Síťové aplikace a správa sítí 2020/2021

Monitoring SSL spojení

Daša Nosková (xnosko05)

Obsah

Kapitola 1

Úvod

Cieľom projektu je vytvoriť program na monitorovanie SSL spojení na danom rozhraní alebo vytvoriť analýzu z predaného *pcap* súboru.

1.1 SSL spojenie

Ssl spojenie poskytuje šifrovaný prenos dát medzi dvoma stranami. Dáta sa prenášajú paketmi rôznej veľkosti cez viac vrstiev. Spojenie sa nachádza medzi transportnou a aplikačnou vrstvou. Ssl spojenie začína klient správou typu client hello, ktorá obsahuje okrem iného aj rozšírenie SNI. Server odpovedá správou server hello v ktorej špecifikuje použitú verziu ssl spojenia a začína sa šifrovaná komunikácia nad tcp protokolom na základe dohodnutých atribútov komunikácie cez handshake výmenu.

1.1.1 SSL záznam

Základnou jednotkou SSL spojenia je záznam zložený z 5 B hlavičky a následujúcimi dátami. Hlavička obsahuje typ záznamu, verziu a dĺžku hlavičky[1]. Typy záznamov sú nasledujúce:

Handshake (0x16)

Handshake správy obsahujú dáta potrebné k handshaku[1].

- **Hello Request** (0x00)
- **Client Hello** (0x01) upresňuje verziu SSL protokolu cez ktorú by chcel klient komunikovať. Session ID, ktoré by chcel klient používať pre dané spojenie. Client hello ďalej obsahuje Cipher suite, čo je zoznam podporovaných šifrovacích algoritmov z klientovej strany a metódu kompresie - znovu list komprimovacých algoritmov podporovaných klientom[1]. Client hello môže obsahovať aj rozšírenia ako napríklad Server Name Identification - SNI.
- **Server Hello** (0x02) správa je podobná Client hello, ale obsahuje iba jednu Cipher Suite a jednu metódu kompresie. Upresňuje, ktorá verzia SSL bude nakoniec použitá[2].
- **Certificate** (0x0B)
- **Server Key Exchange** (0x0C)
- **Certificate Request** (0x0D)
- **Server Hello Done** (0x0E)

- **Certificate Verify** (0x0F)
- **Client Key Exchange** (0x10)
- **Finished** (0x14)

Change Cipher Spec (0x14)

Tento typ záznamu je používaný na oznámenie zmeny v kryptografických šifrách[1].

Upozornenie (alert) (0x15)

Záznamy typu upozornení slúžia na oznámenie udalostí ako upozornenia a chyby ako napr. uzatvorenie spojenia[1].

Dáta aplikácie (0x17)

Zašifrované aplikačné dáta.

