požádán aby po jeho odpovědi spojení bylo zavřeno Connection:Close. Dál bude přijata odpověď serveru. Při čtení odpovědí nebo zápisu našeho dotazu muže vzniknout chyba, v openSSL je vnořena kontrola lze-li chybu která vznikla opravit,když ano tak bude zkoušeno ještě jednou, když ne uživatelů bude vypsáno o tom ze vznikla chyba. Po dotazu musí byt zkontrolováno že byla přijata pozitivní odpověď, jako pozitivní odpověď byly zvoleny odpovědi s kódem 200-299.

7.6 Analýza XML

Po přijetí a ověření http odpovědi, je nutné provést analýzu přijatého dokladu, před analýzou http hlavička je odstraněna. Pro účely analýzy je využita knihovna libxml2. Analýza dokladu je implementována ve třídě XMLParser.
Přidáme doklad do struktury xmlDoc, když všechno proběhne dobře, doklad odpovídá standardu XML.
Před analýzou prvku zjistíme kterému formátu odpovídá dokument, je to bylo popsáno v sekci 2 a v sekci 3.
Analýza RSS/Atom probíhá pomocí názvů prvků podle jejich vnořenosti, není zaručeno že dokument 100 procentně odpovídá formátu který je zvolen na nejvyšší úrovni dokumentu. Jak všechny nutné prvky budou zpracované bude vypsána nutná informace na standardní vystup. Při tvoření projektu byla zvolena částečná kontrola dokumentu, například když dokument formátu RSS neobsahuje v kanálu prvek <title> a má několik článku, název kanálu z prázdného řetězce bude změněn na ***----------***, kvůli tomu aby vystup byl citelný a víc v sekci 8.

7.6.1 Třída `FeedReader`

Jak bylo popsáno nahoře třída FeedReader je řídicí třídou programu. V ni jsou postupně prováděny té akce které už byly popsané. Na začátku bude prováděna analýza argumentu. Pak bude zjištěno zadali uživatel url nebo feedfile, když feedfile bude otevřen a zpracován, bude zjištěno zda řádek je prázdný nebo je komentářem pomocí regulárního výrazu, dál data ze souboru budou zapsány do seznamu. Dál začíná hlavní smyčka programu a cyklí dokud nebude vypracované všechny url adresy. Na začátku smyčky provádíme analýzu url adresy pak provedeme připojení k serveru, dál bude poslán dotaz, bude přijata odpověď a zpracujeme odpověď, a bude provedena analýza přijatého dokladu z výpisem informace na standardní vystup. Když v nějakých z kroků něco se nepovede, bude vypsáno varování nebo chyba na standardní chybový vystup a zvolíme další url adresu.

8 Zajímavé pasáže implementace

8.1 openSSL

Při testování projektu bylo zjištěno že nejde připojit do některých serverů které vyžadují bezpečnostní spojení. Bylo zjištěno že při spojení a provádění „handshake“,`openSSL` v některých případech vyžaduje aby bylo jistě zadáno jméno serveru, pro zadávaní jména byla využita funkce `SSL_set_tlsext_host_name`.

8.1.1 Verze knihovny

Pro překlad a spuštění projektu je nutné využít knihovny `openSSL` do verzi 3.0.

8.2 libxml2

8.2.1 Vypisování informace

Při výpise informace byly zvoleny vlastní změny:

1. Když v sekci `<rss>/<feed>` prvek `<title>` není uveden pro lepší zobrazení, prázdný řetězec bude vyměněn na `***----------***`
2. Když prvek `<title>` není uveden v článku i uživatel vyžaduje nějakou informace kromě nazvu článku například autora a td., prázdný řetězec bude vyměněn na `----`
3. Když nějaký prvek který musí být vypsán není uveden v dokumentu tak prostě nebude vypsán.
4. Když zdroj je ve formátu atom a je požádáno aby byl vypsán autor, když mimo jména autora v dokumentu je uveden jeho emailová adresa, bude vypsáno `jméno(email)`.
5. Když bude vypsáno datum publikace v dokumentu ve formátu `Atom` bude vypsáno `Aktualizace:datum` u `RSS` bude vypsaná `Zverejneno:datum`.

