Vysoké učení technické v Brně
Fakulta informačních technologií

Síťové aplikace a správa sítí

Čtečka novinek ve formátu Atom s podporou TLS

19. listopadu 2018 Martin Dvořák

Obsah

1 Úvod

Tato dokumentace popisuje protokoly HTTP1.1 a HTTPS (za pomocí TLS). Dále vysvětluje standardy RSS1.0, RSS2.0 a Atom standard *feed souborů*. Jsou zde uvedeny i rozšíření jako je *Dublin Core*. Dokumentace primárně popisuje aplikaci, která za využití výše zmíněných a níže vysvětlených protokolů, implementuje zpracování *feed* zdrojů.

Aplikace se zaměřuje na jednotlivé články, které se nacházejí ve zdrojích a srozumitelně je zobrazuje uživateli. Druhá polovina dokumentu je věnována implementačním detailům a náročným částem projektu. V posledních částech dokumentace jsou uvedeny ukázkové příklady spuštění a vysvětlen testovací skript.

2 Úvod do HTTP/S

HTTP[1] je zkratka pro *Hypertext Trasfer Protocol*. Dnešní nejnovější verze HTTP1.1[2] slouží pro výměnu dokumentů v nejrůznějších formátech.

Tuto vlastnost zajišťuje rozšíření *MIME* [3]. Např. je možné přenášet hlavně dokumenty - html,xml, obrázky - jpeg,png, audio/video - mpeg, mp4 a aplikace jako javascript, json atd.

HTTP a poštovní protokoly jsou vůbec nejvíce používány v počítačových sítích. Pomocí jednoznačného URL je identifikován libovolný soubor na internetu. HTTP nepodporuje šifrování, a proto je z hlediska bezpočnosti nedostatečný. Je tedy nahrazován HTTPS spojením, které je šifrované na TCP vrstvě pomocí SSL[4] nebo TLS[5].

2.1 HTTP spojení

HTTP spojení je navázáno na žádost klientské aplikace, převážně jde o internetové prohlížeče. Aplikace může navázat perzistentní spojení pro více dotazů nebo nestálé spojení pro jeden dotaz, k tomu je využit atribut v hlavičce požadavku *Connection* viz níže.

HTTP spojení běží většinou na TCP protokolu a na dobře známém portu (*well known ports*) číslo 80. Dovoluje i specifikovat libovolný jiný port. Před zasláním požadavku na server se musí navázat spojení mezi účastníky pomocí *tří-cestného handshakingu*. Protokol TCP garantuje doručení paketu a jejich správné řazení.

V zaslaném požadavku je nutné specifikovat tzv. *dotazovací metodu*. Protokol HTTP jich implementuje celou řadu. např: GET, POST, HEAD, DELETE, PUT. Pro účely projektu je důležitá nejpoužívanější metoda *GET*[6]. Defaultní metoda *GET* využívá *URI*[6] formát dokumentu. Metoda pouze získává požadovaný dokument pomocí specifikované *URI* adresy dokumentu a neměla by mít jiný efekt.

Jelikož je nutné získat pouze daný zdroj, není nutné používat jiné metody. Metoda *HEAD* může posloužit velice dobře při ladění aplikace, neboť je žádáno pouze o hlavičku odpovědi bez těla s daty.

Metody typu *GET*, *HEAD* jsou definovány jako nebezpečné z pohledu změny informací na serveru, tedy jsou vhodné pouze k získávání dat ze serveru.

2.2 HTTPS spojení

Jak již bylo zmíněno výše, HTTP spojení není bezpečné z pohledu šifrování a přenosu dat. Tuto vlastnost implementuje do protokolu *SSL* nebo *TLS* pomocí asymetrického šifrování. HTTPS tedy používá protokol HTTP ve spojení s SSL nebo TLS. komunikace implicitně probíhá na *well known portu* 443.

Při požadavku na šifrovanou komunikaci musí projít mezi účastníky *handshake* inicializační protokol.

