VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Konfigurace a správa sítí

Čtečka novinek ve formátu Atom s podporou TLS

22.10.2018

Patrik Holop

Obsah

1 Úvod

Web Feed je pojem označujúci dátový formát používaný na zverejňovanie informácií, ktoré sa často aktualizujú, napríklad spravodajské portály alebo blogy. Zhromažďovanie takýchto kolekcií sa nazýva agregácia. Web Feed umožňuje softvérovým programom sledovať zmeny publikované na stránke. Stránka aktualizuje zoznam novopublikovaných článkov v špecifickom formáte, ktorý sa po stiahnutí a spracovaní programom môže využiť, prípadne umožnuje užívateľom sa prihlásiť na odoberanie obsahu.

Medzi najznámejšie formáty takýchto zoznamov patria *Atom* a *RSS*. Napriek tomu, že formát RSS vznikol ako prvý z nich, štandartizoval niekoľko verzií, ktoré sa významne líšia. Cieľom tohto projektu bolo naštudovať si informácie o daných formátoch a vytvoriť program, ktorý bude schopný spracovať takéto zdroje a prezentovať ich užívateľovi.

2 Teoretický podklad

Atom aj RSS štandarty vychádzajú z formátu XML, čo je značkovací jazyk definujúci sadu pravidiel na zakódovanie dokumentov, preto sú často súbory zakončované okrem prípon *.atom* a *.rss* i príponou *.xml*.

2.1 Formát Atom

The Atom Syndication Format je definovaný pomocou *RFC4287*. Formát Atom definuje zoznamy združených informácií ako *Feeds* a jednotlivé položky v rámci zdroja sú nazývané *entries*. Každá položka môže byť rozšírená o doplňujúce informácie (metadáta).

Atom definuje 2 typy dokumentov, *Atom Feed Document* a *Atom Entry Document*. Rozdeľujú sa v tom, že Feed Document definuje zdroj obsahujúci viaceré položky a Entry Document obsahuje informácie o jedinej položke. Pretože tento projekt slúži na spracovanie zdrojov, spracovanie Entry Document nie je podporované.

2.1.1 Atom Feed Document

Každý dokument musí byť *well-formed XML*, ale deklarácia XML v hlavičke dokumentu je nepovinná, takže s tým musí program počítať. Koreňový element daného dokumentu musí byť *atom:feed*. Tento typ elementu reprezentuje celý zdroj a slúži ako obálka pre zoznam *atom:entry* elementov, ktoré popisujú jednotlivé články. Nemusí obsahovať žiaden takýto element.

Medzi najdôležitejšie vlastnosti *atom:feed* elementu patria:
- *title* reprezentuje názov
- *updated* - reprezentuje čas a dátum poslednej aktualizácie zdroja
- *author* - reprezentuje autora

Zdroj má obsahovať práve jeden názov a čas zmeny, ale k autorovi sa viaže viac podmienok.

Na reprezentovanie autora a obecne osôb definuje Atom vlastnú štruktúru *atomPersonConstruct*. Každá osoba musí mať zadané práve jedno meno a prípadne mail a URI spojované s danou osobou. V prípade elementu *updated* je dátum a čas vo formáte ISO 8601. Položka môže obsahovať aj element *published*, reprezentujúci publikovania danej položky.

2.1.2 Atom Entries

Každá položka je definovaná svojim názvom (element *title*) a dobou aktualizácie (element *updated*), ktoré majú rovnaký tvar ako v prípade Atom Feed Document, sú povinné a musia byť obsiahnuté práve raz.

Položka môže obsahovať jedného alebo viacerých autorov. V prípade, že tomu tak nie je, môže obsahovať zdroj (element *source*), ktorý má podobne ako osoby vlastnú štruktúru a môže obsahovať autorov. Až v prípade,

kedy by ani jedna z týchto štruktúr nemala definovaných autorov sa považuje za autora položky autor zdroja definovaný v koreňovom *atom:feed* elemente. Toto poradie je definované v RFC a každý program, ktorý dané dokumenty spracováva, ho musí rešpektovať.

