

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ISA - Síťové aplikace a správa sítí

dokumentácia k 2.variante projektu

dns-export

Obsah

1	Úvod	1
2	Návrh aplikácie	1
2.1	Objektový návrh	1
2.2	Spracovanie paketov	1
2.3	Ukladanie štatistík	1
2.4	Odosielanie a výpis štatistík	1
3	Implementácia	2
3.1	DnsExport.cpp	2
3.2	DnsPacket.cpp	2
3.3	DnsRR.cpp	3
3.4	Helpers.cpp	3
4	Zaujímavé časti kódu	4
4.1	LINUX_SLL datalink	4
4.2	Netlačiteľné znaky	4
4.3	Odosielanie na Syslog	5
4.4	Získanie aktuálneho timestampu	5
4.5	Logovanie	6
4.6	Vytvorenie správ na odoslanie	6
5	Použitie	7
5.1	Čítanie pcap súboru	7
5.2	Odchytávanie na rozhraní	7
5.3	Zaslanie signálu SIGUSR1	7
5.4	Zobrazenie nápovedy	7
6	Prehľad naštudovanej literatúry	8
7	Literatúra	8

1 Úvod

Hlavným účelom aplikácie **dns-export** je spracovávanie dát protokolu *DNS (Domain Name System)* a odosielanie štatistík na centrálny logovací server. Aplikácia teda musí vedieť identifikovať a spracovať *paket* daného protokolu. Jednotlivé pakety môže buď čítať z *pcap* súboru alebo odchytať v reálnom čase zo sieťového rozhrania.

Zo spracovaných dát vytvára štatistiky, ktoré v časovom intervale posiela protokolom *Syslog* na centrálny logovací server alebo jednoducho vypisuje na štandardný výstup. Zbierané štatistiky sú:

- **domain-name** - doménové meno (napr. **nic.cz**) z *DNS query* (dotaz na DNS server)
- **rr-type** - typ DNS záznamu vrátený v odpovedi (napr. **A**, **AAAA**)
- **rr-answer** - všetky položky vrátené v odpovedi (napr. **IPv4 adresa**, **IPv6 adresa**)
- **count** - počet výskytov danej kombinácie **domain-name rr-type rr-answer**

2 Návrh aplikácie

2.1 Objektový návrh

Svoju aplikáciu som sa rozhodol naprogramovať objektovo. Dodržiaval som štruktúru jedna *trieda* na jeden zdrojový súbor. Každá trieda má svoju významom vyhradenú funkciu.

2.2 Spracovanie paketov

Aplikácia spracuje vstupné argumenty a po ich validácii sa buď spustí spracovanie *pcap* súboru, alebo odchytať pakety zo sieťového rozhrania. Pre každý spracovávaný paket sa vytvorí objekt držiaci dáta paketu.

V tomto objekte sa najprv spracujú jednotlivé vrstvy paketu okrem aplikačnej. Ak sa zistí, že paket obsahuje odpovede na DNS dotazy, vytvorí sa pre každú odpoveď objekt, ktorý tieto odpovede spracováva.

2.3 Ukladanie štatistík

Po úspešnom spracovaní odpovedí sa každá odpoveď pridá do objektu, ktorý obsahuje kolekciu všetkých nazbieraných štatistík. Pomocou metódy sa zistí, či sa daná štatistická informácia v objekte už nachádza a v tom prípade sa len inkrementuje počítadlo.

2.4 Odosielanie a výpis štatistík

Po prečítaní z *pcap* súboru sa štatistiky uložené v objekte odošlú na server. V prípade, že server nebol zadaný sa štatistiky vypíšu na štandardný výstup. V prípade živého odchytať sa pred jeho spustením vytvorí separátne vlákno, ktoré bude v zadanom časovom intervale odosielať všetky doposiaľ nazbierané štatistiky. Okrem toho bude aplikácia pri odchytení signálu **SIGUSR1** volať funkciu na výpis štatistík na štandardný výstup.

3 Implementácia

3.1 DnsExport.cpp

Po spustení aplikácie je vytvorený objekt tejto triedy a zavolaná funkcia **Main()**. Nastaví sa odchytyvanie signálov a na základne vstupných argumentov načítaných pomocou vstavanej funkcie **getopt()**¹ sa rozhodne, či sa spustí funkcia **sniffInterface()** alebo **sniffFile()**.

