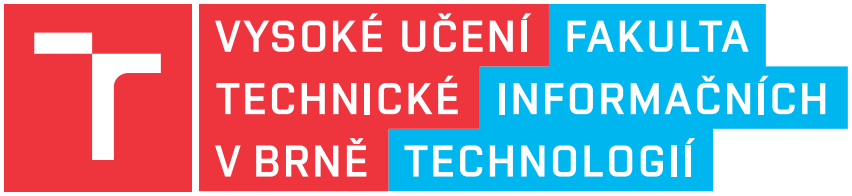

Export DNS informácií pomocou protokolu Syslog

Projekt – ISA

18. Novembra 2018

Timotej Halás

Obsah

1 Úvod

Cieľom projektu je zoznámiť sa s protokolom DNS, protokolom Syslog a naprogramovať C++ nástroj. Ten bude zachytávať sieťovú prevádzku alebo čítať súbory pcap obsahujúce pakety a filtrovať z dát iba pakety, ktoré obsahujú odpovede protokolu DNS. Z týchto vyfiltrovaných dát si bude robiť štatistky a tie bude následne podľa nastavených parametrov programu buď posielať na zadaný Syslog server alebo ich bude vypisovať na štandardný výstup. Ak je nastavené sledovanie sieťovej prevádzky z rozhrania, štatistiky budú spracované v intervaloch nastavených parametrom -t. Ak nie je nastavený, štatistky budú spracované každých 60 sekúnd. V prípade, že je nastavené čítanie súboru, štatistiky budú spracované po prečítaní súboru.

2 Podporované protokoly podľa vrstiev

Pre zachytávanie sieťovej komunikácie a čítanie súborov s uloženou sieťovou komunikáciou bude použitá knižnica libpcap. Dáta, ktoré budú získané pomocu knižnice, majú dáta, ktoré môžu obsahovať viac vrstiev. Každá vrstva obsahuje informácie o tom, odkiaľ kam paket cestuje, či už na hardvérovej vrstve alebo softvérovej. Iba posledná aplikačná vrstva obsahuje dáta určené priamo pre danú aplikáciu kam paket cestoval. Každá vrstva môže obsahovať niekoľko hlavičiek rôznych protokolov. Aby sme sa dostali k dátam na aplikačnej vrstve, musíme tieto hlavičky iných vrstiev odbaliť.

2.1 Linková vrstva

Na linkovej vrstve sa najčastejšie vyskytuje protokol Ethernet, ktorý zaobaľuje ostatné dáta vyššej vrstvy. Obsahuje MAC adresu zdrojového zariadenia a zariadenia, na ktoré sú dáta posielané. Obsahuje tiež informácie o tom, aký nasleduje protokol vyššej vrstvy v nasledujúcich dátach. Ďalej sa tu často vyskytuje IPv4 protokol alebo IPv6 protokol, ktoré budú podrobnejšie vysvetlené v nasledujúcej kapitole. Občas nastane prípad, keď sa v linkovej vrstve vyskytne protokol IPv4 alebo IPv6 zo sieťovej vrsty. Posledný prípad, ktorý v projekte riešim je Linux cooked capture, ktorý tak isto ako ostatné protokoly, ktoré sa môžu vyskytnúť v linkovej vrstve, obsahuje informácie o protokole v ďalšej vrstve.

2.2 Sieťová vrstva

Na sieťovej vrstve riešim protokoly IPv4 a IPv6. Tieto protokoly obsahujú IP adresu zdrojového zariadenia a zariadenia, na ktoré sú dáta posielané. Taktiež obsahujú informácie o tom, aký protokol sa nachádza v dátach nasledujúcich po IP hlavičke. Môžu nastať situácie, kedy za sebou nasleduje viac IP hlavičiek, ktoré tiež riešim. Pri IPv6 existujú rôzne rozšíriteľné hlavičky, ktoré môžu nasledovať po základnej IPv6 hlavičke. Jedna z týchto hlavičiek je Fragmentová hlavička v sekcii 4.5 v dokumente RFC2460. Tá je pridávana k základnej IPv6 hlavičke, ak je paket fragmentovaný, čiže rozdelený na viac častí. Obsahuje informácie o offsete tohto fragmentu v celých dátach, ktoré boli rozdelené, a či ešte prídu ďalšie fragmenty. Tieto dáta existujú aj v IPv4 hlavičke.[8]

2.3 Transportná vrstva

V transportnej vrstve sú väčšinou použité protokoly TCP a UDP. Dá sa z nich zistiť, na aký port a z akého zdrojového portu idú. V tomto prípade zdrojový port musí byť 53, pretože z portu 53

prichádzajú správy DNS. Protokol TCP môže byť segmentovaný. Podobne ako pri IP fragmentácií, z TCP hlavičky sa dá zistiť offset dát v celých dátach, ale v tomto prípade je offset relatívny voči offsetu prvého fragmentu. Z príznakov v TCP hlavičke sa dá zistiť, či bol segment dát posledný.

