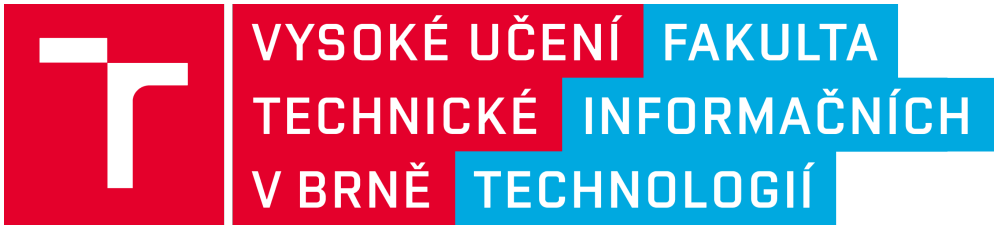

Dokumentace projektu ISA

Čtečka novinek ve formátu Atom a RSS s podporou TLS

6. listopadu 2022 Ladislav Vašina (xvasin11)

Obsah

1 Úvod a popis programu

Cílem tohoto projektu do předmětu `Síťové aplikace a správa sítí` bylo vytvořit program v jazyce C nebo C++ umožňující uživateli zobrazit feed novinek ve formátu RSS 2.0 [7] nebo ATOM [10], ze zadané URL (či souboru obsahující více URL) s podporou TLS (Transport Layer Security) [5]. Program umožňuje uživateli různé druhy výpisu novinek viz 2.

Při navazování zabezpečeného spojení se serverem, se využívá SSL protokolu resp. vrstvy, která je vložená mezi transportní vrstvu (např. TCP/IP) a aplikační vrstvu (HTTP) [4]. Pokud tedy chceme přistoupit k serveru, který má obsahovat RSS/ATOM feed je potřeba ověřit platnost certifikátů. Digitální certifikáty (veřejně podepsané šifrovací klíče) vydávají tzv. certifikační autority (CA), které využívají principu PKI (Public Key Infrastructure) a svoji autoritou potvrzují korektnost údajů, které jsou uvedeny ve veřejném klíči. Certifikáty nám tedy zaručují identifikaci protistrany při navazování spojení. Pro přenos certifikátů se využívá binární podoba DER nebo textový formát PEM (zároveň formát podporovaný funkcemi knihovny openSSL) či CER [3]. Certifikáty mají omezenou platnost, kterou určuje jeho majitel a certifikační autorita.

2 Použití programu

Pokud chce uživatel spustit program, tak ho musí nejdříve přeložit. Přeložení aplikace vytvoří spustitelný soubor `feedreader`.
Samotný překlad se provede následovně:

```
$ make
```

Po úspěšném překladu může uživatel program spouštět následujícím způsobem.
`./feedreader <URL | -f <feedFile>> [-c <certFile>] [-C <certDir>][-T] [-a] [-u]`

- `URL` Místo `URL` uživatel zadá url zdroje, ze kterého chce získávat feed novinek.
- `-f <feedfile>` `feedfile` je soubor obsahující url, ze kterých chceme získat feed novinek. Každá url je na samostatném řádku. Prázdné řádky či řádky začínající znakem (komentář) jsou ignorovány.
- `-c <cerfile>` `certfile` je soubor obsahující certifikát pro ověření.
- `-C <certaddr>` `certaddr` je adresář obsahující certifikáty pro ověření.
- `-T` Zobrazí pro každý záznam informaci o čase, kdy byl záznam změněn.
- `-a` Zobrazí pro každý záznam informaci o jeho autorovi.
- `-u` Zobrazí pro každý záznam informaci o jeho url.
- `-h | --help` Zobrazí nápovědu.

2.1 Ukázka použití programu

```
$ ./feedreader "https://what-if.xkcd.com/feed.atom" -u -a
*** what if? ***
Transatlantic Car Rental
Aktualizace: 2022-09-06T00:00:00Z
URL: https://what-if.xkcd.com/160/

...výstup zanedbán...
```

3 Návrh a implementace aplikace

Program `feedreader` byl vytvořen v jazyce C++. Důvodem upřednostnění jazyka C++ oproti jazyku C, byla možnost využití řetězců, což zjednodušilo část řešení jednotlivých podproblémů při implementaci.

