VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačných technologií


# SIEŤOVÉ APLIKÁCIE A SPRÁVA SIETÍ
2020/2021

**Programovanie sieťovej služby**

**Monitoring SSL spojení**

16. novembra 2020 Dávid Oravec (xorave05)

# Obsah

# 1 Uvedenie do problematiky

## 1.1 Zadanie

V rámci projektu vytvorte jednoduchý nástroj, ktorý spracuje pcap súbor a zobrazí informácie o SSL spojení.

## 1.2 Prvotné zoznámenie s problemtikou

Základné znalosti ohľadom zachytávania paketov boli nadobudnuté už pri projekte do predmetu IPK. Pri príprave na riešenie projektu do predmetu ISA bolo nutné preštudovať niekoľko literárnych zdrojov o SSL spojeniach a paketoch so šifrovaním TLS ako napríklad [1].

## 1.3 Návrh aplikácie

Aplikácia `sslsniff` bola postavená na aplikácií `ipk-sniffer` z predmetu IPK. Aplikácia je tvorená z dvoch modulov komunikujúcich medzi sebou - `sslsniff.cc`[1] a `sslsniff.h`[2]

# 2 Implementácia

Aplikácia je naprogramovaná v jazyku C++ s podporou syntaxe jazyku C. Program začína spracovaním uživateľských argumentov. Na základe prepínačov sa otvorí zariadenie na odposluch buď online alebo offline. Otvorenie zariadenia je zabezpečené pomocou funkcií `pcap_open_online()` a `pcap_open_offline()`[3].

## 2.1 Podporované prepínače a obmedzenia

- **-i** *interface* ⇒ rozhranie na ktorom sa pakety zachytávajú
  - ak je prepínač zadaný, ale nie je zadané žiadne rozhranie, tak sa vypíše zoznam dostupných rozhraní systému

    Príklad spustenia:

    ```
    sudo ./sslsniff -i
    ```

  - ak je prepínač zadaný, ale zadané rozhranie nie je podporované, program končí s chybou

    Príklad spustenia:

    ```
    sudo ./sslsniff -i enp0s8
    ```

- **-r** *file* ⇒ súbor vo formáte pcapng
  - ak je zadaný tento prepínač, ale súbor neexistuje[4] alebo je v nesprávnom formáte, program končí chybou

    Príklad spustenia:

    ```
    sudo ./sslsniff -r this_file_doesn't_exists.pcapng
    ```

[1] Modul so zdrojovým kódom

[2] Hlavičkový súbor s prototypmi funkcií, využitými knižnicami a preddefinovanými makrami

[3] `https://github.com/the-tcpdump-group/libpcap/blob/master/pcap/pcap.h`

[4] Pri zadaní relatívnej cesty je nutné aby bol súbor v rovnakom priečinku ako projekt, inak je potreba zadať absolútnu cestu

  - pri zadaní prepínača a existujúceho súboru začne program tento súbor spracovávať

    Príklad spustenia:

    ```
    sudo ./sslsniff -r this_file_exists.pcapng
    ```

- V prípade, že nie je zadaný ani jeden z prepínačov, program končí a uživateľovi je vypísaná krátka nápoveda

  Príklad spustenia:

  ```
  sudo ./sslsniff
  ```

- V prípade, že sú zadané obidva prepínače správne, program uprednostňuje prepínač **-i**

  Príklad spustenia:

  ```
  sudo ./sslsniff -i enp0s3 -r this_file_exists.pcapng
  ```

Po spracovaní argumentov a otvorení zariadenia sa začne spracovanie paketu pomocou funkcie `pcap_loop()`, ktorá opakovane volá callback funkciu `processPacket()`.

## 2.2 Spracovanie paketu

Každý paket sa začína spracovávať v callback funkcií `processPacket()`. V tejto funkcií sú využité štruktúry z knižníc `netinet/ip.h`, `netinet/tcp.h` a `netinet/if_ether.h`. Keďže ide o monitoring SSL spojenia, tak sa spracúvajú len pakety s protokolom TCP.