2.2.1 TLS

Je nejnovější protokol pro šifrování komunikace. Pomocí asymetrické kryptografie poskytuje *autentizaci* a bezpečnost při komunikaci. Typicky je autentizován pouze server, tudíž si klient může být jistý, s kým komunikuje.

Zahájením komunikace si obě strany vymění informace o TLS, a poté následuje vyměna klíčových hodnot k šifrování. V závislosti na šifrovacích algoritmech jsou vyměněna data. Například se používá pro šifrování *RSA*[8] nebo *Diffie–Hellman*[9][10] algoritmus

Server během zahájení komunikace posílá svůj certifikát založený na *X.509*[7]. Klientská strana jej ověřuje u certifikační jednotky. Pokud dojde k úspěšnému ověření certifikátu, server je autentizovaný.

3 Hlavička HTTP

Zpracování hlavičky u HTTP komunikace je esenciálním prvkem celé komunikace. Hlavička obsahuje informace o samotné komunikaci ve formě návratového kódů a časové známky. Dále jsou v ní obsaženy metadata samotného přenášeného dokumentu.

Délka hlavičky je proměnlivá. každý jeden záznam hlavičky musí končit koncem řádku v systému Microsoft Windows. Tedy hodnotami *carriage return* hexadecimálně *0D* a *line feed*, čemuž odpovídá hexadecimální hodnota *0A*. Konec celé hlavičky je implementován pomocí *dvojitého* odřádkování (hexadecimálně *0A 0D 0A 0D*).

3.1 Hlavička požadavku

Hlavička požadavku se skládá z výběru metody, která se má na serveru provést. Následuje adresa konkrétního souboru na serveru, který chceme ovlivnit. První řádek zakončuje verze HTTP. Následuje klíčové slovo *Host*, které specifikuje DNS *jméno* serveru případně jeho *IP adresu*.

Další velmi důležitou položkou je druh spojení mezi účastníky, uvedený klíčovým slovem *Connection*. Udává, zda se bude vytvořené spojení udržovat *keep-alive*. Užitečné v případě zasílání více požadavků v blízké době, odpadá takto režie na znovu ustálení spojení. Opakem je spojení typu *Close*, kde po vyřízení požadavku spojení mezi serverem a klientem zaniká.

V položce *User-Agent* specifikuje klientská aplikace svoji totožnost. Následovat můžou položky specifikující typ a parametry dokumentu. Položka *Accept* uchovává informaci, jaké typy dokumentu klient požaduje. Daná položka je ve formátu *MIME*[12][13]. Dále lze specifikovat kódování dokumentu či jeho jazyk, přes klíčová slova *Accept-Encoding* a *Accept-Language*.

Pro webové stránky typu *PHP* lze vkládat do hlavičky *HTTP* protokolu malé informace ve formě *Cookie*. Těmito informacemi se dají vytvářet stavy mezi serverem a klientem na protokolu, který je beze-stavový.

3.2 Hlavička odpovědi

V prvním řádku odpovědi se nacházejí informace o verzi protokolu HTTP a návratový kód (*status code*), který vrací server na daný požadavek. Kód je tvořen tříciferným číslem. První číslice rozděluje kód do pěti skupin. Skupina *1xx* obsahuje informační odpovědi. *2xx* oznamuje úspěšnost dotazu. *3xx* udává přesměrování dotazovaného URL. Skupina *4xx* obsahuje definice chyb na straně klienta, nejčastěji *špatný požadavek*(400) nebo *nenalezeno*(404). Poslední skupina *5xx* reprezentuje skupinu chyb, které se ocitly na straně serveru.

Následuje položka *date*, která udává, kdy byl požadavek obsloužen. V závislosti na požadavek jsou nastaveny parametry *keep-alive* spojení nebo v druhém případě je zopakováno *Connection: close*. Je uchována informace o tom, jaký typ serveru daný požadavek obsloužil např: `Server: Apache`.