3.1 Hlavní funkce programu `main`

Program se při spuštění této funkce nejprve dostane do části, kde kontroluje vstupní argumenty, které uživatel zadal. Kontrola probíhá cyklickým procházením vstupních argumentů a nastavováním globálních proměnných (flagů), podle kterých se dále dá v programu rozhodovat. Při nekorektním použití přepínačů programu, je program ukončen chybovou hláškou a uživateli je zobrazeno validní použití přepínačů.

Pokud si uživatel vybral možnost `-f <feedfile>` je zavolána funkce `check_feed_file_urls` 3.2, která projde a zkontroluje validitu URL v poskytnutém souboru.

Pokud možnost `-f <feedfile>` nebyla vybrána pokračuje se v kontrole URL, kterou uživatel zadal jako vstupní argument. Ke kontrole URL se využívá regulárního výrazu za použití knihovny regex [8]. Regulární výraz je rozdělen do skupin, podle kterých je následně možné získávat a ověřovat existenci různých částí URL např. http/https, explicitně zadaný port, ...

Následně je zavolána funkce `make_conn_parse` 3.3, která se stará o spojení a výpis feedů. Nakonec je provedeno vyčištění a uvolnění struktur, které využívaly knihovny `openSSL` [2] a `libxml2` [9].

3.2 Funkce `check_feed_file_urls`

Tato funkce dostane jako vstupní parametr vektor, který je naplněn jednotlivými neprázdnými (nezakomentovanými) řádky vstupního souboru. Skrze vektor se následně iteruje, a pokud je daná URL validní, je pro ni zavolána již dříve zmíněná funkce `make_conn_parse` 3.3. Pokud ovšem kontrola URL selže, je uživateli vypsána chybová hláška, že daná URL není validní a poté je pokračováno s další URL v souboru.

3.3 Funkce `make_conn_parse`

Na počátku této funkce se inicializuje `openSSL` [2] knihovna, kterou budeme využívat k navázání spojení a ověřování certifikátů. Funkce se na základě proměnné `sslFlag` rozhodne, zda začneme navazovat zabezpečené připojení či nikoliv. Jelikož byla kontrola URL implementována pomocí regulárního výrazu, který byl rozdělený do skupin, můžeme využít informace, zda uživatel zadal port

explicitně v URL či nikoliv. Tuto informaci nutně potřebujeme znát pro úspěšné navázání spojení se serverem. Pokud tedy port nebyl explicitně zadán, tak u HTTP spojení přiřadíme port 80 a u HTTPS port 443. Dále se pomocí funkce `SSL_CTX_load_verify_locations` určují cesty k certifikátům, které uživatel mohl poskytnout jako argumenty programu. Pokud uživatel nevyužije přepínače `-c` nebo `-C`, je použito výchozí cesty k certifikátům za použití funkce `SSL_CTX_set_default_verify_paths` [1]. Dále je ověřená platnost X.509 certifikátu [6] pomocí následujícího kódu:

```
if(SSL_get_verify_result(ssl) != X509_V_OK)
    exit_feed("!!!Nelze ověřit certifikát!!!");
```

Pro získání obsahu dané stránky musíme vytvořit HTTP požadavek, který zašleme oné stránce. Po tomto požadavku budeme v cyklu číst odpověď serveru pomocí funkce `BIO_read`. V dané odpovědi si pomocí funkce `find` naleznu index, kde se nachází počátek XML souboru s feedem. Hlavičku HTTP odpovědi nacházející se před tímto indexem mohu zahodit, jelikož ji pro vypsání feedů nepotřebuji. Pokud počátek XML souboru není nalezen, pak daná URL neobsahuje RSS 2.0 či ATOM feed.

Od hlavičky očištěná odpověď, je poté předána funkci `xmlParseDoc` z knihovny `libxml2`, která se pokusí rozparsovat daný feed tak, aby bylo následně možné lépe pracovat s daným XML souborem. Pokud funkce `xmlParseDoc` zahlásí jakoukoliv chybu, kvůli které není schopna daný feed rozparsovat (nevalidita XML, neukončené tagy, apod.), je program ukončen s chybou (v případě užití jedné url jako argument, ne feedfilu). Kořenový element, je následně poslán jako parametr funkci `check_feed_format` 3.4, která rozhodne, zda se jedná o feed ve formátu RSS 2.0 nebo ATOM. Dle výsledku této funkce je následně volána buď funkce `print_atom` nebo `print_rss` 3.5.