2.4 Aplikačná vrstva

2.4.1 DNS

Z dát ktoré nasledujú po hlavičke transportnej vrstvy beriem do úvahy iba tie, ktoré obsahujú dáta formátu DNS správy. Tá je rozdelená na viaceré časti. Prvou z nich je DNS hlavička. Obsahuje informácie o tom, či správa je požiadavka na server alebo odpoveď zo serveru, počet otázok, ktoré sa nachádzaju v správe, počet odpovedí a rôzne iné informácie, ktoré sú potrebné pri parsovaní DNS správy. Ignorujem správy, ktoré majú nastavený príznak QR popísaný v sekcii 4.1.1 dokumentu RFC2460 na hodnotu 0. Ďalej nasleduje otázka, na ktorú sa pýta klient, a posledná sekia je sekcia odpovedí, ktorých môže byť viac. Odpovede môžu byť rôznych typov. Program musí podporovať typy DNS odpovedí: A, AAAA, CNAME, PTR, MX, NS, SOA, TXT, SPF, DNSKEY, RRSIG, NSEC, DS. [8]

2.4.2 Syslog

Syslog je protokol, ktorý slúži na zaznamenávanie alebo posielanie správ na Syslog server. Správy môžu mať rôzne priority a typy správ tiež môžu byť rôzne. Správa musí mať konkrétny formát tak, ako je popísane v dokumente RFC5424 [1]. Syslog protokol sa v tomto projekte používa na zasielanie štatistík na server určený parametrom -s. Štatistiky sa budú odosielať po čase určenom parametrom -t. Ak nie je zadaný, budú sa odosielať po 60 sekundách.

3 Štatistiky

Štatistiky budú získavané z odpovedí v protokole DNS. Štatistiky sa vytvárajú ako počet rovnakých odpovedí. Príklad štatistík:

www.google.cz. A 172.217.19.67 15
wis.fit.vutbr.cz. CNAME agata.fit.vutbr.cz. 8

Prvý riadok znamená, že odpovedí typu www.google.cz. A 172.217.19.67 sa počas pracovania programu vyskytlo 15. Druhý riadok znamená, že odpovedí typu wis.fit.vutbr.cz. CNAME agata.fit.vutbr.cz. sa vyskytlo 8.

4 Návrh programu

Keďže C++ poskytuje objektovo orientovaný prístup k návrhu programu, rozhodol som sa ho použiť. Program mám rozdelený do viacerých tried, ktoré budú popísané ďalej.

4.1 Spracovanie vrstiev paketu

Pred implementáciou som si postupne vytvoril triedy pre každú jednu vrstvu. Pre parsovanie IP správ som vytvoril triedy ipv4 a ipv6 a triedu ip_fragments, ktorá sa stará o skladanie fragmentovaných IP

dát či už IPv4 alebo IPv6 paketov. IPv4 som implementoval za použitia dokumentu RFC791 a pri IPv6 je to dokument RFC2460. Pre TCP som vytvoril triedu tcp, ktorá sa stará o spracovávanie TCP hlavičky a tcp_segments, ktorá slúži na skladanie segmentovaných TCP dát. Pred implementáciou som si postupne vytvoril triedy pre každú jednu časť DNS správy. Prvou je hlavička (dns_header), ďalšou otázka (dns_question), zhluk otázok (dns_questions), ak sa otázok v správe nachádza viac a nakoniec záznam (dns_resource_record), ktorý využíva trieda DNS záznamy (dns_resource_records). Tá len zlukuje tieto záznamy. Pre parsovanie rôznych typov DNS odpovedí je trieda dns_type, ktorá obsahuje implementáciu parsovania všetkých podporovaných typov. Pri DNSSEC typoch RRSIG a DNSKEY je potrebné časť záznamov vypisovať vo formáte base64. Na spracovanie som použil implementáciu pána Reneho Nyffeneggera. Tieto časti spracovania DNS správ ďalej využíva trieda dns, ktorá ich použije v správnom poradí, aby získali údaje z dát paketu. Ak nastane nejaká chyba pri čítaní dát alebo program pristúpi do nepovolenej pamäti, je vyhodená niektorá z výnimiek popísaných v súbore exceptions.h. Spracovanie paketu je ukončené a začne sa spracovávať ďalší paket. Trieda packet_parser využíva všetky IP, TCP a DNS triedy, aby spracovala každý jeden paket. [6, 2, 3, 4, 8, 9, 7, 5]