Pro tento projekt jedna z nejdůležitějších hlaviček *Transfer-Encoding* obsahuje informaci o přenosovém kódování daného požadavku. Tedy v nastavení *Chunked* je nutné aplikačně ošetřit správné interpretování odpovědi.

Dalšími dodatečnými hlavičkami nás server informuje o vlastnostech dokumentu.

4 Zdroje (Feed)

Aplikace musí zpracovávat následující xml protokoly. *Parsování* je prováděno pomocí knihovny *libxml*. V podkapitolách bude popsáno chování aplikace při extrahování dat z těchto zdrojů. Půjde především o informace o autorovi, URL odkazu a Poslední změně.

4.1 RSS1.0

Xml kořenový prvek daného protokolu je párový tag `<rdf:RDF>`. Název celého zdroje se nachází v párovém prvku `<channel>` a plní funkcí hlavičky protokol. Informace o jednotlivých článcích jsou v párovém prvku uloženy `<item>`. V tomto prvku se nacházejí potřebné tagy `<title>` a `<link>`, ze kterých extrahuje titulek článku a URL odkaz s ním spojený.

Jak je vidět RSS1.0 neimplementuje informaci o autorovi článku a ani neudržuje časovou značku poslední aktualizace.

4.1.1 Dublin Core

Rozšíření RS1.0, které implementuje do prvku `<item>` námi hledané další informace. V `<dc:date>` lze naleznout časový záznam poslední aktualizace článku. V `<dc:creator>` aplikace hledá autora článku. Pokud není zadán titulek pomocí RSS1.0 tak jej aplikace získává z rozšíření `<dc:title>`.

4.1.2 Syndication

Rozšíření, které se zaměřuje na časové změny v dokumentu. Aplikace reaguje na prvku `<sy:updateBase>` kde se nastavuje čas aktualizace. Pokud je zadán prvek `<sy:updatePeriod>`, je uživateli zobrazen údaj `Periodická aktualizace od:` časové razítko s krokem: časová jednotka.

Pokud je uvedeno více časových razítek, aplikace zaznamenává a vypisuje pouze poslední z nich, tedy nejaktuálnější.

4.2 RSS2.0

Novější ze specifikací RSS, Obsahuje mírné odlišnosti od předchůdce. Kořenový prvek je přejmenovaný na `<rss>`. Prvek `<channel>` plní stále roli hlavičky xml dokumentu, zjišťuje se z ní titulek pro zdroj. Je pozměněná její struktura, tedy již prvky typu `<item>` jsou v těle prvku `<channel>`

Prvek `<item>` aplikaci poskytuje stejné prvky jako RSS1.0. rozšířené o `<author>`, ze kterého je čerpán autor daného článku a `<pubDate>`, který slouží k určení poslední změny článku.

4.3 Atom

Protokol atom má rozdílnou strukturu dokumentu. Kořenový prvek `<feed>` obsahuje informace o názvu zdroje. Obsahuje i informace o článcích, které jsou uloženy v elementu `<entry>`

Záznam o titulku nalezneme v prvku `<title>`. U prvku `<link>` je nutno hledat odkaz v atributu `href` nikoliv jako hodnotu prvku. Časovou stopu nalezneme v prvku `<update>`. Atom implementuje více informací o osobě. Tedy je možné zjistit i e-mail autora pokud je zadán. V prvku `<autor>` se nalezne prvek s jménem autora `<name>` a jeho e-mailovou adresu `<email>`.

5 Komunikační protokol a návrh aplikace

Komunikační protokol popisuje standart RFC 2616. Zařízení mezi sebou komunikují pomocí *TCP* soketu na portu *80* při nešifrované volbě a na portu *443* při šifrované komunikaci. Tato část se zabývá konkrétní implementací protokolu *HTTP/1.1* a její šifrovanou alternativou a následně zpracování a vyfiltrování důležitých informací z *xml* formátu odpovědi. Komunikace je tvořena pomocí soketu *BIO*[14] z knihovny *OpenSSL*. Tento druh soketu odporuje jak šifrované tak nešifrované komunikace.