3.4 Funkce `check_feed_format`

Tato funkce jako argument přijímá xmlNode, což je kořenový uzel našeho XML dokumentu. Rekurzivně se pak skrze děti uzlů iteruje a hledá se výskyt uzlu s hodnotou „feed“ nebo „rss“, které jsou unikátní pro každý z formátů, což znamená, že následně můžeme rozhodnout o formátu feedu.

3.5 Funkce `print_atom` a `print_rss`

Princip těchto funkcí je naprosto identický. Funkci přijde jako argument kořenový uzel, přes jehož děti se začne iterovat a hledá se „hlavní“ uzel obsahující potřebné informace např. uzel „channel“ nebo „entry“, který obsahuje další uzly. Skrze ně se také iteruje, dokud se nenajdou požadované informace (autor, url, apod.) v určité úrovni stromové struktury.

Jelikož ATOM formát může mít uvedené jak jméno autora, tak jeho email jsou oba tyto záznamy vypsány s prefixem „`Autor: `“.

4 Testování

Testy pro program `feedreader` jsou napsány jako shellový skript v souboru `test.sh`.
Testy se spustí příkazem:

```
$ make test
```

Testovací matice obsahuje testy funkcí vstupních argumentů a očekávaných výstupů programu. Uživateli je po dokončení všech testů zobrazen souhrn výsledků testů.

Program byl testován lokálně na `Ubuntu 20.04.5 LTS`, `Fedora 37`, fakultním serveru Merlin (`CentOS Linux release 7.9.2009`) a Eva (`FreeBSD 13.1-STABLE`).

5 Závěr

Implementace toho projektu mě v celku bavila a dle mého názoru hlavně i obohatila. Obohatila mě o znalosti z problematiky TLS, PKI, nastudoval jsem si RSS 2.0 a ATOM formáty či korektní syntaxi URL. Dále jsem si v praxi vyzkoušel použití knihovny pro práci s regulárními výrazy, knihovny `OpenSSL` či knihovny `libxml2`. Také se mi podařilo implementovat všechny aspekty programu definované v jeho zadání.

Použitá literatura

[1] *OpenSLL - Set default verify paths* [online]. [vid. 2022-10-13]. Dostupné z: https://www.openssl.org/docs/man3.0/man3/SSL_CTX_set_default_verify_paths.html#:~:text=SSL_CTX_set_default_verify_paths()%20specifies%20that,the%20default%20store.

[2] *OpenSSL library* [online]. 1999. [vid. 2022-10-13]. Dostupné z: https://www.openssl.org/.

[3] *Digitální certifikát* [online]. San Francisco (CA): Wikimedia Foundation, 2001-. [vid. 2022-11-05]. Dostupné z: https://cs.wikipedia.org/wiki/Digit%C3%A1ln%C3%AD_certifik%C3%A1t.

[4] *Secure Sockets Layer* [online]. San Francisco (CA): Wikimedia Foundation, 2001-. [vid. 2022-11-05]. Dostupné z: https://cs.wikipedia.org/wiki/Secure_Sockets_Layer.

[5] *Transport Layer Security*. San Francisco (CA): Wikimedia Foundation, 2001-. [vid. 2022-10-13]. Dostupné z: https://en.wikipedia.org/wiki/Transport_Layer_Security.

[6] *X.509* [online]. San Francisco (CA): Wikimedia Foundation, 2001-. [vid. 2022-10-13]. Dostupné z: https://cs.wikipedia.org/wiki/X.509.

[7] *RSS 2.0 specification* [online]. Březen 2009. [vid. 2022-10-13]. Dostupné z: https://www.rssboard.org/rss-specification.

[8] *Cpp reference - regex* [online]. Duben 2011. [vid. 2022-10-13]. Dostupné z: https://en.cppreference.com/w/cpp/regex.

[9] *Libxml2* [online]. Září 1999. [vid. 2022-10-13]. Dostupné z: https://gitlab.gnome.org/GNOME/libxml2.

[10] NOTTINGHAM a SAYRE. *Atom specification* [online]. Prosinec 2005. [vid. 2022-10-13]. Dostupné z: https://www.rfc-editor.org/rfc/rfc4287.