Odkazy (element *atom:link*) vo formáte Atom sú uložené ako atribúty daných elementov. Rozoznáva sa niekoľko typov takýchto odkazov, pričom typ odkazu značí hodnota atribútu *rel* a hodnota odkazu je v atribúte *href*. V prípade odkazov priamo na položky sú dôležité typy *self*, ktorý značí odkaz priamo na danú položku a *alternate*, ktorý označuje alternatívny zdroj informácií o danej položky. V prípade, že atribút *rel* nie je určený, považuje sa daný odkaz za typ *alternate*. Pretože formát Atom umožňuje aj odkazy typu *sftp* a podobne, v programe je overované, či sa naozaj jedná o platnú adresu URL.

2.2 Formát RSS

Formát RSS (Really Simple Syndication) je naopak od formátu Atom zameraný na jednoduchosť s možnosťou jednoduchého rozšírenia. Bol navrhnutý ako formát popisujúci kanál obsahujúci jednotlivé položky. Jednotlivé elementy ako autor alebo dátum nereprezentujú zložené štruktúry ako pri formáte Atom, ale informácie sú uložené ako hodnota elementu. Odkazy pre formát RSS by mali obsahovať URL, ale niektoré zdroje toto pravidlo porušili, takže sa v programe overuje, či sa jedná o platnú URL adresu.

2.2.1 RSS 1.0

Podobne ako najstaršia verzia formátu RSS 0.9, ani verzia 1.0 sa v súčasnosti bežne nepoužíva, ale neustále ho niekoľko spravodajských portálov používa, preto daný projekt túto verziu podporuje. Formát RSS 1.0 bol navrhnutý spolu s XML vo formáte RDF (Thre Resource Description Framework), ktorý upravuje syntax XML pre popis zdrojov. RSS je 1.0 je spätne kompatibilný s verziou RSS 0.9*.

Rovnako ako pri formáte Atom, ani v tomto formáte nie je deklarácia XML povinná, odporúča sa len pre spätnú kompatibilitu. Koreňovým elementom RSS 1.0 dokumentu je RDF element v tvare *rdf:RDF*.

Koreňový element obsahuje povinný element *channel*, ktorý reprezentuje popisovaný kanál. Kanál musí mat zadaný reprezentujúci názov a ďalšie popisujúce metadáta o kanále, medzi najdôležitejšie patrí RDF tabuľka obsahu (element items), ktorý reprezentuje zoznam všetkých položiek popisovaných v danom dokumente.

Ďalej môže koreňový element obsahovať aj ďalšie záznamy súvisiace v HTML vykresľovaním a štruktúrou daného kanálu, napr. obrázky alebo textové vstupy, ktoré nás v rámci tohoto projektu nezaujímavú.

Jednotlivé položky vo formáte RSS sú nazývané *item*. Každá položka má svoj názov (element *title*), môže byť rozšírená o URL odkaz (element *link*) a iné. Formát RSS 1.0 priamo nedefinuje formát autora ani formát času poslednej aktualizácie, viac v sekcii Rozšírenia.

2.2.2 RSS 2.0

Táto verzia už nevychádza z RDF, ale jednotlivé XML elementy sú adresované priamo. Vlastnosti má podobné ako formát RSS 1.0, upravuje ale podstatné vlasnosti dokumentu.

Koreňový element je definovaný ako *rss* s atribútom *version="2.0"* Element *channel* obsahuje metadáta, ale namiesto zoznamu položiek a ich dodatočnej definícii v rámci koreňového elementu obsahuje ich definície priamo. Taktiež môžu položky obsahovať informácie o autorovi pomocou elementu *author*. Autor by mal byť podľa špecifikácie reprezentovaný jeho emailovou adresou. Pridal sa aj element *pubDate*, ktorý upravuje formát času a dátumu publikovania danej položky. Elementy *author* aj *pubDate* sú voliteľné.

2.3 Moduly a rozšírenia

Formát RSS bol navrhnutý na jednoduché rozširovanie pomocou modulov. Najznámejší z nich je *Dublin Core*. Dublic Core upravuje menný priestor tak, že umožnuje pridať nové elementy s predponou *dc:*. Elementy, ktoré ovplyvňujú tento projekt, sú alternatívne značenie pre autora a dátum. Dublic Core je možné použiť aj ako rozšírenie pre formát Atom.