8.3 Vypínání varovaní

Při testování bylo zjištěno že při nesprávném `RSS/Atom` dokladu knihovna `libxml2` vypisuje na standardní vystup varování, a bylo to vypnuto pomocí přenastavení funkce která zpracovává varování.
Přenastaveno bylo na vlastní funkci která prostě ignoruje varování pomocí funkce `xmlSetGenericErrorFunc` a `xmlThrDefSetGenericErrorFunc`.

9 Testování

Testování proběhlo v několika etapách. Program byl testován při návrhu, testování probíhalo na vlastním systému[1] bez virtualizace.
Pro začáteční testovaní byly napsány „Unit“ testy. „Unit“ testy byly napsány pro testování třech tříd `ParserArguments`, `URLParser`, `Connect`. Byly otestovány mimořádné varianty, například když uživatel nezadal url adresu a td.
Testy byly využity jak v dobu napsaní kódu tak i při celostním testovaní programu.
Pro testování programu jako celku byly napsány testy. Testy byly napsány jako jediný skript v jazyce `Python`. `Python` byl zvolen pro zjednodušení práce a zamezení se na problému testování. Kvůli tomu že některé zdroje které byly použity pro testování se mohou měnit, nemůže probíhat několik testů ale to bylo vyřešeno tím že bude vypsáno varovaní o kontrole aktualizace zdrojů, ale většina zdrojů byla použita z ukázek RSS a nemění se.

```
❯ make test
g++   tests/unit/*.cpp Connect.cpp FeedReader.cpp ParseArguments.cpp UrlParser.cpp XMLParser.cpp -o testfile  -I/usr/include/lib
xml2  -static-libstdc++ -lcrypto -lssl -lxml2
./testfile
TEST:Tests ArgumentParser are SUCCESS
TEST:Tests URLParser are SUCCESS
TEST:Tests Connect are SUCCESS
g++  Connect.cpp Error.cpp FeedReader.cpp main.cpp ParseArguments.cpp UrlParser.cpp XMLParser.cpp Connect.h Error.h FeedReader.h
 ParseArguments.h UrlParser.h XMLParser.h -o feedreader -I/usr/include/libxml2  -static-libstdc++ -lcrypto -lssl -lxml2
python ./tests/complex/test.py ''
Tests all:
Normal url test success
All arguments test success
Certificate dir test success
Certificate file test success
Certificate file and dir test success
Atom test success
Feed list test success
```