Štruktúra paketu je ako na obrázku[1] 1.1 dole, pričom určité atribúty môžu mať variabilnú dĺžku alebo vôbec nebyť v zázname zahrnuté.

```
<pre><code>const unsigned char good_data_2[] = {
    // TLS record
    0x16, // Content Type: Handshake
    0x03, 0x01, // Version: TLS 1.0
    0x00, 0x6c, // Length (use for bounds checking)
        // Handshake
        0x01, // Handshake Type: Client Hello
        0x00, 0x00, 0x68, // Length (use for bounds checking)
        0x03, 0x03, // Version: TLS 1.2
        // Random (32 bytes fixed length)
        0xb6, 0xb2, 0x6a, 0xfb, 0x55, 0x5e, 0x03, 0xd5,
        0x65, 0xa3, 0x6a, 0xf0, 0x5e, 0xa5, 0x43, 0x02,
        0x93, 0xb9, 0x59, 0xa7, 0x54, 0xc3, 0xdd, 0x78,
        0x57, 0x58, 0x34, 0xc5, 0x82, 0xfd, 0x53, 0xd1,
        0x00, // Session ID Length (skip past this much)
        0x00, 0x04, // Cipher Suites Length (skip past this much)
            0x00, 0x01, // NULL-MD5
            0x00, 0xff, // RENEGOTIATION INFO SCSV
        0x01, // Compression Methods Length (skip past this much)
            0x00, // NULL
        0x00, 0x3b, // Extensions Length (use for bounds checking)
            // Extension
            0x00, 0x00, // Extension Type: Server Name (check extension type)
            0x00, 0x0e, // Length (use for bounds checking)
            0x00, 0x0c, // Server Name Indication Length
                0x00, // Server Name Type: host_name (check server name type)
                0x00, 0x09, // Length (length of your data)
                // "localhost" (data your after)
                0x6c, 0x6f, 0x63, 0x61, 0x6c, 0x68, 0x6f, 0x73, 0x74,
            // Extension
            0x00, 0x0d, // Extension Type: Signature Algorithms (check extension type)
            0x00, 0x20, // Length (skip past since this is the wrong extension)
            // Data
            0x00, 0x1e, 0x06, 0x01, 0x06, 0x02, 0x06, 0x03,
            0x05, 0x01, 0x05, 0x02, 0x05, 0x03, 0x04, 0x01,
            0x04, 0x02, 0x04, 0x03, 0x03, 0x01, 0x03, 0x02,
            0x03, 0x03, 0x02, 0x01, 0x02, 0x02, 0x02, 0x03,
            // Extension
            0x00, 0x0f, // Extension Type: Heart Beat (check extension type)
            0x00, 0x01, // Length (skip past since this is the wrong extension)
            0x01 // Mode: Peer allows to send requests
};
```

Obrázek 1.1: Všeobecná štruktúra ssl záznamu

[1] https://stackoverflow.com/questions/17832592/extract-server-name-indication-sni-from-tls-client-hello

Kapitola 2

Návrh aplikácie

Pri návrhu aplikácie sa vychádzalo zo štruktúry paketu ako na obr. 1.1. Aplikácia pracuje so základnou štruktúrou ssl záznamu, ktorá reprezentuje ssl spojenie. Záznamy sú ukladané do pola záznamov a vyhladávajú sa pomocou portu klienta, kedže na jednom porte môže byť naviazané maximálne jedno spojenie.

SSL spojenie je vždy nad TCP protokolom a teda začína príchodom SYN od klienta. Záznam je vložený do pola SSL záznamov, ak príde spomínaný SYN. Spojenie sa pokladá za vytvorené iba ak prídu obidve client hello aj server hello správy.

Pri návrhu aplikácie sa považuje spojenie ukončené prijatím prvého FIN z klientovej strany. Vypisujú sa iba ukončené spojenia. Neukončené spojenia sa zahodia.

Kapitola 3

Implementácia

Program je členený do súborov `sslsniff.c.h` `analyser.c.h` `packet.c.h` `error.h`.

3.1 sslsniff

Tento modul obsahuje metódu main a spracovanie argumentov zo vstupu. Po spracovaní argumentov sa zavolá funkcia `open_handler` z modulu analyser.c.

3.2 analyser

Vo funkcii `open_handler` sa alokuje buffer pre dané SSL spojenia. Každé SSL spojenie je reprezentované štruktúrou `struct ssl_data` zo súboru `packet.h`. Na základe prítomných parametrov -i a -r funkcia predáva riadenie funkciám `analyse_file_packets` a `analyse_interface_packets` v ktorých sa nastaví filter na filtrovanie tcp paketov cez `set_filter`. Funkcia `process_packet` parsuje jednotlivé časti paketu ako ip adresu, porty a podľa toho či nájde zdrojový port v bufferi vyhodnotí adresáta paketu ako klienta alebo server a podľa toho pokračuje funkciami `process_client` alebo `process_server`.

V prípade, že ide o paket od klienta so SYN flagom, vloží sa do bufferu ssl štruktúra.

