VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Síťové aplikace a správa sítí

Monitoring SSL spojení

2020/2021

Matej Otčenáš, xotcen01

Október 26, 2020

Obsah

1 Úvod

Projekt sa vzťahuje k predmetu *Síťové aplikace a správa sítí (ISA)*, vedený na univerzite *Vysoké učení technické v Brne* fakulta *Informačných technológií*.

1.1 Cieľ projektu

Úlohou je vytvoriť jednoduchý nástroj, ktorý je schopný spracovať a analyzovať pcap[1] súbor a zobraziť informácie o danom SSL/TLS spojení. Nástroj je rovnako schopný analyzovať živý odposluch komunikácie na vybranom sieťovom rozhraní. Analýza TLS/SSL spojení je možná a podporovaná pre verzie TLS 1.2 a nižšie.

1.2 Spustenie projektu

Na rozbalenie projektu je potrebné použiť príkaz `tar -xvf xotcen01.tar`. Použijeme príkaz `cd ./xotcen01`. Následne je potrebné program skompilovať. K tomu je nutné použiť jednoduchý príkaz `make`. V tomto štádiu máme spustiteľný program. Pre spustenie programu s použitím súboru zadáme následovný príkaz: `sudo ./sslsniff -r nazov_suboru.pcapng`. Analogicky pre spustenie programu so živým odposluchom použijeme príkaz: `sudo ./sslsniff -i nazov_rozhrania`. Pre ukončenie živého odposluchu použijeme klávesovú skratku `Ctr-C`.

1.3 Výstup projektu

Konečným výstupom programu môže byť *nula* až *n* monitorovaných TLS/SSL spojení vo formáte: *<timestamp>, <client_ip>, <client_port>, <server_ip>, <SNI>, <bytes>, <packets>, <duration_sec>*,

kde platí:

timestamp - udáva čas príchodu prvého TCP SYN paketu zo strany klienta

client_ip - IP adresa klienta, ktorý nadviazal spojenie so serverom

client_port - číslo portu klienta, z ktorého bola nadviazaná komunikácia

server_ip - IP adresa serveru, na ktorý bola nadviazaná komunikácia

SNI[2] (Server Name Indication) - je to rozšírenie TLS protokolu, ktorým

[1] https://www.tcpdump.org/pcap.html
[2] https://https.cio.gov/sni/

klient udáva, pod ktorým hostname-om sa pokúša pripojiť na začiatku handshake procesu

bytes - udáva počet bytov obsiahnutých v paketoch s TLS aplikačným protoklom. Program počíta so segmentáciou dát medzi viacero TCP packetov. Do veľkosti bytov sa nezahrňuje veľkosť TLS hlavičky.

packets - udáva celkový počet TCP paketov od nadviazania spojenia pomocou prvého TCP SYN až po posledný TCP FIN vrátane.

duration_sec - výsledná dĺžka nadviazaného spojenia v sekundách od prvého TCP SYN po posledný TCP FIN.

```
2020-10-20 16:13:55.281332,192.168.1.14,52010,99.86.239.49,static.twitchcdn.net,6415,18,30.358924
2020-10-20 16:14:04.519046,192.168.1.14,52063,147.229.9.21,wis.fit.vutbr.cz,15641,31,32.589666
```

Obrázok 1: Výstup programu verzia IPv4

2 Implementácia

Projekt bol implementovaný v jazyku C++11, vďaka ktorému bolo možné použiť jednoduchšiu manipuláciu s reťazcami pomocou STL knižníc. Napriek tomu značná časť programu používa základné konštrukcie jazyka C, z dôvodu využívania knižníc pre prácu s paketmi, ktoré sú implementované taktiež v jazyku C. Projekt je členený do viacerých modulov, vďaka ktorým je program prehľadnejší a ľahšie debugovateľný.