Element *dc:creator* je navrhnutý ako alternatíva pre RSS element *author*, pokiaľ nemá byť zverejnená autorova e-mailová adresa ale napríklad uživateľské meno. *Date* sa používa ako alternatíva k RSS elementu *pubDate*.

2.4 Spojenie

V rámci tohto projektu sa získavajú zdrojové dokumenty pomocou aplikačného protokolu HTTP. Užívateľ zadá URL daného zdroja a pomocou HTTP GET požiadavku sa získa zdrojový súbor ako obsah odpovede.

Pre tento účel sa pracuje s protokolom HTTP/1.1. Táto verzia HTTP protokolu je podporovaná na väčšine serverov a zároveň priniesla rýchlejšie spracovanie požiadavkov. Ďalší dôvod, prečo sa nepoužíva staršia verzia HTTP/1.0 je ten, že z hľadiska budúcnosti sa môže stať zavrhnutou skôr ako verzia HTTP/1.1 a mnoho webových nástrojov používa ako východzí protokol HTTP/1.1.

Pri HTTP/1.1 je potrebné riešiť možnosť rozdelenia správy na menšie celky (*chunk*). To, či daná HTTP odpoveď využíva rozdelovanie na menšie celky, sa dá zistiť z HTTP hlavičky. V takom prípade je treba zo správy odstrániť informácie o veľkostiach nasledujúcich celkov.

2.4.1 HTTP

Na vytvorenie správneho požiadavku je nutné vytvoriť správnu HTTP požiadavku. Príklad HTTP požiadavky používanej v tomto projekte:

```
GET /news/news-rss.php HTTP/1.1\r\n
Host: www.fit.vutbr.cz\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 ...
Connection: Close\r\n\r\n
```

User-Agent je charakteristický reťazec, ktorý umožnuje identifikovať softvér a systém, ktorý požiadavku vytvoriť. Je nutné ho vložiť pre prístup k niektorých zdrojov. User-Agent používaný v tomto projekte je skopírovaný z prehliadača Mozzila Firefox.

Východzí port pre HTTP komunikáciu je 80.

2.4.2 HTTPS a TLS

Protokol HTTPS umožňuje šifrované spojenie HTTP komunikácie pomocou vytvorenia SSL/TLS vrstvy, ktorá umožňuje danú komunikáciu šifrovať.

Prvá fáza vytvorenia bezpečného spojenia sa nazýva *SSL handshake*. Klient pošle požiadavku serveru, kde požaduje bezpečné pripojenie a priloží verziu SSL/TLS, ktorá by sa mala pri komunikácii používať, spolu s typom šifrovacieho a kompresného algoritmu. Server odpovedá potvrdením požiadkov klienta a posiela klientovi certifikát. Certifikát slúži na overenie totožnosti serveru a obsahuje jeho verejný kľúč. Server má právo po klientovi požadovať jeho overenie pomocou klientskeho certifikátu, ale sa to stáva vo výnimočných prípadoch.

Druhá fáza spočíva vo výmene kľúčov. Klient vygeneruje tzv. Pre-Master Key a zašifruje ho pomocou verejného kľúča poskytnutého serverom a pošle ho serveru. V tomto kroku sa využíva asymetrické šifrovanie. Server daný kľúč dešifruje pomocou svojho privátneho kľúča. V predchodzích krokov si server a klient zároveň vymenili náhodnú postupnosť znakov, ktoré poslúžia spolu s Pre-Master Key na vygenerovanie nového kľúča

nazývaného Master-Secret key. Túto hodnotu sú schopný klient aj server vygenerovať samostatne a bude slúžiť ako symetrický kľúč počas celého spojenia.

V poslednom kroku obe strany vegenerujú Change Cipher Spec správu, ktorá slúži na zmenu aktuálne používaneho šifrovania v rámci komunikácie. Po tomto kroku je celá komunikácia šifrovaná.

Keď server posiela certifikát klientovi v prvej fázi, klient ho overí porovnaním voči dôveryhodným certifikátom. Užívateľ má právo zadať úložisko takýchto certifikátov prípadne priamo certifikačný súbor.