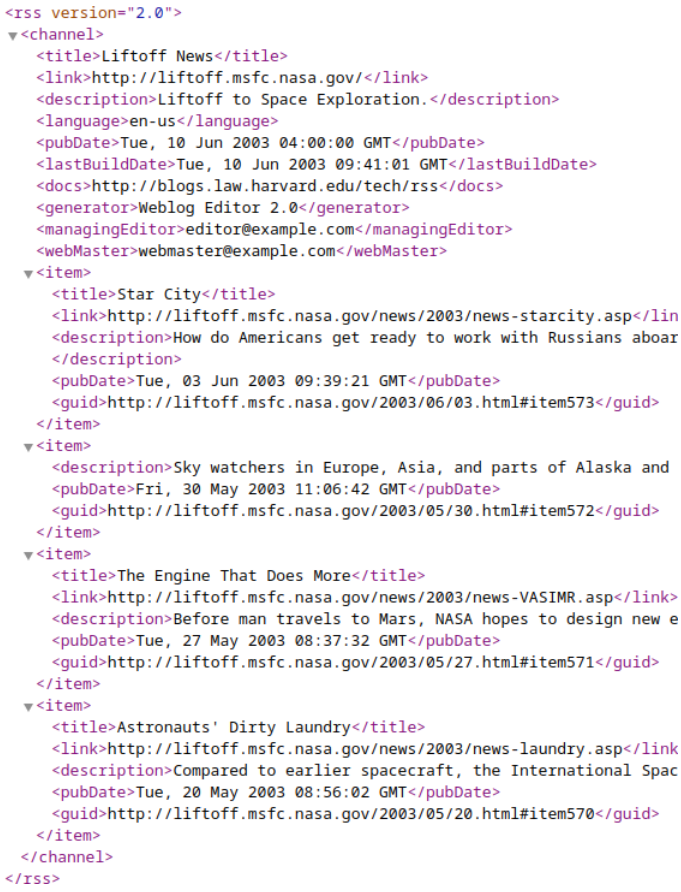
Obrázek 6: Příklad spuštění testů

Podle obrázku č:6 jde vidět že test prochází.
Takže testování bylo prováděno manuálně, bylo ověřeno všechny-li požadované elementy program vypsal a jestli se shodují elementy s elementy v `RSS/Atom` dokladu.

```
❯ ./feedreader 'https://www.rssboard.org/files/sample-rss-2.xml' -u -T
*** Liftoff News ***
Star City
URL: http://liftoff.msfc.nasa.gov/news/2003/news-starcity.asp
Zverejneno: Tue, 03 Jun 2003 09:39:21 GMT

----
Zverejneno: Fri, 30 May 2003 11:06:42 GMT

The Engine That Does More
URL: http://liftoff.msfc.nasa.gov/news/2003/news-VASIMR.asp
Zverejneno: Tue, 27 May 2003 08:37:32 GMT

Astronauts' Dirty Laundry
URL: http://liftoff.msfc.nasa.gov/news/2003/news-laundry.asp
Zverejneno: Tue, 20 May 2003 08:56:02 GMT
```

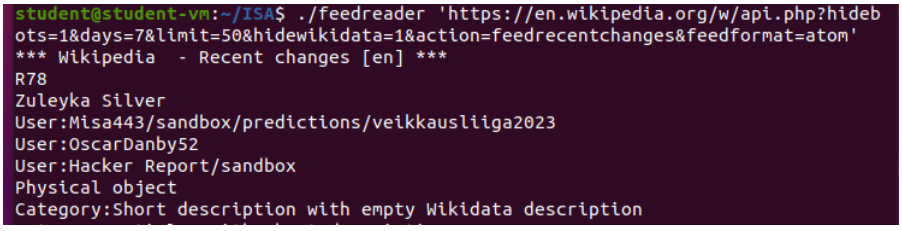
Obrázek 7: Příklad manuálního testování

[1] Arch Linux

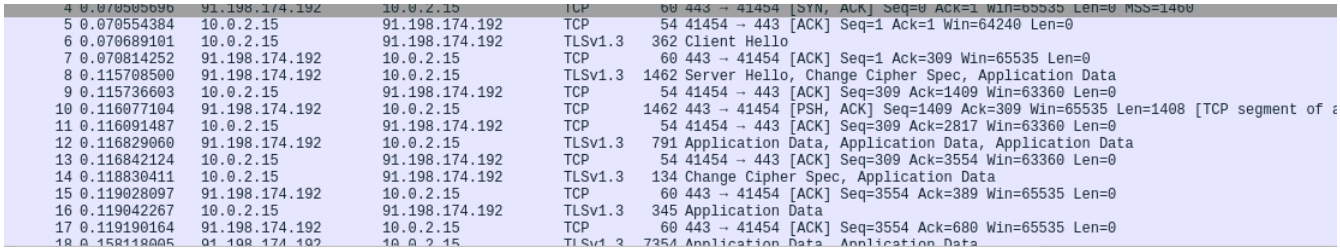


Obrázek 8: Příklad manuálního testování

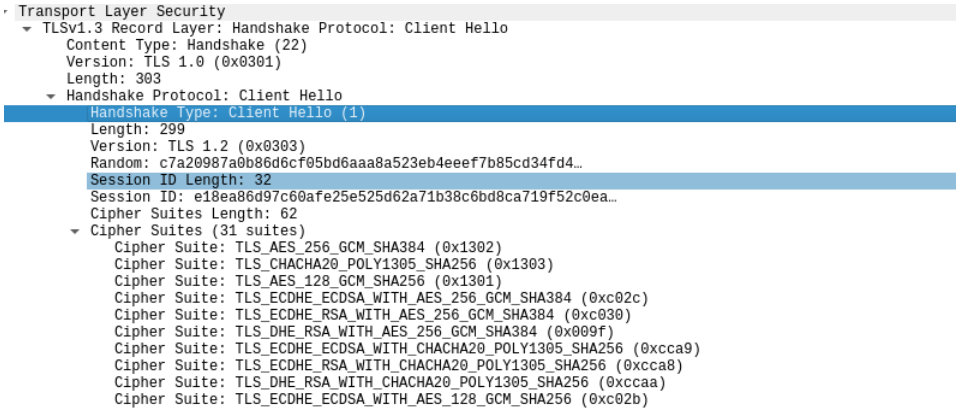
Podle obrázku č:7 a obrázku č:8 lze vidět že program na výstupu vypíše to co požádal uživatel a výstup se shoduje s obsahem `RSS` dokumentu.
Také pro testování byl využit virtuální nastroj s `ubuntu` aby bylo možný se podívat na komunikace ve `wiresharku`.