5.1 Formát HTTP dotazu

V této podkapitole bude prakticky předvedena hlavička, kterou odesílá aplikace na server. Aplikace se bude dožadovat souboru na adrese *http://www.fit.vutbr.cz/news/news-rss.php.//*

Nutné je zkontrolovat zda je na vstupu *http* nebo *https* protokol. V prvním kroku je nutné rozdělit *URL* na část, která popisuje server tedy *www.fit.vutbr.cz* a cestu k požadovanému souboru */news/news-rss.php*. V dané ukázce není uvedený port komunikace a v kombinaci s *http* bude použit implicitní *port 80*.

Aplikace požaduje spojení typu *close*. Není nutné udržovat spojení při vytvoření pouze jednoho požadavku. Feedreader specifikuje *MIME* typ přijímaného dokumentu a *znakovou sadu*. A v poslední řadě zadefinuje *User-Agent* prvek na hodnotu tvůrce aplikace.

```
GET /news/news-rss.php HTTP/1.1
Host: www.fit.vutbr.cz
Connection: Close
User-Agent: FeedreaderXdvora21
Accept: xml
Accept-Charset: utf-8
```

Každý řádek je ukončen dvojicí znaků hexadecimálně *0A 0D*. Hlavička je ukončena přidáním dvojice znaků na konec hlavičky.

5.1.1 Aplikační realizace

Implementačně je tato část realizována funkcí *fill_packet*, která vrací ukazatel na řetězec. Řetězec reprezentuje celou *HTTP* hlavičku. Funkci je předána relativní cesta na serveru a *DNS* jméno serveru. Funkce vytváří postupným kopírováním do řetězce výslednou hlavičku.

5.2 Formát HTTP odpovědi

Tato podkapitola navazuje na podkapitolu 5.1 v logické návaznosti neboť bude zde popsána odpověď na *http://www.fit.vutbr.cz/news/news-rss.php*.

```
HTTP/1.1 200 OK
Date: Mon, 05 Nov 2018 16:47:57 GMT
Server: Apache
Content-Location: news-rss.php.cz
Vary: negotiate,accept-language
TCN: choice
Connection: close
Transfer-Encoding: chunked
Content-Type: text/xml; charset=utf-8
Content-Language: cs
Expires: Mon, 05 Nov 2018 16:47:57 GMT
```

Aplikace nejdříve kontroluje, zda došlo ke korektní odpovědi. Odpověď proběhla v pořádku dle návratového kódu *200*. Následně je ověřeno, zda položka *Content-Type* odpovídá formátu *xml*. Implementačně je to realizováno funkcí *readhead* a následně *check_contetn_type*. V dalším kroku je zjistěno, zda je zpráva nějak kódována.

5.3 Transfer-Encoding

Dalším důležitým prvkem hlavičky je hodnota *Trasnfer-Encoding*, která udává, zda je tělo odpovědi kódováno. Hodnota *chunked* vkládá hexadecimální číslice do těla zprávy a je nutné tyto čísla správně interpretovat. První *chunked* číslo se nachází na prvním řádku těla odpovědi. Toto číslo udává kolik *Bytů* je mezi tímto a dalším *chunkem*. Aplikace tento problém řeší pomocí funkce *read_chunk*.

5.3.1 Aplikační realizace

Funkce *read_chunk* dostává na vstup ukazatel na první chunk. Pomocí funkce *strtol* je převedena hodnota do číselné proměnné. Pomocí *ukazatelové aritmetiky* je do paměti překopírována zpráva bez režijních dat. Následně je přeskočeno na další *chunk* a akce se opakuje do přečtení celé zprávy.