## 2.3 Monitoring spojenia

Pre každý TCP paket, ktorý v sebe nesie príznak **SYN** je vytvorená štruktúra `connectionInfo`. V nej je uchovaná IP klienta spolu s portom, IP servera a port a časový údaj, kedy bol paket poslaný. Takto naplnená štruktúra je uložená do listu z knižnice `std::list`.

### 2.3.1 SSL spojenie

Aby bolo spojenie považované za SSL spojenie, musí prebehnúť pozdrav medzi klinetom a serverom.
Keď takýto pozdrav prebehne, tak štruktúra prislúchajúcemu spojeniu je aktualizovaná a nesie v sebe informáciu o úspešnom pozdrave.

V každom TCP pakete so šifrovaním TLS sa nachádza tzv. „TLS hlavička“, ktorú tvorí 5 po sebe idúcich bytov. Program kontroluje a hľadá v jednotlivých paketoch práve túto sekvenciu. Pokiaľ ju paket obsahuje, jedná sa o TLS paket, ktorý je súčasťou SSL spojenia.

Hlavičku tvorí typ záznamu, verzia protokolu a dĺžka dát v bytoch. Program podporuje verzie protokolu **TLS 1.0, TLS 1.1, TLS 1.2** a **TLS 1.3**.

## 2.4 Výpis ukončeného spojenia

Spojenie je monitorované od prvého paketu s príznakom **SYN** až po posledný paket s príznakom **FIN**. Aplikácia čaká na ukončenie spojenia ako aj zo strany klienta, tak zo strany servera.

Výnimkou je paket s príznakom **RST**. V takomto prípade je spojenie okamžite považované za ukončené a vypíše sa na štandardný výstup.

Na výpis ukončeného spojenia bola vytvorená funkcia `endConnection()`. Tá vyhľadá dané spojenie v liste štruktúr a overí či obidve strany ukončili spojenie príznakom **FIN**. Spojenie je po úspešnom vypísaní odstránené z listu štruktúr.

### 2.4.1 Formát výpisu

<timestamp>,<client ip>,<client port>,<server ip>,<SNI>,<bytes>,<packets>,<duration sec>

### 2.4.2 Popis položiek

- <timestamp> - je vo formáte `yyyy-mm-dd hh:mm:ss.s` a získava sa z prvého paketu spojenia
- <client ip> - IP adresa klienta
- <client port> - port klienta na ktorom komunikácia prebieha
- <server ip> - IP adresa servera
- <SNI> - názov serveru, nachádza sa v Client Hello správe
- <bytes> - veľkosť SSL dát, ktoré sa nachádzajú v „TLS hlavičke“.
- <packets> - počet všetkých paketov tvoriacich SSL spojenie (vrátane oboch paketov s príznakom **FIN**)
- <duration sec> - trvanie spojenia v sekundách

# 3 Popis implementácie jednotlivých položiek

## 3.1 <timestamp>

Timestamp udáva časový údaj o tom, kedy bolo SSL spojenie začaté. Je získaný z prvého paketu s príznakom **SYN**, ktorý otvára spojenie.

Pomocou funkcie `getTimestamp()` je časový údaj spracovaný a zapísaný k príslušnému spojeniu do štruktúry s dátami.

## 3.2 <client ip> a <server ip>

Na získanie týchto dvoch položiek slúžia funkcie `getSource()` a `getDest()`. Na základe toho, či sa jedná o IPv4 alebo IPv6 spojenie, je vo funkcií použitá príslušná štruktúra a následne z IP hlavičky vytiahnutá a spracovaná IP adresa.

## 3.3 <client port>

Port klienta je obsiahnutý v `struct tcphdr*`. Je nutné tieto dáta konvertovať do správnej podoby pomocou funkcie `ntohs()`.

