Monitoring SSL spojení

Marián Lorinc - xlorin01

Úvod do problematiky SSL monitoringu

SSL je protokol na zaistenie bezpečného prenosu informácií medzi klientom a serverom. SSL pracuje medzi vrstvami transportnej vrstvy (iba nad TCP protokolom) a aplikačnej vrstvy.

Formát SSL protokolu

SSL sa delí na viacero pod protokolov:

- TLS Record protokol
- Handshake
- Change Cipher Spec
- Alert
- Application data

V ďalších podkapitolach nebudú spomenuté protokoly Change Cipher Spec, Alert a Application data. Tieto protokoly neboli významne pre projekt.

TLS Record protocol

TLS record protokol slúži na ako metadata pre ostatné protokoly. Súčasťou tohto protokolu sú dáta ako typ ďalšieho pod protokolu, verzia SSL a veľkosť ďalšieho pod protokolu. Za týmito metadatami už nasledujú dáta pod protokolu.

Handshake

Protokol handshake sa používa na začiatku SSL spojenia, ktoré nie je ešte šifrované. Pri vykonávaní handshake dochádza k výmene kryptografických kľúčov, session id, druh kompresie a ďalších potrebných dát k prenosu. Taktiež Handshake, presnejšie Client Hello, obsahuje SNI, ktoré program má za úlohu odchytiť. SNI sa nachádza v rozšírení *server_name*. Rozšírenie obsahuje dáta, ktoré nie sú povinné pre protokol Handshake.

Vznik SSL spojenia

SSL pracuje nad protokolom TCP. To znamená, že predtým ako vytvoríme SSL spojenie, musíme vytvoriť TCP spojenie. TCP spojenie sa vytvára protokolom Handshake.

Postup vytvorenia spojenia TCP

1. klient odošle serveru paket s príznakmi SYN a vygeneruje nové sekvenčné číslo
2. server odpovie paketom SYN + ACK, odošle rovnaké sekvenčné číslo a acknowledge číslo nastaví na hodnotu sekvenčné číslo + 1

3. klient odpovie príznakom ACK a odošle serveru sekvenčné číslo, ktoré má istú hodnotu ako acknowledge číslo zo serverovej odpovede a na nastaví acknowledge číslo na istú hodnotu ako sekvenčné číslo

Postup vytvorenia spojenia SSL

1. klient odošle serveru Client Hello správu. Ak klient chce obnoviť minulé spojenie, môže poslať rovnaké sessionId ako pri poslednom spojení.
2. server odpovie klientovi Server Hello správou, ktorá môže byť doplnená dodatočnými správami.

Návrh aplikácie

Aplikácia je naprogramovaná v objektovom orientovanom jazyku C++. V nasledujúcich podkapitolách bude členený návrh podľa sieťových vrstiev.

Linková vrstva - Dátový rámec

Dátový rámec je pomerne jednoduchý na spracovanie. Na jeho spracovanie sa použila štruktúra zo štandardnej knižnice. Dátový rámec na konci môže obsahovať dodatočné nuly, ak by celkový rámec bol menší ako 64 bajtov. Tieto dodatočné nuly môžu spôsobovať v programe problémy, tak, že program si ich pomýli za skutočné dáta. Avšak tento problém sa rieši na sieťovej vrstve.

Sieťová vrstva - Paket

Na sieťovej vrstve je to o niečo komplikovanejšie. Momentálne sa v Internete používajú dve verzie IP protokolu. IPv6 a IPv4. Na vyriešenie tohto problému slúži trieda IPHelper, ktorá má spoločne rozhranie pre oba protokoly. Spracovanie IPv4 hlavičky je pomerne jednoduché. V prípade projektu nebolo potrebné pristupovať k dodatočným parametrom(options). IPHelper v prípade hlavičky IPv4 kopíruje potrebné dáta a prevedie dáta zo sieťového endiannu do systemoveho endiannu. Veľkosť hlavičky vynásobí koeficientom 4. V prípade IPv6 je to komplikovanejšie. IPv6 hlavičku môžu dopĺňať IPv6 rozšírenia a fragmenty. IPHelper v tomto prípade prečíta celú hlavičku a rozšírenia (rozšírenia sú vo formáte lineárneho zoznamu) a zastaví sa, až keď ďalšia hlavička má hodnotu TCP. Počas prechádzania rozšírení a fragmentov IPHelper počíta veľkosť IPv6 hlavičky + všetkých rozšírení + fragmentov.

IP hlavičky obsahujú dôležitú hodnotu, ktorá je length. Pomocou tejto hodnoty sa dá odvodiť veľkosť dát na aplikačnej vrstve. V prípade IPv4, hlavička length obsahuje veľkosť IPv4 hlavičky, TCP hlavičky, SSL dát. V prípade IPv6 hlavička length obsahuje veľkosť rozšírení, fragmentov, TCP hlavičiek a SSL dát. Narozdiel od IPv4 neobsahuje veľkosť samotnej IPv6 hlavičky. V niektorých prípadoch napríklad pri jumbotrone. V takomto prípade sa veľkosť dát odvádza od veľkosti paketu z pcap hlavičky.