6 Implementační detaily

V této kapitole budou popsány zajímavé pasáže implementace. Většinu zdrojových souborů provází proměnná *debug* datového typu *bool*. Nastavení této proměnné do hodnoty *true*, bude program vypisovat hlavičku požadavku, hlavičku odpovědi a následně vypíše celý paket na *stdout*. Po těchto debugovacích informacích je vypsána normální reakce programu.

6.1 HTTP modul

Zpracování odpovědi předchází fáze, kdy se všechny pakety načtou po jednom do *bufferu*. Z tohoto bufferu jsou ukládány do jednoho řetězce. Řetězci je následně zpracována hlavička, pokud zkontrolování hlavičky proběhne v pořádku je provedeno odstranění kódování, pokud je nějaké nastaveno. Následně je odpověď již bez hlavičky poslána dalšímu modulu na zpracování *xml* dat.

V případě *HTTPS* komunikace a volby přepínače *-C* je nutné mít vytvořené *hashe* daného certifikátu například příkazem *c_rehash* v bash terminálu. Ověření certifikátů probíhá pomocí knihovny *openssl* a za použití funkce *SSL_CTX_load_verify_locations*. [15]

Při zvolení kombinace argumentů *-C* a *-c* a pří korektním spuštění programu je nejprve ověřován certifikát získaný pomocí argumentu *-c*, následně je zkoušeno ověřit certifikát pomocí certifikátů ve složce zadané argumentem *-C*. Při alespoň jednom ověření je server ověřený a pokračuje se v komunikaci.

6.1.1 Datové struktury

Při získávání argumentů od uživatele jsou přepínače uloženy do datové struktury *flags_t*. Struktura poskytuje lepší manipulaci a přenositelnost dat.

```
typedef struct {
bool T_flag;
bool a_flag;
bool u_flag;
}flags_t;
```

6.2 XML modul

XML parser si uloží celou strukturu a program dále k jednotlivým prvkům přistupuje pomocí stromevé struktury xml dokumentu. Pomocí kořenových prvků je rozeznána jednotlivá verze xml. Na konkrétní verzi je zavolána příslušná funkce z výběru: *parse_atom_item*, *parse_rss_item*, *parse_rss2_item*. Pokud není rozeznám protokol je volána funkce *parse_item* pro zjištění informací.

Doplňující informace jsou vypisovány vždy ve stejném pořadí. Pokud uživatel danou položku nevyžaduje je pouze přeskočena a není zobrazena. Pořadí pro výpis je následující: asociovaného *URL*, jméno *autora* následně jeho *e-mail* a poslední vypisovanou informací je poslední časová aktualizace.

Pokud zdroj neobsahuje svůj název je vypsán místo názvu text `*** feed name does not set ***`. Stejným postupem se program zachová pokud neexistuje název článku v daném zdroji. Při neexistenci názvu článku je vypsán text `title name have not set`.

6.2.1 Datové struktury

Pokud je hledaná informace v daném článku duplikována je brána *poslední* hodnota jako nejaktuálnější, a proto bude uživateli zobrazena. Samotný sběr informací je prováděn nezávisle na vstupních požadavcích uživatele.

Program se snaží na dané specifikaci xml sbírat co nejvíce dat. A následně v poslední fázi při provádění výpisu uživateli jsou zohledněny jeho požadavky a logickým součinem je provedeno vypsání jen některých informací, viz funkce *cat*. Libxml knihovna používá vlastní předdefinované datové typy jako např *xmlChar**

```
typedef struct{
xmlChar* title;
xmlChar* link;
xmlChar* author;
xmlChar* date;
xmlChar* email;
xmlChar* sy_period;
bool sy;
}item_t;
```

7 Návod na spuštění

V první fázi je nutné přeložit projekt na konkrétní architektuře. K tomuto úkolu mimo jiné slouží soubor *Makefile*.

Příkaz `#make` provede spuštění překladu, překládá zdrojové souboru ve složce *source* do složky *bin* výsledný spustitelný soubor *feedfile* se nachází v kořenovém adresáři.