V prvom prípade sa pred započatím odchytyvania spustí funkcia **sendingLoop()**, ktorá sa v separátnom vlákne stará o pravidelné odosielanie na Syslog server. Pre odoslanie volá funkcia **sendStats()**, ktorá si volá pomocné funkcie na vytvorenie správ ako napr.: **getMessages()** alebo **getFormattedTime()** a následne *UDP* protokolom odosiela správy. Je podporovaná IPv4 aj IPv6 adresa Syslog serveru. Správa nie je dlhšia ako 1 kB a jednotlivé štatistiky sú oddelené znakom |. Aplikácia odosiela v Syslog správe *hostname*, pretože nepredpokladá existenciu *FQDN* ani statickej IP adresy. Ak neboli odchytené žiadne validné DNS pakety, odosiela sa správa s informáciou o nulovom počte DNS paketov.

V druhom prípade ak nebol zadaný Syslog server, je po prečítaní súboru zavolaná funkcia **getMessages()**, ktorá štatistiky vypíše na štandardný výstup a aplikácia končí. Rovnaká funkcia je volaná aj pri prijatí signálu **SIGUSR1**.

O všetky akcie ohľadom odchytyvania paketov sa stará knižnica **libpcap**². Pre každý paket je volaná *callback* funkcia **pcapHandler()**. Tam je vytvorený objekt triedy **DnsPacket** a zavolaná funkcia **Parse()**.

Ak boli v pakete nájdené validné DNS odpovede - *answers*, je zavolaná funkcia **addRecords()**, ktorá do globálnej premennej **recordList** pridá nový objekt triedy **DnsRecord**, ak sa daná štatistika v liste ešte nenachádza, alebo len zvýši počítadlo u nájdenej štatistiky.

3.2 DnsPacket.cpp

Vo funkcii **Parse()** je najprv zavolaná funkcia **getTransProt()**. Tá parsuje vrstvu sieťového rozhrania (podporované sú *ethernet* a pseudo protokol *linux sll*, ktorý sa tvorí napríklad pri počúvaní na všetkých rozhraniach) a sieťovú vrstvu (podporované sú IPv4 a IPv6). Následne zistí transportný protokol (podporované typy sú UDP a TCP). Pri parsovaní sú využívané štruktúry z linuxových hlavičkových súborov **netinet**. Pri zistení veľkosti TCP hlavičky som sa inšpiroval vláknami z fóra stackoverflow.com³ a ⁴ kvôli kompatibilitite. Ak sa zistí, že je paket fragmentovaný, funkcia končí a paket je zahodený, pretože aplikácie nepodporuje skladanie fragmentov.

Parsovanie DNS hlavičky začína vo funkcii **dnsParse()**. Tam prebehnú validácie, či ide skutočne o DNS paket. *Query* pakety sú ignorované, aplikácia pracuje len s *answers*. Ak sú nájdené *answers*, je zavolaná funkcia **parseRRSet()**, kde je hneď na začiatku zavolaná funkcia **skipQuestion()**, ktorá skočí na pozíciu prvej *answer*.

Pre každú *answer* je vytvorený objekt triedy **DnsRR**, ktorý spracuje dáta a nastaví pozíciu na ďalšiu *answer*. Ak boli získané nejaké dáta, objekt je uložený do listu **Answers**.

¹<http://man7.org/linux/man-pages/man3/getopt.3.html>

²<https://www.tcpdump.org/pcap.html>

³<https://stackoverflow.com/questions/6639799/calculate-size-and-start-of-tcp-packet-data-excluding-header>

⁴https://stackoverflow.com/questions/21893601/no-member-th-seq-in-struct-tcp_hdr-working-with-pcap

3.3 DnsRR.cpp

Každá answer začína položkou *domain-name*, ktorá sa získa funkciou **readDomainName()**. Pri komprimovanej položke je funkcia volaná rekurzívne maximálne 10-krát, aby sa zamedzilo prípadným cyklom. Ak sa podarí získať *domain-name*, nasleduje získanie *rr-type* a dĺžky dát. Vo funkcii **applyParser()** je **switch**, v ktorom sa na základe typu záznamu rozhodne ktorý z parserov⁵ pre dáta použiť. Aplikácia podporuje spracovanie nasledujúcich typov DNS záznamov: **A**, **AAAA**, **CNAME**, **MX**, **NS**, **SOA**, **TXT**, **SPF**, **PTR**, **DNSKEY**, **DS**, **RRSIG** a **NSEC**. Okrem zavolania konkrétneho parsera sa typ uloží ako *string* hodnota pre jednoduchšie ukladanie štatistík. Ak sa podarilo získať dáta, je vrátená pozícia za odpoveďou a aplikácia pokračuje parsovaním ďalšej answer.