Východzí port pre HTTPS komunikáciu je 443.

3 Implementácia

Tento projekt je implementovaný v jazyku C++. Využíva objektovo orientovaný návrh a je modulárny. Implementácia zachováva všetky vlastnosti formátov a spojení popísaných v teoretickej časti okrem kontroly povinných prvkov, aby bola podľa zadania robustná.

3.1 Spracovanie parametrov

Pri spracovaní parametrov program sa využíva funkciu *getopt*. Každému možnému parametru je pridelená značka (flag typu bool), ktorý značí, či už bol parameter zadaný a umožňuje kontrolovať duplicitu. Je kontrolované, či užívateľ zadal feedfile alebo URL zdroja, a chyba nastane ak nebola zadaná žiadna z nich, prípadne obe. Kontroluje sa aj či užívateľ nezadal neplatný argument. Validita hodnôt argumentov sa kontroluje až pri pokuse o ich spracovanie. Je možné kombinovať parametre *-c* a *-C*. Hodnoty zadaných argumentov sa ukladajú do premenných typu string a posielajú na spracovanie.

Funkcia *getopt* podporuje zadávanie parametrov v kombinovanej forme, napr. *-Tua*. Program podporuje vypísanie nápovedy pomocou parametru *-h*. V prípade zadania parametru *-h* sa všetky ostatné zadané parametre ignorujú.

3.2 Spojenie pomocou OpenSSL

Spojenie a sieťová komunikácia využíva knižnicu OpenSSL a implementácia využíva postup a funkcie zverejnený na oficiálnej stránke [1]. Knižnica ponúka rozhranie BIO, ktoré slúži ako abstrakcia nad socketami a ich príslušnými funkciami. V tomto projekte abstrahuje prácu so spojením trieda *ConnectionClient*, ktorá ponúka metódy na získanie obsahu dokumentu zo zadaného URL zdroja.

Daná knižnica sa používa pri implementovaní bezpečnej i nezabezpečenej komunikácii projektu a je definovaná v súbore *client.h*.

Najprv sa skontroluje validita zadaného URL pomocou regulárneho výrazu, ktorý extrahuje *hostname* a *port*. Ak port nebol zadaný, nastaví sa ako východzia hodnota port 80 pre HTTP a 443 pre HTTPS. Vytvorí sa odkaz na štruktúru *BIO*. Pomocou funkcie *BIO_new_connect* sa vytvorí spojenie a *BIO_do_connect* spojenie skontroluje.

Po vytvorení a overení spojenia musí klient poslať HTTP GET požiadavku, ktorú zostaví podľa popisu v teoretickej časti. Tá sa pomocou funkcie *BIO_write* pošle. Potom stačí v cykle načítať celú odpoveď HTTP serveru a uložiť ju v pamäti.

V prípade zabezpečenej komunikácie sa v prípade, že užívateľ nezadal parameter *-c* ani *-C* použije funkcia *SSL_CTX_set_default_verify_paths*, ktorá nastaví úložisko certifikátov na východziu hodnotu. Implementácia tohto projektu predpokladá používanie zložiek, ktoré boli hashované pomocou príkazu c_rehash. V opačnom prípade sa zadané hodnoty odovzdajú funkcii *SSL_CTX_load_verify_locations*, ktorá sa pokúsi najprv aplikovať certifikačný súbor a následne zložku. Pred zahájením komunikácie je treba overiť certifikát pomocou funkcie *SSL_get_verify_result*.

[1] https://developer.ibm.com/tutorials/l-openssl/

Ako bolo popísané v teoretickej časti, tento program pracuje s protokolom HTTP/1.1, takže je nutné skontrolovať správu, či neobsahuje rozde1ovanie na chunky. Oddelí sa hlavička správy a jej obsah. Ak rozdelovanie obsahuje, načíta sa hexadecimálne číslo N, ktorým obsah začíta a po N znakoch sa vyskytne daľšia číselná značka, ktorú je nutné odstrániť. Takto sa postupne odstraňujú (spolu s okolitými znakmi ukončenia riadku) až pokým sa nedostane k číslu 0, čo značí koniec správy. Takto spracovaný obsah správy reprezentuje XML dokument.