Příkaz `#make clean` provede smazání přeložených souborů, tedy provede smazání souborů ve složce *bin* a souboru *feedfile* v kořenovém adresáři.

Příkaz `#make test` provede spuštění automatických testů pomocí souboru *test.sh* a složky *feeds*, která obsahuje soubory potřebné k automatickým testům, viz níže.

7.1 Adresářová struktura

Projekt je rozčleněn na několik podadresářů z důvodu přehlednosti a lepší správy souborů. V této podkapitole je nastíněno rozmístěná souborů a význam jednotlivých složek.

```
feeds/
Makefile
manual.pdf
readme
source/
test.sh
```

feeds/ - Obsahuje soubory potřebné ke spouštění testů přes skript *test.sh*.
Makefile - Slouží k automatickému překladu zdrojových souborů.
manual.pfd - Tento soubor, který popisu celý projekt.
readme - Základní popis programu a jeho omezení.
source/ - Složka se zdrojovými soubory aplikace.
test.sh - Skript pro automatické testování.

7.2 Spuštění aplikace

Aplikaci *feedreader* lze spustit s následujícími parametry.
`feedreader <URL | -f <feedfile>> [-c <certfile>] [-C <certaddr>] [-T] [-a] [-u]`

Je vyžadován právě jeden vstupní zdroj, buď formou URL adresy dokumentu nebo s parametrem *-f*, který očekává textový soubor. Soubor obsahuje na začátku řádku jednotlivé URL adresy. Může obsahovat libovolné množství prázdných řádků. Komentář je definován řádkově a to znakem *#*.

Volitelný parametr -c specifikuje soubor obsahující certifikát, pomocí kterého se bude autentizovat server. Pokud není zadán parametr *-c* nebo *-C* jsou použity výchozí certifikáty pomocí funkce `SSL_CTX_set_default_verify_paths`.

Volitelný parametr -C specifikuje složku obsahující certifikáty, pomocí kterých se bude autentizovat server. Pokud není zadán parametr *-c* nebo *-C* jsou použity výchozí certifikáty pomocí funkce `SSL_CTX_set_default_verify_paths`.

Pokud jsou zadány parametry *-c* a *-C* současně, zkouší se server ověřit nejdříve pomocí certifikátu zadaného v parametru *-c* následně v případu neúspěchu je provedeno ověření pomocí zadané složky.

Volitelný parametr -T spouští doplňující informace o poslední časové aktualizaci záznamu. Na nový řádek je vypsán text `Aktualizace:` s příslušným časovým razítkem. V případě zadání času pomocí `sy:updateBase` v rozšíření RSS1.0 a následně specifikování kroku datovou strukturou `sy:updatePeriod`, je vypsán text `Periodická aktualizace od:` časové razítko s krokem: časová jednotka.

Volitelný parametr -a spouští doplňující informace o autorovy článku. Na nový řádek je vypsán text `Autor:`, za kterým následuje autor článku. V případě *feed* souboru ve formátu *Atom* je možné definovat *email* autora. V případě zadání e-mailu se vypisuje `Email:` s hodnotou získanou v prvku. E-mail se vypíše hned po jméno autora.

Volitelný parametr -u udává vypisování URL odkazu, který je uveden v článku. Na nový řádek je vypsán text `URL:`, za kterým následuje URL odkaz přiložený v xml konstrukci.

V případě zadání nevhodných kombinací parametru, a to hlavně neuvedení zdroje nebo definice pomocí URL a zároveň pomocí parametru -f, je spuštěna *nápověda* a poté program korektně ukončen.

7.3 Testovací skript

V kořenovém adresáři projektu je obsažen testovací skript *test.sh*, který obsahuje dvě desítky základních testů. Testovací skript lze pustit po přeložení aplikace pomocí příkazu *#make test*.