4.2 Syslog a štatistiky

Obe položky som implementoval podľa návrhového vzoru singleton, pretože za behu programu treba pristupovať iba k jednej inštancií štatistík a tak isto je iba jedna inštancia kódu, ktorý odosiela štatistiky. Syslog je implementovaný v triede syslog. Pri posielání správ využívam protokol DNS. Správu formátujem podľa formátu v dokumente RFC5424. Štatistiky sú riešené v triede statistics. Zaobaľujú mapu, ktorej kľúčom je jedna DNS odpoveď, a inkrementáciou hodnoty tohto kľúča sa počíta počet rovnakých odpovedí. [6, 1]

4.3 Multithreading – Riešenie časovača pre odosielanie správ a reakcia na signál SIGUSR1

Jedným z problémov pri implementácii bola situácia, kedy by ubehol čas pre odoslanie alebo výpis získaných štatistík bez straty práve spracovávaných dát. Pre vyriešenie tohto problému som sa rozhodol použiť vlákna. Program je rozdelený na 3 vlákna. Hlavné vlákno zachytáva sieťovú prevádzku a spracováva pakety a po spracovaní použije triedu statistics, aby do nej pridala získané dáta z aktuálneho paketu. Druhé vlákno je v nekonečnom cykle, kde spí po dobu zadanú parametrom -t. Keď sa zobudí, pristúpi k triede statistics, získa z nej štatistky a následne ich odošle na server alebo vypíše na štandardný výstup podľa toho, či je alebo nie je nastavený server parametrom -s. Tretie vlákno rieši spracovávanie signálu SIGUSR1. Ak program dostane signál SIGUSR1, pristúpi k triede statistics, prečíta z nej dáta a vypíše ich na štandardný výstup. Keďže všetky vlákna pristupujú k triede statiscis, môže nastať situácia, kedy hlavné vlákno zapisuje do triedy statistics a v tom momente z neho bude druhé alebo tretie vlákno chcieť čítať čo nie je dobré. Pre vyriešenie tohto problému je prístup do triedy statistics chránený mutexom, ktorý dovolí k tejto triede prístup naraz iba jednému vláknu a iné vlákno si bude musieť počkať, kým aktuálne pristupujúce dokončí svoju prácu. Vytvorenie vlákien je riešené v triede dns-export.

4.4 Riešenie IP fragmentácie a TCP segmentácie

TCP segmentácia a IP fragmentácia fungujú obdobne. IPv4 a IPv6 majú jediný rozdiel pri fragmentácií v tom, že v IPv4 sú informácie o fragmentoch priamo v hlavičke a pri IPv6 je použitá rozšírená hlavička o fragmentovú hlavičku, v ktorej sa nachádzajú tieto informácie v rovnakom

formáte ako v IPv4 hlavičke. Z fragmentovaného/segmentovaného paketu sa získa údaj offsetu dát a informácia o tom, či je aktuálny paket posledný. Rozdiel je v tom, že IP paket obsahuje offset číslovaný od 0, ale TCP paket obsahuje relatívny offset od prvého sekvenčného čísla v prvom pakete, ktoré môže byť rôzne od 0. K akému paketu patria fragmenty v IP protokole identifikujem podľa ID, zdrojovej a cieľovej IP adresy, ktoré sa dajú zistiť z IP hlavičky. Pri TCP identifikujem segmenty podľa položky ACK v hlavičke, zdrojového a cieľového portu. Fragmenty/segmenty zbieram do mapy, ktorej kľúčom je offset a dáta sú fragmenty/segmenty, ktoré nasledujú za hlavičkami. Ak prišiel posledný paket a veľkosť prijatých dát je rovnaká ako očakávaná veľkosť dát, ktoré majú prísť, spojím tieto dáta do jedného vektoru a pokračujem v parsovaní dát ako DNS správy. Pakety môžu prísť v rôznom poradí, pretože využívam vlastnosť mapy, ktorá zoraďuje kľúče v tomto prípade offsety a tým pádom sa nikdy nemôže stať, že by pakety neboli zoradené v správnom poradí aj keď prišli v nesprávnom. [2, 3, 8]

