Discord bot

implementovaný v jazyku C++

9. novembra 2020 Andrej Ježík

Obsah

1 Úvod

Cieľom projektu bolo vytvoriť program v jazyku C++, ktorý je schopný komunikovať cez ssl pripojenie s api Discordu. Bot má za úlohu pomocou aktivovaného tokenu sa pripojiť na server a kanál isa-bot kde má sledovať správy užívateľov a preposielať ich späť na server ako echo.

2 Pojmy

2.1 SSL

SSL alebo Secure Sockets Layer je protokol, ktorý slúži na založenie overeného a zašifrovaného spojenia medzi zariadeniami na sieti. SSL funguje pomocou priradzovania entít ako sú webstránky a spoločnosti ku kryptografickým párom kľúčov pomocou digitálneho dokumentu známeho ako SSL certifikát. Každý pár kľúčov pozostáva z privátneho a verejného kľúča. Privátny kľúč sa nikdy neposiela a verejný kľúč môže byť voľne distribuovaný pomocou certifikátu. Pomocou matematického vzťahu medzi kľúčmi je možné pomocou verejného kľúča zašifrovať správy, ktoré môžu byť následne rozšifrované iba pomocu privátneho kľúča. Vlastík privátneho kľúča taktiež môže s jeho pomocou podpísovať dokumenty. Podpis môže byť skontrolovaný hociktorým zariadením, ktoré má daný verejný kľúč.

2.2 HTTP

HTTP alebo Hypertext Transfer Protocol je protol aplikačnej vrstvy pre prenos hypermédiálnych dokumentov, ako sú HTML. Bol navrhnutý na komunikáciu medzi webovým prehliadačom a webovým serverom, ale taktiež môže byť použitý na iné účely. HTTP nasleduje klasický client-server model, podľa ktorého klient otvára spojenie aby odoslal požiadavku na server a čaká na odpoveď servera. HTTP je bezstavový protokol, čo znamená, že server si neuchováva žiadne dáta medzi dvomi požiadavkami.

2.3 HTTPS

HTTPS alebo HTTP cez SSL je použitie SSL (Secure Socket Layer) alebo TLS (Transport Layer Security) ako podvrstvy pod klasickou HTTP aplikačnou vrstvou. HTTPS zašifruje a rozšifruje uživateľské požiadavky, tak ako aj stránky, ktoré sú odpoveďou web serveru.

3 Návrh a implementácia

K implementácii bol zvolený viacparadigmový programovací jazyk C++. V sekcii implementačné detaily budú popísané podrobnejšie logické celky implementácie projektu. Na vytvorenie SSL spojenia boli využité knižnice openSSL. V projekte sa nevyskytujú relatívne nízko-úrovňové konštrukcie keďže knižnica OpenSSL zaobaľuje infraštruktúru soketov a špecifikáciu adries vo vysoko-úrovňových bezpečnostných konštrukciach.

3.1 Implementačné detaily

- *Argumenty*

  Na získanie argumentov používame knižnicu getopt. Trieda Arguments sa stará o získanie a spracovanie vstupných argumentov programu. Argumenty budú dostupné v objekte arguments.
  Podrobnejšie argumenty rozoberieme v nasledujúcej sekcii.

- *SSL spojenie*

  Vytvoríme si socket pomocou funkcie socket, vytvoríme si SSL štruktúru pomocou funkcie SSL_new

nastavíme deskriptor súborov fd ako vstupno/výstupné zariadenie pre SSL. Ako ďaľší krok inicializujeme SSL komunikáciu so serverom s funkciou SSL_connect.
Pre posielanie paketov nám slúži funkcia send_packet, ktorá využíva na zaslanie funkciu z SSL_write, pri chybe ukončí program s návratovou hodnotou -1.
Funcia recv_packet slúži následne pre prímanie odpovede od servera. Využívame funkciu SSL_read(), pri príjmaní zistíme či je odpoveď chunked alebo nie a podľa toho vieme kedy ukončiť čítanie.