7.4 Příklady spuštění

Formát výstupu v případě doplňovacích informací

```
*** <feed_source>***
<Title>
URL: <url>
Autor: <author>
Email: <email>
Aktualizace: <time>

<Title>
URL: <url>
...
```

Formát výstupu bez doplňovacích informací

```
*** <feed_source>***
<Title>
<Title>
<Title>
<Title>
...
```

7.4.1 Ručním spuštění

HTTPS komunikace, pouze názvy článků

```
./feedreader https://tools.ietf.org/dailydose/dailydose_atom.xml
```

Výstup

**** The Daily Dose of IETF ****
The Daily Dose of IETF - Issue 3225 - 2018-11-02
The Daily Dose of IETF - Issue 3224 - 2018-11-01
The Daily Dose of IETF - Issue 3223 - 2018-10-31
...

HTTPS komunikace s parametry -u -T

```
./feedreader https://tools.ietf.org/dailydose/dailydose_atom.xml -a -u -T
```

Výstup

**** The Daily Dose of IETF ****
The Daily Dose of IETF - Issue 3225 - 2018-11-02
URL: https://tools.ietf.org/dailydose/3225.html
Aktualizace: 2018-11-02T05:00:17Z

https://www.overleaf.com/project/5bd1e2a96ab1976e02d85758 *The Daily Dose of IETF - Issue 3224 - 2018-11-01*
URL: https://tools.ietf.org/dailydose/3224.html
Aktualizace: 2018-11-01T05:00:14Z
...

Neexistující URL

```
./feedreader https://this.adrress.not.exists.atom.xml
```

Výstup

Error attempting to connect

7.4.2 Pomocí testovacího skriptu

Formát výstupu automatického testu

```
INSTANCE <number>
NOTE: popis daného testu
<spuštěný výraz>

<feedreader odpověď>
```

spuštění automatického skriptu

```
#make test
```

Výstup

...
INSTANCE 8
Note: test http and https comunication with default certificates
aplication must find atom information from file of feeds
./feedreader -f ./feeds/feed_atoms

**** Wikipedia - Recent changes [en] ****
Nicola Correia-Damude
...

INSTANCE 12
Note: test corect instance with specific https port explicit set
aplication must successly connet with server
./feedreader https://xkcd.com:443/atom.xml

**** xkcd.com ****
Challengers
Ballot Selfies
Who Sends the First Text?
I'm a Car
...

Reference

[1] Popis RFC 1945 standartdu pro HTTP1.0 , https://www.ietf.org/rfc/rfc1945.txt

[2] Popis RFC 2616 standartu pro HTTP1.1, https://tools.ietf.org/html/rfc2616

[3] Popis RFC 2045 standartu pro MIME, https://www.ietf.org/rfc/rfc2045.txt

[4] Popis RFC 6101 standartu pro SSL, https://tools.ietf.org/html/rfc6101

[5] Popis RFC 8446 standartu pro TLS, https://tools.ietf.org/html/rfc8446

[6] Popis metod HTTP a URI, https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html

[7] Popis RFC 5280 standartu pro X.509, https://tools.ietf.org/html/rfc5280

[8] Popis RFC 8017 standartu pro RSA, https://tools.ietf.org/html/rfc8017

[9] Popis RFC 7919 standartu Diffie-Hellman, https://tools.ietf.org/html/rfc7919

[10] Popis RFC 2631 standartu algoritmu Diffie-Hellman, https://www.ietf.org/rfc/rfc2631.txt

[11] Předloha pro obrázky,https://www.mnot.net/rss/tutorial/

[12] Popis RFC 2045 standartu MIME ,https://tools.ietf.org/html/rfc2045

[13] Popis RFC 2046 standartu MIME ,https://tools.ietf.org/html/rfc2046

[14] BIO soket OPENSSL ,https://www.openssl.org/docs/man1.0.2/crypto/BIO_s_socket.html

[15] Důsledný popis funkce pro ověřování ceritifkátů.
https://www.openssl.org/docs/man1.0.2/ssl/SSL_CTX_load_verify_locations.html