3.3 Spracovanie XML

Na spracovanie získaných dát (reťazec obsahujúci XML dokument) sa používa knižnica *libxml2* [2].

Tá nám umožnuje pomocou funkcie *xmlParseMemory* načítať XML dokument z reťazca do štruktúry abstrahovanej danou knižnicou a zároveň skontrolovať platnosť formátu XML dokumentu. Ďalej sa pracuje s odkazom na koreňový element typu *xmlNode*. Knižnica poskytuje funkcie na získanie atribútov a každý element obsahuje odkaz na jeho podelementy a súrodencov.

Trieda *ConnectionClient* poskytuje funkcie na spracovanie celého súboru obsahujúceho zdroje podľa pravidiel v teoretickej časti, stiahnutie obsahu jednotlivých zdrojov a extrahovanie informácií. Najprv sa získa podľa koreňového elementu informácia, s akým typom dokumentu sa pracuje. Podľa toho sa zavolajú funkcie pre spracovanie RSSv1, RSSv2 a Atom dokumentov. Cieľom funkcií je naplniť dáta tried *AtomEntryInformation RssItemInformation* definované v súbore *information.h*. Tie obsahujú atribúty pre všetky informácie, ktoré je možné pre jednotlivé položky vyextrahovať. Ak je hodnota daného atribútu prázdny reťazec, predpokladá sa, že sa v položke / dokumente nenašiel. Triedy *AtomItemInformationSummary* a *RssItemInformationSummary* obsahujú informácie o všetkých položkách zo súboru, atribúty pre globálne informácie (napr. autor celého zdroja) a im definované metódy *print_info* obsahujú pravidlá, v akom poradí a ktoré informácie vypisovať podľa špecifikácie.

3.4 Prezentácia výsledkov

V súbore *information.cpp* je definované poradie a pravidlá výpisu informácií podľa teoretickej časti. V prípade, že je obsah elementu *title* určitej položky prázdne, vypíše sa reťazec `Empty title`. Autor, URL a dátum a čas sa vypíšu jedine v prípade, že dané informácie bolo možné získať.

Dodatočné informácie sa vždy vypisujú v poradí:
1. autor
2. URL
3. dátum a čas

Ak daný dokument využíva rozšírenie *Dublin Core*, informácie *dc:creator* a *dc:date* danej položiek *item* alebo *entry* majú prednosť pred globálnymi informáciami o celom zdroji, prípadne kanála, ako sú napríklad autor celého zdroja.

4 Testovanie

Na testovanie je využitý script *test.sh* v zložke *tests*, ktorý testuje nasledovné oblasti:
- spracovanie parametrov
- vytvorenie spojenia a stiahnutie dokumentu
- spracovanie formátu a výstup

V rámci spracovania parametrov sa testujú platné i neplatné parametre, ich kombinácie a ich hodnoty.

[2] http://www.xmlsoft.org/html/index.html

Vytvorenie spojenia sa testuje na nasledujúcich zdrojoch a očakáva sa, že budú pri testovaní dostupné. Testuje sa vytvorenie spojenia pri použití východzích i špecifikovaných portov, overenie pri použití neplatných certifikátov a podobne.
- http://www.fit.vutbr.cz/news/news-rss.php
- https://tools.ietf.org/dailydose/dailydose_atom.xml

Pretože by sa mohol obsah zdrojov časom meniť a nebolo by možné správne spracovanie testovať, na tento účel sú v archíve uložené zdroje a ich manuálne spracované výsledky. Program podporuje prepínač *-p*, ktorý umožnuje načítať obsah XML dokumentu zo štandartného vstupu bez predchádzajúceho HTTP požiadavku. Stiahnuté testovacie zdroje boli vybrané a upravené tak, aby obsahovali čo najviac možností formátu položiek vo formátoch Atom a RSS. Zdroj formátu Atom napríklad obsahuje položky s viacerými autormi, s autorom zadaným v elemente *source*, bez autora, aby sa musel použiť autor celého zdroja a podobne. Tieto zdroje sa spracujú programom a výstup sa porovná pomocou programu `diff` s uloženým očakávaným výstupom.