2.1 Problémy počas implementácie a ich riešenia

Počas implementácie programu som sa stretol s niekoľkými problémami, ktoré mierne komplikovali vývoj samotného programu, kde pri ich riešení bola použitá primárne referenčná literatúra, vďaka ktorej sa implementácia značne zjednodušila. Riešenia problémov boli následovné:

1. Získanie SNI:

   Na získanie SNI je potrebné predovšetkým získať prvotný TLS Client Hello paket, ktorý zasiela strana klienta. Keďže nebolo možné použiť akékoľvek dostupné knižnice sa spracovanie TLS packetu, je nutné daný TCP payload[3] parsovať "manuálne", tzn. využiť pointrovú aritmetiku a

[3] https://techterms.com/definition/payload

bitové operácie. Aby bolo možné spracovať korektné serverové meno, je potrebné sa posúvať po určitých veľkostiach bytov od TLS hlavičkového záznamu. Program počíta s variabilitou veľkostí rôznych častí TLS záznamu a dokáže na ne korektne odpovedať, aby nedošlo ku chybnému získaniu serverového mena [Gro08] [Dri18].

2. Získanie počtu bytov:

   Získanie výsledného počtu bytov TLS paketov je pravdepodobne najkomplikovanejšia časť programu, nie tak z pohľadu štruktúry, ako z pohľadu logiky. Je nutné sa vysporiadať s tzv. segmentáciou paketov[4], čo narušuje uniformnosť štruktúry TLS dát v jednotlivých paketoch. Aby bolo možné získať správnu veľkosť bytov, je potrebné uvažovať, že v každom TCP payload-e spracovávaného packetu existuje TLS hlavička, ktorej veľkosť sa vždy odpočíta od celkovej veľkosti TCP payload-u. Pre lepšie pochopenie je použitý následovný pseudokód [Sch19]:

```
while(pocet_bytov++ < payload) {
    if(ukz == TLS_hlavicka_typ1 ||
       ukz == TLS_hlavicka_typ2 ||
       ukz == TLS_hlavicka_typ3 ||
       ukz == TLS_hlavicka_typ4)
    {
        sucet_hlaviciek += 5; //+5 velkost TLS
            hlavicky
        posun_o = posun_o_velkost_hlavicky(ukz,
            payload);
        ukz = posun_ukz(ukz, posun_o + 4);//+4 posun
            na koniec hlavicky
        pocet_bytov += posun_o + 4;
    }
    else ukz++; // posun ukazatela na dalsi byte
}
velkost_bytov = payload - sucet_hlaviciek
```

3. Získanie TCP príznakov [5] (*angl. flags*):

   Pri získavaní príznakov je možné si prvotne zjednodušiť samotný prístup s použitím knižničných funkcií. Knižnica `netinet/tcp.h` poskytuje poskytuje vo svojich štruktúrach premennú `th_flags`, ktorej vymaskovaním sa efektívne dala nadobudnúť hľadaná hodnota príznaku (napr. SYN, ACK, FIN). Problém však nastáva pri použití knižnice na vzdialenom stroji *merlin*, ktorý uvedenú premennú vo svojej knižnici neob-

[4] https://www.inetdaemon.com/tutorials/basic_concepts/communication/segmentation_and_reassembly.shtml
[5] https://www.geeksforgeeks.org/tcp-flags

sahoval (dôvod nebol zistený). V takomto prípade bolo potrebné použiť znovu pointrovú aritmetiku a využitie bitových operácií, tzn. "manuálne" parsovať TCP hlavičku, ako príklad uvediem získanie príznaku SYN priamo v jazyku C++: `auto SYN = ((*(tcp_hdr + 13)) & 2)` [Inf81].

4. Validácia úspešne zahájenej a ukončenej komunikácie

   Aby bolo možné považovať monitoring SSL za úspešne zahájenú je nutné obdržať v prvom kroku dva TCP pakety s nastaveným príznakom SYN, ktorý značí začiatok komunikácie. V druhom kroku musí prebehnúť korektný SSL/TLS *Handshake*, čo znamená že budú obdržané pakety so správou *Client Hello* a *Server Hello*. Za korektné ukončenie komunikácie sa považujú dva TCP pakety s nastaveným príznakom FIN, ktorý značí ukončenie komunikácie alebo jeden paket s príznakom RST, ktorý značí, že niečo nie je v poriadku s TCP komunikáciou. Program avšak považuje nastavený príznak RST ako za validné ukončenie spojenia. Pri všetkých týchto prípadoch sa počíta s najbežnejšou kombináciou *request* od klienta a *response* od servera.

