Dokumentace ISA

Varianta 1: Čtečka novinek ve formátu Atom s podporou TLS

Autor: Tomáš Kukaň, xkukan00
17. 10. 2018

Obsah

HTTP a HTTPS

HTTP (HyperText Transfer Protocol) a HTTPS (HyperText Transfer Protocol Secure) jsou oba protokoly na aplikační vrstvě a slouží tedy k přenosu dat mezi aplikacemi. Jsou si podobné, nejdůležitější rozdíl je že data posílána pomocí HTTPS jsou oproti HTTP zašifrována za pomoci asymetrické kryptografické metody.

Samotná HTTP(S) komunikace probíhá na principu dotaz-odpověď (request-response). Klient který vyžaduje data od nějaké služby, vyšle dotaz a dotazovaný server na něj odpoví.

Každý dotaz i odpověď se vždy skládají z hlavičky a těla, kde tělo může mít i nulovou délku.

HTTP/1.0 protokol obsahuje několik metod: GET, HEAD, POST. HTTP/1.1 byla obohacena o další: PUT, DELETE, TRACE, OPTIONS, CONNECT.
GET metoda: jediná metoda kterou jsem použil v projektu sloužící k požádání o HyperTextovou stránku, RSS feedy apod.

HTTP hlavička

Hlavička se skládá z dotazu (request-line) a poté nepovinných hlavičkových položek v tomto formátu:

```
HTTP-header    = field-name ":" [ field-value ] CRLF

field-name     = token
field-value    = *( field-content | LWS )

field-content  = <the OCTETs making up the field-value
                 and consisting of either *TEXT or combinations
                 of token, tspecials, and quoted-string>
```

[1]

V hlavičkových položkách se přenáší různé metadata, informace o prohlížeči a také cookies - metadata které využívá server např. pro uložení id uživatele, stavu spojení.

SSL a TLS

SSL (Secure Sockets Layer) a TLS (Transport Layer Security) jsou kryptografické protokoly používané k bezpečné (šifrované) HTTPS komunikaci. TLS je novější verze SSL a jsou mezi nimi pouze malé rozdíly.

Inicializační protokol (handshake)

Nejdůležitější část TLS (SSL) komunikace je její počátek, tedy inicializační protokol. Při něm se server a klient dohodnou na šifrovací metodě kterou použijí, a potom se pomocí asynchronně šifrované komunikace dohodnou na klíči který se použije na synchronní šifrování. Synchronní šifrování se používá protože je o mnoho méně náročné než asynchronní.

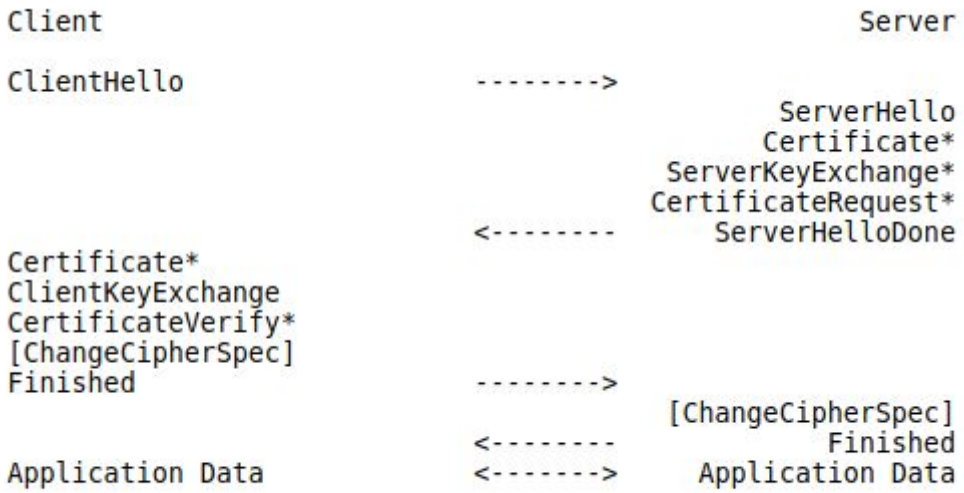

Figure 1. Message flow for a full handshake

[2]

Jakmile inicializační protokol doběhne, veškerá aplikační data budou moci být synchronně šifrována.

Návrh/implementace aplikace

Obecný návrh

Pro modulární návrh jsem se inspiroval Pythonovskými balíky a pro HTTP komunikaci jsem vytvořil modul requests. Ten má jen jednu funkci *get_response*, která vytvoří GET request na dané url podle argumentů. Dále jsem vytvořil jeden modu *arguments* pro zpracování argumentů programu, modul *xml_reader* který slouží k parsování přijatého XML souboru a nakonec modul *main* ve kterém se nachází i funkce *main*.
Ve funkci *main* se poté využijí všechny moduly nejdříve na přijetí argumentů, vytvoření seznamu url k dotazování, poté probíhá dotazování a parsování odpovědí.

Nejobtížnější pasáže

Nejobtížnější rozhodně bylo vyřešení TLS komunikace. Naštěstí mi s tímto velmi pomohl tutoriál [3] kde je tento problém do detailu vysvětlen a použit na příkladu.
Velmi užitečná byla funkce *SSL_CTX_set_default_verify_paths()*, která, v případě že není zadaná cesta k certifikátu, nastaví tuto cestu na výchozí cesty v daném systému.

Citace

1. T. Berners-Lee, RFC1945: Hypertext Transfer Protocol -- HTTP/1.0 [online]. 1996-04 [cit. 2018-11]. Dostupné z: https://tools.ietf.org/html/rfc1945

2. T. Dierks, E. Rescorla, RFC5246: The Transport Layer Security (TLS) Protocol, Version 1.2 [online]. 2008-08 [cit. 2018-11]. Dostupné z: https://tools.ietf.org/html/rfc5246

3. Secure programming with the OpenSSL API – IBM Developer [online]. Dostupné z: https://developer.ibm.com/tutorials/l-openssl/