3.4 Helpers.cpp

Táto trieda obsahuje makrá pre logovanie - **LOGGING** a ukončovanie aplikácie v prípade problému - **ERR_RET**.

Funkcia **ToHex()** prevedie netlačiteľný znak do hexadecimálneho tvaru, napr.: 0x01. Je používaná ak sa napríklad v *domain-name* alebo **TXT** zázname objaví netlačiteľný znak. Druhá varianta prevedie 16-bitový integer do hexadecimálneho tvaru a používa sa pri parsovaní záznamov **DNSKEY** a **DS**.

Funkcia **Base64Encode()** je využívaná parseroch pre záznamy typu **DNSKEY**, **DS** a **RRSIG**. Kóduje bajtové pole ako base64 string, ktorý sa zobrazí v štatistikách. Funkcia je dostupná na githube⁶ pod *Zlib* licenciou.

⁵Konkrétnejší popis jednotlivých parserov je v komentároch v zdrojovom kóde. Veľmi špecifický je záznam typu **SPF**. V súčasnosti sa posielajú iba záznamy typu **TXT** ktoré sa dajú na základe určitých pravidiel označiť ako **SPF**. Toto aplikácia nezisťuje a každý **TXT** záznam označí typom **TXT**. Keby náhodou prišiel záznam typu **SPF**, je parsovaný rovnako ako klasický **TXT** záznam.

⁶<https://github.com/ReneNyffenegger/cpp-base64/blob/master/base64.cpp>

4 Zaujímavé časti kódu

4.1 LINUX_SLL datalink

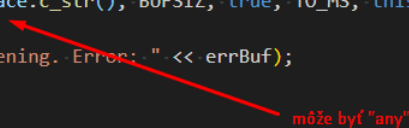
Pri odchyťovaní na všetkých rozhraniach (interface=any), nemajú pakety ethernet hlavičku ale tzv. *"linux-cooked capture header"*. To sa dá zistiť pomocou funkcie `pcap_datalink()`. Ak ide o `LINUX_SLL_DATALINK`, hlavička je parsovaná iným spôsobom ako ethernet:

```
// Open device for sniffing. TO_MS = 100 ms is value used by Wireshark
if ((myPcap = pcap_open_live(parameters.interface.c_str(), BUFSIZ, true, TO_MS, this->errBuf)) == NULL)
{
    ... ERR_RET("Unable to open interface for listening. Error: " << errBuf);
}

datalink = pcap_datalink(myPcap);

if (this->datalink == LINUX_SLL_DATALINK)
{
    ... ethType = ntohs(*(uint16_t*)(packet + 14));
    ... this->position += 16;
}
else if (this->datalink == ETHERNET_DATALINK)
{
    ... // Read the ethernet header
    ... eptr = (struct ether_header*)packet;
    ... ethType = ntohs(eptr->ether_type);
    ... this->position += sizeof(struct ether_header);
}
else
{
    ... return false; ... // Unsupported
}

// Check ethernet type
switch (ethType)
```



4.2 Netlačiteľné znaky

Ak sa v DNS answer (konkrétne v domain-name alebo v dátach TXT záznamu) objaví netlačiteľný znak, je vypísaný v hexadecimálnom tvare s prefixom 0x. Touto funkciou je prevádzaná aj bitová mapa v NSEC zázname. Pre 16 bitové čísla existuje druhá verzia funkcie **ToHex**, používaná pri záznamoch typu DNSKEY a DS:

```
// Symbolize char as hex number
static string ToHex(uint8_t ch)
{
    ... stringstream ss;
    ... ss << "0x" << setfill('0') << setw(2) << hex << (int)ch;
    ... return ss.str();
}

// Symbolize uint16_t as integer
static string ToHex(uint16_t ch)
{
    ... stringstream ss;
    ... ss << "0x" << setfill('0') << setw(4) << hex << (int)ch;
    ... return ss.str();
}
```