## 3.4 <SNI>

„Server name indication“ sa nachádza v pozdrave od klienta (Client Hello message).[2] Nie je to však pravidlom a nie vždy sa SNI objavuje v Client Hello pakete.

Pri implementácii bol využitý kód z [3], ktorý bol modifikovaný pre účely projektu. Cieľom je dostať SNI z `Extensions` v pakete a uložiť ho do príslušnej štruktúry.

### 3.5 <bytes>

Celková dĺžka SSL spojenia v bytoch. Je získavaná z každej TLS hlavičky v pakete daného spojenia a následne prirátaná do atribútu príslušnej štruktúry spojenia.

### 3.6 <packets>

Počet paketov v jednom SSL spojení. Pakety sa počítajú od prvého TCP paketu s príznakom **SYN** až po druhý paket s príznakom **FIN**.

### 3.7 <duration sec>

Dĺžka trvania SSL spojenia v sekundách. Čas je získaný z prvého TCP paketu s príznakom **SYN** a z posledného paketu spojenia, ktorý obsahuje druhý príznak **FIN**.

## 4 Testovanie

Projekt bol testovaný na referenčnom Linux image pomocou open-source programu `Wireshark`.

Testovanie prebiehalo vytvorením pcapng súboru a následným overovaním správnosti výstupu aplikácie oproti zachyteným spojeniam vo Wireshark.

## 5 Riešenie problémových situácií

Najväčším problémom pri implementácií boli tzv. „Reassembled packets“.

Tieto pakety sú veľké a v programe Wireshark sa zobrazujú ako niekoľko rôznych paketov. Tento problém je riešený tak, že vždy, keď je v pakete nájdená TLS hlavička, uloží sa veľkosť dát *x* (nachádzajú sa v hlavičke TLS). Je známe, že ďalšia TLS hlavička sa môže objaviť až o *x* bytov ďalej a preto je ukazateľ posunutý o *x* bytov ďalej. Tam sa opäť vyhľadáva ďalšia TLS hlavička a takto to pokračuje, až kým veľkosť *x* nie je väčšia ako celková veľkosť momentálne spracúvaného paketu.

## 6 Manuálová stránka

Súčasťou zadania bolo vytvoriť manuálovú stránku pre aplikáciu. Vychádzal som z doporučenej literatúry na stránkach projektu.[4]

Pre zobrazenie manuálovej stránky stačí rozbaliť odovzdaný tar súbor do zložky, otvoriť v nej terminál a zadať `man -l sslsniff.1`

## 7 Použitie

Program je použiteľná aplikácia na monitorovanie SSL spojenia. Väčšina príkladov spustenia už bola spomenutá v 2.1.

Po stiahnutí archívu je nutné obsah rozbaliť do zložky. Následne v danej zložke skompilovať program pomocou `make`.

Na zachytávanie online je nevyhnutné púšťať program s administrátorskými právami pomocou `sudo`. Pri zadaní obidvoch prepínačov správne, je uprednostnené zachytávanie online. Online zachytávanie je ukončené po zaslaní signálu `KeyboardInterrupt`. V prípade potreby, sa pri spustení `sudo ./sslsniff` bez prepínačov zobrazí krátka nápoveda.

# Použitá literatúra


[1] SSL Introduction with Sample Transaction and Packet Exchange. online, may 2019. Dostupné z: https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-socket-layer-ssl/116181-technote-product-00.html

[2] Blake-Wilson, e. a.: Transport Layer Security (TLS) Extensions. RFC 3546, Network Working Group, jun 2003. Dostupné z: https://tools.ietf.org/html/rfc3546#section-3.1

[3] Lundquist, D.: sniproxy. 2012. Dostupné z: https://github.com/dlundquist/sniproxy/blob/master/src/tls.c

[4] Wirzenius, L.: Writing manual pages. online, oct 2020. Dostupné z: https://liw.fi/manpages/