Obrázek 9: Testování komunikace



Obrázek 10: Testování komunikace



Obrázek 11: Testování komunikace

Podle obrázku č:8,č:9 č:10 lze vidět že komunikace je prováděna s využitím `TLS 1.3` jak bylo popsáno v sekci 5

Po testování lze říct že program funguje správně a výpis informace prochází podle požadavku uživatele a komunikace s serverem je nastavena správně.

10 Návod na použití

Pro překlad a spuštění projektu je nutné využít knihovny `openSSL` do verzi 3.0.
Po kompilaci programu je vytvořen soubor `feedreader`.
Spuštění: ./feadreader <url adresa> | -f <feedfile>> [-c <certfile>] [-C <certaddr>] [-T] [-a] [-u] [-h]

- `url` adresa zdroje z kterého bude přijata informace pro výstup programu.
- `-f <feedfile>` cesta do souboru `feedfile`. Soubor musí obsahovat v sobě jednotlivé zdroje, každý zdroj na jednom řádku, takže je možnost komentářů, řádek s komentářem musí se začínat
- `-c <certfile>` cesta do certifikátu který bude využít pro ověřování serveru.
- `-C <certaddr>` cesta do úložiště certifikátů, které bude možné použit pro ověřování serveru
- `-T` pro každý záznam se zobrazí čas publikovaní nebo čas změny
- `-a` pro každý záznam se zobrazí autor
- `-u` pro každý záznam se zobrazí asociované url
- `-h` vypíše návod na použití programu

Parametry mohou být v jakémkoliv poradí. Parametry musí být odděleny mezerami.
Je nutné aby byl zadán jeden z parametru url nebo feedfile. Všechny odstátní parametry nejsou povinné, také lze zadat spolů `-c <certfile>` a `-C <certaddr>`.

Literatura

[1] Cadenhead, R.: RSS 2.0 Specification. [online]. rev. 2.dubna.2014. [vid. 2022-11-11].
Dostupné z: https://goo.su/ZJ5oGl

[2] Corporation, C. I.: Secure programming with the OpenSSL API. [online]. rev. 15.srpna.2018. [vid. 2022-11-08].
Dostupné z: https://developer.ibm.com/tutorials/l-openssl/

[3] Corporation, C. I.: The TLS 1.3 Protocol. [online]. rev. 25.srpna.2022. [vid. 2022-11-08].
Dostupné z: https://goo.su/OHfz5Oo

[4] Corporation, C. I.: How an SSL connection is established. [online]. rev. 6.dubna.2022. [vid. 2022-11-08].
Dostupné z: https://goo.su/nB1pM

[5] Enabled, A.: Atom Enabled. [online]. rev. října.2007. [vid. 2022-11-11].
Dostupné z: https://www.rfc-editor.org/rfc/rfc5023

[6] Enabled, A.: Atom Enabled. [online]. rev. října.2007. [vid. 2022-11-11].
Dostupné z: http://www.atomenabled.org/developers/syndication/

[7] J. Gregorio, E.: The Atom Publishing Protocol. [online]. rev. 6.dubna.2019. [vid. 2022-11-11].
Dostupné z: http://www.atomenabled.org/

[8] Wikipedia, t. f. e.: RSS. [online]. rev. 17.října.2022. [vid. 2022-11-11].
Dostupné z: https://cs.wikipedia.org/wiki/RSS

[9] Wikipedia, t. f. e.: RSS. [online]. rev. 30.října.2022. [vid. 2022-11-11].
Dostupné z: https://en.wikipedia.org/wiki/RSS

[10] Wikipedia, t. f. e.: OpenSSL. [online]. rev. 9.listopadu.2022. [vid. 2022-11-08].
Dostupné z: https://en.wikipedia.org/wiki/OpenSSL