4.3 Odosielanie na Syslog

Pri spustení vlákna na odosielanie sa najprv vyčká 20 ms a až potom sa spustí cyklus na odosielanie. Pretože aplikácia využíva parameter `to_ms` vo funkcii `pcap_open_live()`, chceme aby sa stihli vytvoriť všetky štatistiky pred odoslaním. Kód je inšpirovaný vláknom na stránke [cplusplus.com](http://www.cplusplus.com/forum/beginner/91449/)⁷.

```
// Thread sending stats to syslog server
static void* sendingLoop(void* time)
{
    // Initial sleep -- waiting for pcap to start and parse something
    chrono::milliseconds openLiveT0(20); // By given parameter
    this_thread::sleep_for(openLiveT0);

    // http://www.cplusplus.com/forum/beginner/91449/
    chrono::seconds interval*((int*)time); // By given parameter
    while (true)
    {
        this_thread::sleep_for(interval);
        sendStats();
    }
}
```

4.4 Získanie aktuálneho timestampu

Syslog správa potrebuje timestamp v špecifickom tvare. Kód je inšpirovaný vláknom na stránke [codereview.stackexchange.com](https://codereview.stackexchange.com/questions/11921/getting-current-time-with-milliseconds)⁸. Aplikácia odosiela čas v *UTC*.

```
// Returns the local date/time formatted as 2014-03-19 11:11:52
// https://codereview.stackexchange.com/questions/11921/getting-current-time-with-milliseconds
static string getFormattedTime()
{
    timeval curTime;
    gettimeofday(&curTime, NULL);
    int milli = curTime.tv_usec / 1000;

    char buffer[80];
    strftime(buffer, 80, "%Y-%m-%dT%H:%M:%S", gmtime(&curTime.tv_sec));

    char currentTime[84] = "";
    sprintf(currentTime, "%s.%03dZ", buffer, milli);

    return currentTime;
}
```

⁷<http://www.cplusplus.com/forum/beginner/91449/>

⁸<https://codereview.stackexchange.com/questions/11921/getting-current-time-with-milliseconds>

4.5 Logovanie

Zdrojový kód obsahuje *macro* na logovanie, ktoré veľmi pomáha pri debugovaní. Logovanie sa dá jednoducho zapnúť a výpnúť zmenou hodnoty premennej **VERBOSE**..

```
#define VERBOSE 0.....//If 1 logging is active

//Macro for logging data when debugging
#define LOGGING(message) \
....if (VERBOSE == 1)....\
....{\
....cerr << yellow << "LOG: " << ... \
....message << reset << endl;.....\
....}
```

4.6 Vytvorenie správ na odoslanie

Aplikácia vytvára správy s dĺžkou nepresahujúcou 1KB. Na Syslog server je v jednej správe posielaných viac štatistík oddelených znakom |. V prípade, že je list so štatistikami prázdny sa odošle reťazec "No DNS packets found".:

```
// Return statistics formatted as syslog messages
static list<string> getMessages(string hostname)
{
....list<string> messages;
....string message;
....list<DnsRecord>::iterator it;
....list<DnsRecord> listToSend = recordList;.....// Create copy (safer multithreading)

....// Create list of all messages
....for (it = listToSend.begin(); it != listToSend.end(); it++)
....{
........string record = "." | "." + it->GetString();

........// We can add another record to one message if limit wasn't exceeded
........if ((record + message).size() > MESSAGE_SIZE)
........{
............message.erase(0, 3);.....// Remove leading |
............message = "<134>1 " + getFormattedTime() + "." + hostname + " dns-export" + "-----" + message;
............messages.push_back(message);
............message = "";
............}
............message += record;
........}
....}

....// At the end add last message to the list if not ""
....if (message.compare(""))
....{
........message.erase(0, 3);.....// Remove leading |
........message = "<134>1 " + getFormattedTime() + "." + hostname + " dns-export" + "-----" + message;
........messages.push_back(message);
....}
....else if (messages.size() == 0)
....{
........message = "No DNS packets found";
........messages.push_back(message);
....}
....return messages;
}
```