Transportná vrstva - Segment

Hlavičku transportnej vrstvy je pomerne jednoduché spracovať ako to bolo aj v prípade linkovej vrstvy. Keďže hlavička TCP je fixná a neobsahuje rozšírenia ako to bolo v prípade IPv6, je možné použiť štruktúru zo štandardnej knižnice.

Náročnejšou úlohou je zo synchronizovať pakety a zoradiť ich v poradí v akom prišli. Pakety počas jedného spojenia nemusia byť smerované istou cestou v sieti. Niektoré linky sú rýchlejšie, čo spôsobuje, že pakety neprídu v istom poradí, v akom boli odoslané. Program rieši tento problém iba čiastočne. Program predpokladá, že nedôjde ku žiadnej chybe prenosu a žiadne pakety nebudú preposielané odznovu. Taktiež predpoklada, že pakety, ktoré obsahujú dáta vyšších vrstiev a majú rovnaké sekvenčné čísla pridu neskoršie ako pakety bez dát. Ak sa poruší táto podmienka, paket bez dát sa zahodí. Program očakáva, že ďalší paket bude mať sekvenčné číslo *posledný paket seq + posledny paket veľkosť dát vyššej vrstvy*. Táto kontrola sa vykonáva pre oba smery.

Aplikačná vrstva - Dáta

Do tejto vrstvy patrí SSL + samotné aplikačné dáta. Na tejto vrstve už dáta prichádzajú v správnom poradí,ale môžu byť fragmentované. Fragmentácia dát je riešená pomocou triedy TlsParser. TlsParser sa vytvorí pre každé spojenie dvakrát. Je to z toho dôvodu, že program monitoruje komunikáciu oboma smermi. TlsParser funguje na princípe konečného automatu a visitor návrhového vzoru. Návrhový vzor visitor je na túto implementáciu vhodný, keďže sa môže stať, že napríklad pole length v Record Layer protokolu bude fragmentovaný. TlsParser si v takom prípade vie zapamätať odsadenie do premennej, ktorá neprečítala úplnú hodnotu. TlsParser počas toho ako spracováva dáta, validuje dáta ako je napríklad *handshake type* a *ssl version*. Taktiež úlohou TlsParser je konvertovať endiannes číselných dát.

Analýza SSL hlavičiek

Pri spracovávaní SSL hlavičiek program pracuje s hodnotami ako *length* a *content type*. Ak je v *content type* inakšia hodnota ako hodnota *Handshake*, Record Protokol hlavička sa preskočí. V prípade, že *content type* má hodnotu *Handshake,* spracováva sa pod protokol Handshake. Pri Hanshake pod protokole program vyhľadá hodnotu *handshake type.* Ak je *handshake type Client Hello*, spracuje sa celá hlavička. Ak je *handshake type Server Hello* poznačí sa, že sa vyskytla v spojení. Pri *Client Hello* program hľadá hodnoty *length, extension length a odsadenie ku rozšíreniam*. Pri rozšíreniach je to obdobne ako pri *Client Hello*, opäť sa hľadá typ rozšírenia (v prípade programu SNI).

Manažment paketov

Program prijíma nové pakety od knižnice pcap pomocou metódy *pcap_loop.* Tie sa následné posielajú do inštancie triedy SessionManager. SessionManager má na

starosti triedenie paketov do tcp pripojení pomocou kombinácie portov a IP adries. Ak SessionManager nájde spojenie, kde by mohol paket patriť, skúsi ho vložit do spojenia. Spojenie, inštancia triedy Session, má na starosti zoraďovanie paketov podľa sekvenčných čísel a posielanie dát do TlsParserov. Spojenie môže odmietnuť paket, ak nespĺňa podmienku sekvenčného čísla alebo ak je spojenie ukončené. Každý paket v SessionManager bufferi, ktorého časový rozdiel medzi posledným paketom v bufferi prevýši timeout, je vyhodený ako neplatný paket.

Návod na použitie programu

Kompilácie

make

Vymazanie kompilačných súborov

make clean

Analýza spojení

sslsniff [-i <rozhranie>] [-r <pcapng súbor>]

Ak je spojenie nekorektne zakončené, miesto hodnoty duration je hodnota -.
Ak je ACK paket s rovnakym SEQ číslom za ACK paketom s dátami, tento paket sa zahodí (implementačný problém).

Literatura

https://en.wikipedia.org/wiki/IPv4
https://en.wikipedia.org/wiki/IPv6_packet
https://tools.ietf.org/html/rfc5246
http://blog.fourthbit.com/2014/12/23/traffic-analysis-of-an-ssl-slash-tls-session/
https://tools.ietf.org/html/rfc793
https://www.tcpdump.org/pcap.html
https://en.wikipedia.org/wiki/Ethernet_frame