Na konci testovania sa vypíše krátka štatistika o testoch.

Testy je možné spustiť príkazom *make test* v koreňovej zložke archívu. Testovanie vykonajte až po preklade programu pomocou príkazu *make*.

Pri správnom postupe by mal byť výstup testov nasledovný:

```
eva ~/Dokumenty> make test
sh ./tests/test.sh

*** Test info ***
You can find input files in tests/input/
You can find manually parsed results in tests/results/

*** Running tests ***

 ** Parameters **
  - No parameters (OK - finished with error)
  - Printing help (OK - finished successfully)
  - Passing doubled parameter (OK - finished with error)
  - Combining feedfile and URL (OK - finished with error)
  - Not valid parameter / more URLs (OK - finished with error)

 ** Connection (downloading document)**
  - Testing HTTP with default port (http://www.fit.vutbr.cz/news/news-rss.php) (OK - finished successfully)
  - Testing HTTP with custom port (http://www.fit.vutbr.cz:80/news/news-rss.php) (OK - finished successfully)
  - Testing HTTP : not existing URL (http://www.fit.vutbr.cz/NOT_VALID) (OK - finished with error)
  - Testing HTTPS with default port (https://tools.ietf.org/dailydose/dailydose_atom.xml) (OK - finished successfully)
  - Testing HTTPS with custom port (https://tools.ietf.org:443/dailydose/dailydose_atom.xml) (OK - finished successfully)
  - Testing HTTPS : not existing URL (https://tools.ietf.org/dailydose/NOT_VALID) (OK - finished with error)
  - Testing HTTPS with /dev/null as certaddr (https://tools.ietf.org/dailydose/dailydose_atom.xml) (OK - finished with error)
  - Testing not valid URL (abc.ietf.orga) (OK - finished with error)

 ** Parsing ATOM **
  - Parsing corrupted XML content (tests/input/atom/corrupted.xml) (OK - finished with error)
  - Parsing valid XML content : no -Tua params (tests/input/atom/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Author only (tests/input/atom/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Link only (tests/input/atom/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Date only (tests/input/atom/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : All params -Tua (tests/input/atom/valid_1.xml) (OK - finished successfully)

 ** Parsing RSSv1 **
  - Parsing corrupted XML content (tests/input/rss_v1/corrupted.xml) (OK - finished with error)
  - Parsing valid XML content : no -Tua params (tests/input/rss_v1/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Author only (tests/input/rss_v1/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Link only (tests/input/rss_v1/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Date only (tests/input/rss_v1/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : All params -Tua (tests/input/rss_v1/valid_1.xml) (OK - finished successfully)

 ** Parsing RSSv2 **
  - Parsing corrupted XML content (tests/input/rss_v2/corrupted.xml) (OK - finished with error)
  - Parsing valid XML content : no -Tua params (tests/input/rss_v2/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Author only (tests/input/rss_v2/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Link only (tests/input/rss_v2/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : Date only (tests/input/rss_v2/valid_1.xml) (OK - finished successfully)
  - Parsing valid XML content : All params -Tua (tests/input/rss_v2/valid_1.xml) (OK - finished successfully)

*** Tests finished ***
Passed: 31
Failed: 0
```

5 Použité zdroje

NOTTINGHAM M., SAYRE R. *The Atom Syndication Format* [online]. Release Date: December, 2005. Dostupné z: https://tools.ietf.org/html/rfc4287

BRICLEY D., DAVIS Ian et al. *RDF Site Summary (RSS) 1.0* [online] Dostupné z: http://web.resource.org/rss/1.0/spec

RSS 2.0 Specification [online]. Marec, 2009. Dostupné z: http://www.rssboard.org/rss-specification

MATOUSEK P. *Síťové služby a jejich architektura*. Brno, 2014. Publishing house of Brno University of Technology VUTIUM. ISBN 978-80-214-3766-1.

BALLARD K., *Secure programming with the OpenSSL API* [online]. November, 2018. Dostupné z: https://developer.ibm.com/tutorials/l-openssl/