5 Použitie

5.1 Čítanie pcap súboru

Aplikácia sa spúšťa s argumentmi **-r** a **-s**:

```
xnerec00@d01-206b:/mnt/c/Users/tomn/Documents/git/FIT/5.sem/isa/src$ ./dns-export -r sample/cname.pcap -s 192.168.139.3
```

V tomto prípade sa na **stdout** nič nevypíše, štatistiky sú odoslané na Syslog server, napr.:

```
Nov 16 22:52:35 d01-206b dns-export solid.preyproject.com CNAME control.preyproject.com 2 i control.preyproject.com A 107.178.244.155 1 i control.preyproject.com AAAA 2600:1901:0:9af5:0:0:0:0 1
```

V prípade, že nie je zadán Syslog server sa štatistiky vypíšu na **stdout**:

```
xnerec00@d01-206b:/mnt/c/Users/tomn/Documents/git/FIT/5.sem/isa/src$ ./dns-export -r sample/cname.pcap
solid.preyproject.com CNAME control.preyproject.com 2
control.preyproject.com A 107.178.244.155 1
control.preyproject.com AAAA 2600:1901:0:9af5:0:0:0:0 1
```

5.2 Odchytyvanie na rozhraní

Aplikácia sa spúšťa s argumentmi **-i**, **-s** a **-t**:

```
xnerec00@d01-206b:/mnt/c/Users/tomn/Documents/git/FIT/5.sem/isa/src$ ./dns-export -i enp0s8 -s 192.168.139.3 -t 45
```

V tomto prípade budú štatistiky odosielané na Syslog server každých 45 sekúnd. Ak nie je zadán argument **-t**, štatistiky sú odosielané každých 60 sekúnd. Ak nie je zadán argument **-s**, štatistiky sa neodosielaajú.

5.3 Zaslanie signálu SIGUSR1

Pri odchytyvaní na rozhraní sa štatistiky vypíšu na **stdout** pri zaslaní signálu SIGUSR1:

```
root      1869  0.0  0.0 18712   776 tty1    R+   23:28   0:00 ./dns-export -i any
root      1874  0.0  0.1 155296  1748 pts/0   R+   23:28   0:00 ps aux
[root@isa ~]# kill -10 1869
```

```
[root@isa src]# ./dns-export -i any
google.com A 172.217.23.206 1
```

5.4 Zobrazenie nápovedy

Ak je zadán argument **-h**, na **stdout** je vypísaná nápoveda a program skončí. Nápoveda je vypísaná aj v prípade nevalidnej kombinácie argumentov.:

```
xnerec00@d01-206b:/mnt/c/Users/tomn/Documents/git/FIT/5.sem/isa/src$ ./dns-export -h
Usage: dns-export [-r file.pcap] [-i interface] [-s syslog-server] [-t seconds]
file.pcap - file to sniff
interface - interface to sniff (or ANY for all interfaces)
syslog-server - Syslog server where the stats are sent
seconds - time period of sending stats to Syslog server
```

6 Prehľad naštudovanej literatúry

Práca s knižnicou libpcap a parsovanie ethernetovej hlavičky a transportných protokolov - príklady k predmetu ISA (`sniff.c`, `read-pcap.c`) a rôzne voľne dostupné zdroje na internete.

Syslog protokol - prednáška `isa-logovani.pdf` a RFC 5424

Parsovanie DNS hlavičky a jednotlivých typov záznamov - RFC 1035

Parsovanie záznamov pre DNSSEC - RFC 4034

Parsovanie hlavičky pri LINKTYPE_LINUX_SLL - `tcpdump.org`

Špecifickosť SPF záznamu - `support.dnssimple.com`

7 Literatúra

<https://tools.ietf.org/html/rfc5424>

<https://tools.ietf.org/html/rfc1035>

<https://www.ietf.org/rfc/rfc4034.txt>

http://www.tcpdump.org/linktypes/LINKTYPE_LINUX_SLL.html

<https://support.dnssimple.com/articles/spf-record/>

https://en.wikipedia.org/wiki/List_of_DNS_record_types

<https://solarianprogrammer.com/2011/12/16/cpp-11-thread-tutorial/>

<https://www.rhyous.com/2011/11/13/how-to-read-a-pcap-file-from-wireshark-with-c/>

<https://www.tcpdump.org/pcap.html>

https://www.tcpdump.org/manpages/pcap_open_live.3pcap.html