- *HTTP požiadavky*

  Každý typ požiadavky má svoju funkciu:
  get_guilds zostaví požiadavku na ktorú sa nám vráti odpoveď s ID servermi na ktoré má bot prístup.
  get_channel zostavi požiadavku naktorú sa nám vráti odpoveď s informáciami o danom kanále a tak ďalej

- *Získanie a spracovanie správ*

  Pri štarte programu si uložíme identifikátor poslednej z každého kanálu, ktorý spracuvávame. Následne periodicky sledujeme kanály a porovnávame identifikátor poslednej správy z uloženým identifikátorom, ak sa líšia tak pošleme HTTP požiadavku o všetky správy ktoré boli pridané na kanál. Následne správy spracuvávame jednu po druhej. Taktiež testujeme či neprišla odpoved "Too much requests"v tom prípade počkáme daný čas a pokúsime sa request opäť odoslať.Tento postup máme v cykle ak by chybová správa prišla viac krát.

4 Návod na použitie

4.1 Spustenie programu

Program sa musí spustiť s parametrami, ktoré určujú jeho správanie počas behu. Paramatre sú na začiatku progamu vyhodnotené a pri nesprávnej kombinácii alebo neznámych parametroch bude program ukončený s nenulovým návratovým kódom

Zoznam parametrov a ich význam:

- *-h/–help*

  Parameter môže byť zadaný samostatne alebo s ľubovoľným iným parametrom ktorý popser podporuje. Tento parameter slúži na vypísanie krátkej pomocnej nápovedy na štandardný výstup a následne bude program ukončený.

- *-t/–token*

  Parameter udáva aký autetizačný token bude pre Discord bota použitý. Pri nesprávnom tokene bude program ukončný s návratovou hodnotou -1.

- *-v/–verbose*

  Parameter zmení chovanie programu a to tým, že bude vypisovať správy ktoré sa odošlú na discord kanál aj na štandardný výstup.
  Formát správy bude nasledový: echo <channel_id>: <username> - <message>.

- *-d/–debug*

  Program bude Vypisovať ladiace informácie, ktoré zahrňujú odpovede na požiadavky poslané serveru, kritické požiadavky odoslané na server, výpis kritických premenných v programe.

- *-p/–period*

Program dokáže zmeniť periódu dotazovania sa na server/y z počiatočných 2 sec, povolené sú hodnoty int > 0 pri zadaní čísla mimo intervalu alebo nečíselnej hodnoty sa program ukončí s návratovou hodnotou -1 a hláškou „Invalid argument“.

5 Bonusové Rozšírenia

5.1 Podpora viac kanálov

Program je schopný monitorovať viac kanálov, podľa toho koľko kanálov isa-bot sa na kanály vyskytuje.

5.2 Podpora viac serverov

Program je schopný monitorovať viac serverov, podľa toho na koľko serverov má bot povolený prístup.

5.3 Podpora argumentu -d/–debug

Podrobný popis v sekcii Zoznam parametrov a ich význam

5.4 Podpora argumentu -p/–period

Podrobný popis v sekcii Zoznam parametrov a ich význam

6 Záver

Počas implementácie som narazil na viac problémov, ktoré sa mi ale podarili vyriešiť, pre príklad:
Vysporiadať sa a zistiť ako sa príjmajú správy pri atribúte chunked a bez neho, keďže som to na začiatku nevedel tak sa neukončila funkcia SSL_read, ktorá ale už nemala čo čítať, keď celý obsah prišiel.
Program využíva na získavanie údajov z odpovedí serveru regexy, čo mi nepríde ako najlepšia prax pri fakte že existuje mnoho parserov, ktoré sú odladené a splňujú požiadavky na použitie na daný účel

Projekt fungoval na všetkých testovacích vstupoch a teda si myslím, že projekt je plne funkčný a zodpovedá zadaniu.

7 Zdroje

Literatúra

[1] SSL.com
`https://www.ssl.com/faqs/faq-what-is-ssl/`

[2] MDN web docs
`https://developer.mozilla.org/en-US/docs/Web/HTTP`

[3] Marty Kalin
`https://opensource.com/article/19/6/cryptography-basics-openssl-part-1`