4.5 Trieda memory_block

Pri tejto problematike môže nastať veľmi jednoducho prípad prístupu do nepovolenej pamäti napríklad pri komprimovaných DNS paketoch, kde treba skočiť do inej časti pamäti. Taktiež môže nastať pri spracovaní nejakého datového typu dlhšieho ako jeden bajt. Preto som si vytvoril triedu memory block, ktorá nielen kontroluje pohyb po pamäti ale taktiež to, či pri prístupe do pamäti nie je spôsobený prístup mimo povolenú časť. Pri nepovolenom prístupe vyhodí táto trieda výnimku typu memory_error. Vtedy hociktorá trieda, ktorá používa túto triedu vie, že pristupuje kam nemá a tým pádom nenastane Segmentation fault. Ak teda program zachytí výnimku memory_error, berie paket ako poškodený, ignoruje ho a prejde na ďalší, ak je nejaký dostupný.

4.6 Ostatné triedy a funkcia main

Trieda dns_packet_capture využíva knižnicu libpcap pre získavanie paketov a triedu packet_parser pre parsovanie paketov. V triede utils su nejaké pomocné metódy napríklad na parsovanie doménového mena v štítkovom (label) formáte, ako je popísané v RFC1035 v sekcii 4.1.4. Trieda dns_export nastaví triedu dns_packet_capture a ak je nastavené sledovanie rozhrania, spustia sa pomocné vlákna. Funkcia main je v súbore main.cpp a nastavuje triedu dns_export podľa informácií získaných z parametrov pomocou triedy argument_parser. [4]

5 Návod na použitie programu

Pred spustením treba program preložiť príkazom make. Parameter -h vypíše nápovedu a program sa ukončí. Tento parameter sa neda kombinovať s inými. Pre čítanie súboru obsahujúceho pakety je treba použiť parameter -r, po ktorom nasleduje cesta k súboru s paketmi. Pre zaznamenávanie sieťovej prevádzky z rozhrania je nutné spustiť program s parametrom -i, za ktorým nasleduje názov rozhrania. Ak sa majú štatistiky posielať na Syslog server, musí byť nastavený parameter -s a za ním nasledovať doménove meno, IPv4 alebo IPv6 adresa tohto serveru. Ak nie je nastavený, štatistiky sa vypisujú na štandardný výstup. Interval odosielania štatistík sa nastavuje parametrom -t a číslo, ktoré za ním nasleduje, vyjadruje čas v sekundách. Ak nie je nastavený čas, je nastavený na 60 sekúnd. Nie je možné naraz používať parametre -i a -r a tak isto nie je možné použiť naraz -r a -t.

Literatúra

[1] Gerhards, R.: RFC 5424: The Syslog protocol. [online], [vid. 2018-17-11]. Dostupné z: https://tools.ietf.org/html/rfc5424

[2] Information Sciences Institute, U. o. S. C.: RFC 791: INTERNET PROTOCOL. [online], [vid. 2018-17-11]. Dostupné z: https://tools.ietf.org/html/rfc791

[3] Information Sciences Institute, U. o. S. C.: RFC 793: TRANSMISSION CONTROL PROTOCOL. [online], [vid. 2018-17-11]. Dostupné z: https://tools.ietf.org/html/rfc793

[4] Mockapetris, P.: RFC 1035: Domain names - implementation and specification. [online], [vid. 2018-17-11]. Dostupné z: https://tools.ietf.org/html/rfc1035

[5] Nyffenegger, R.: base64 (C++). [online], [vid. 2018-17-11]. Dostupné z: https://github.com/ReneNyffenegger/cpp-base64

[6] Postel, J.: RFC 768: User Datagram Protocol. [online], [vid. 2018-17-11]. Dostupné z: https://tools.ietf.org/html/rfc768

[7] R. Arends, e. a.: RFC 4034: Resource Records for the DNS Security Extensions. [online], [vid. 2018-17-11]. Dostupné z: https://tools.ietf.org/html/rfc4034

[8] S. Deering, R. H.: RFC 2460: Internet Protocol, Version 6 (IPv6). [online], [vid. 2018-17-11]. Dostupné z: https://tools.ietf.org/html/rfc2460

[9] S. Thomson, e. a.: RFC 3596: DNS Extensions to Support IP Version 6. [online], [vid. 2018-17-11]. Dostupné z: https://tools.ietf.org/html/rfc3596