Client hello a server hello sa kontrolujú nasledujúcim spôsobom, kde `CONTENT_B` reprezentuje nultý B v SSL zázname, kedže na tejto pozícii sa nachádza informácia o type záznamu. `HANDSHAKE_B` je 5. byte a obsahuje kód typu handshaku:

```
if((payload[CONTENT_B] == HANDSHAKE) && (payload[HANDSHAKE_B] == SERVER_HELLO))
```

V prípade client hello, sa nájde rozšírenie SNI a pridá do ssl štruktúry.

Každý tcp paket na danom porte sa pripočítava do celkového počtu prenesených paketov v danom spojení pomocou funkcie `increment_count_packets`. Funkcia `increment_bytes` spočítava dlžkú SSL hlavičiek, kedže jeden packet môže obsahovať viac hlavičiek postupne sa vypočítava začiatok nasledujúcej. Ak chceme získať celkovú hodnotu ssl záznamu musíme k danej dĺžke záznamu pripočítať ešte 5 B, ktoré zaberajú práve typ záznamu, verzia a spomínaná dĺžka.

Ak príde FIN flag od klienta, tak sa pokladá spojenie za ukončené, vypočíta sa jeho trvanie, vypíše sa na výstup a odstráni sa z bufferu ssl spojení.

3.3 packet

Súbor packet.c obsahuje pomocné funkcie na spracovávanie informácií z tcp paketu ako napríklad spracovanie ip adries, portov, kontrola flagov, výpočet dĺžky spojenia alebo výpis na výstup.

3.3.1 SNI

Najzaujímavejšou časťou je extrakcia SNI z paketu. Kedže pozícia SNI nie je fixná, postupne sa rozkúskujú variabilné časti ssl záznamu ako dĺžka šifier a rozšírení vo funkcii `get_ext_pos`. Keď sa nájde pozícia začiatku rozšírení tak sa postupuje obdobne až kým začiatok rozšírenia neodpovedá kódu rozšírenia SNI 0x00. Maximálne sa tento cyklus avšak vykoná 2krát, kedže SNI rozšírenie je vždy nanajvýš na druhej pozícii. Funkcia vráti -1 v prípade ak SNI nebolo nájdené alebo pozíciu začiatku SNI mena v prípade ak je toto rozšírenie podporované. K tomuto je použité makro `SNI_EXT_OFFSET` s hodnotou 9, pretože prvých 8 B rozšírenia SNI tvoria typ rozšírenia, dĺžka, dĺžka SNI listu, typ SNI a prístupom na získanú pozíciu v pakete získame dĺžku samothého mena serveru.

Keď je už známy byte dĺžky SNI serveru pokračuje funkcia `add_sni` v spracovaní. Ak bolo SNI rozšírenie nájdené, tak sa získa dĺžka SNI mena cez funkciu `get_len` a predá sa riadenie funkcii `get_SNI`, ktorá vráti meno serveru ako reťazec znakov.

Kapitola 4

Použitie

`./sslsniff -i rozhranie -r <file>`
`<file>` musí byť súbor typu .pcapng
V prípade použitia odpočúvania na rozhraní, musí byť aplikácia spustená ako root.

4.1 Formát výstupu

`<timestamp>,<client ip>,<client port>,<server ip>,<SNI>,<bytes>,<packets>,<duration sec>`
SNI = Server Name Identification rozšírenie,
bytes = počet prenesených B v SSL hlavičkách,
packets = počet prenesených paketov v danom spojení.

Bibliografie

[1] *SSL Introduction with Sample Transaction and Packet Exchange*. https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-socket-layer-ssl/116181-technote-product-00.html. Accessed: 2020-11-12.

[2] *Traffuc Analysis of an SSL/TLS Session*. http://blog.fourthbit.com/2014/12/23/traffic-analysis-of-an-ssl-slash-tls-session/. Accessed: 2020-11-12.