3 Návrh programu z pohľadu podpory protokolov

Program podporuje v sieťovej vrstve oba protokoly IP, tzn. IPv4 a IPv6. Na transportnej vrstve je program zameraný výlučne na protokol TCP z dôvodu monitoringu SSL/TLS spojení. Posledné z podporovaných protokolov sú na aplikačnej vrstve práve protokoly SSL/TLS. Tie majú však niekoľko verzií. Pri protokole SSL sa jedná o verzie SSLv2 a SSLv3, ktoré ale neobsahujú SNI, čo program rozpozná a ako riešenie poskytne pre SNI len prázdny reťazec. V prípade TLS sú to verzie TLSv1.0 až TLSv1.3. Program ale podporuje verzie TLSv1.2 a nižšie. [Gri13].

```
2020-10-09 17:41:06.226817,2001:67c:1220:c1b0:f07b:9da4:45d3:caa9,45548,2a00:1450:4014:80d::200e,photos.google.com,5174,16,0.066527
2020-10-09 17:41:07.249191,2001:67c:1220:c1b0:f07b:9da4:45d3:caa9,38228,2a00:1450:4014:800::2003,ssl.gstatic.com,4031,18,0.066876
```

Obrázok 2: Výstup programu verzia IPv6

```
2020-10-07 10:44:27.052213,127.0.0.1,33007,127.0.0.1,,2252,15,0.509794
2020-10-07 10:44:38.300458,127.0.0.1,43259,127.0.0.1,,2251,15,0.505479
```

Obrázok 3: Výstup programu s chýbajúcim SNI

4 Využitie vytvoreného programu

Program je možné využiť pri jednoduchej analýze a monitoringu SSL/TLS spojení. V prípade, že sa snažíme získať napríklad prenesené množstvo bytov v TLS paketoch, alebo celkový počet prenesených TCP paketov od zahájenia komunikácie pomocou prvého TCP SYN paketu až po druhý TCP FIN paket.

5 Testovanie programu

Program bol vyvíjaný v prostredí Linux distribúcie Arch, na ktorej bol kontinuálne aj testovaný. Súčasne bol program testovaný aj na dodanom referenčnom prostredí Linux distribúcie Ubuntu 18.04 a taktiež na vzdialenom stroji Merlin, ktorý je poskytovaný fakultou. Testovacie súbory dosiahli zhodné výsledky na všetkých strojoch. Pri testovaní boli použité mnohé testovacie súbory typu *pcapng*.

5.1 Porovnávanie s programom Wireshark

Pre vývoj a testovanie bol použitý program Wireshark[6], ktorý bol použitý napríklad v prípade ručného spočítania prenesených bytov TLS komunikácie na určitom porte a následne porovnaný s výstupom programu, pričom sa zhodovali počty bytov, paketov, výsledky časov a pod.

Zdroje

[Inf81] California 90291 Information Sciences Institute University of Southern California 4676 Admiralty Way Marina del Rey. *TRANSMISSION CONTROL PROTOCOL*. RFC 793. RFC Editor, Sept. 1981. URL: `https://tools.ietf.org/html/rfc793`.

[Gro08] Network Working Group. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. RFC Editor, Aug. 2008. URL: `https://tools.ietf.org/html/rfc5246`.

[Gri13] Ilya Grigorik. "Transport Layer Security (TLS)". In: (2013). URL: `https://hpbn.co/transport-layer-security-tls/`.

[Dri18] Michael Driscoll. "The Illustrated TLS Connection". In: (2018). URL: `https://tls.ulfheim.net/`.

[Sch19] Jan Schaumann. "Capturing specific SSL and TLS version packets using tcpdump". In: (Mar. 2019). URL: `https://www.netmeister.org/blog/tcpdump-ssl-and-tls.html`.

[6] https://www.wireshark.org/