VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Generování NetFlow dat ze zachycené síťové komunikace
Síťové aplikace a správa sítí – projekt

Lukáš Vincenc (xvince01) 31. října 2022

Obsah

1 Úvod

NetFlow má několik využití, ale jeho hlavním je monitorování toku dat po síti. Skládá se z několika částí. Těmi jsou exporter, collector a aplikace k analýze získaných flow dat.[1]. V rámci tohoto projektu jsme měli za úkol implementovat exporter, který odešle packety zpracované do flows collectoru.

2 Fungování exporteru

Exporter jako takový bere ze vstupu packety a slučuje je do tzv. flows podle několika vlastností packetu. Může se jednat o aktivní komunikaci v reálném čase, ale v našem projektu pracujeme pouze s předem vygenerovanými soubory ve formátu `.pcap`. V těch mohou být uloženy různé packety z již proběhlé komunikace. Do jednoho flow patří packety, pokud mají stejnou:

- Zdrojovou IP adresu
- Cílovou IP adresu
- Zdrojový port
- Cílový port
- Protokol
- Type of Service

Formát flow záznamu se poté může lišit podle toho, jaká verze je používaná. V mé implementaci podporuji pouze NetFlow v5. Každý záznam se poté skládá z hlavičky a vlastního těla záznamu. Na následujících obrázcích můžeme vidět jejich strukturu popsanou na veřejných stránkách webu Cisco[2].

[1] https://en.wikipedia.org/wiki/NetFlow

[2] https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html#wp1003394

Bytes	Contents	Description
0-1	version	NetFlow export format version number
2-3	count	Number of flows exported in this packet (1-30)
4-7	SysUptime	Current time in milliseconds since the export device booted
8-11	unix_secs	Current count of seconds since 0000 UTC 1970
12-15	unix_nsecs	Residual nanoseconds since 0000 UTC 1970
16-19	flow_sequence	Sequence counter of total flows seen
20	engine_type	Type of flow-switching engine
21	engine_id	Slot number of the flow-switching engine
22-23	sampling_interval	First two bits hold the sampling mode; remaining 14 bits hold value of sampling interval

Tabulka 1: Hlavička flow záznamu

Většinu hodnot dokážeme z packetů získat, ale ne všechny. Ty, co neznáme proto nastavuji na výchozí nulu. U hlavičky neznáme `engine_type`, `engine_id` a `sampling_interval`.

Bytes	Contents	Description
0-3	srcaddr	Source IP address
4-7	dstaddr	Destination IP address
8-11	nexthop	IP address of next hop router
12-13	input	SNMP index of input interface
14-15	output	SNMP index of output interface
16-19	dPkts	Packets in the flow
20-23	dOctets	Total number of Layer 3 bytes in the packets of the flow
24-27	First	SysUptime at start of flow
28-31	Last	SysUptime at the time the last packet of the flow was received
32-33	srcport	TCP/UDP source port number or equivalent
34-35	dstport	TCP/UDP destination port number or equivalent
36	pad1	Unused (zero) bytes
37	tcp_flags	Cumulative OR of TCP flags
38	prot	IP protocol type (for example, TCP = 6; UDP = 17)
39	tos	IP type of service (ToS)
40-41	src_as	Autonomous system number of the source, either origin or peer
42-43	dst_as	Autonomous system number of the destination, either origin or peer
44	src_mask	Source address prefix mask bits
45	dst_mask	Destination address prefix mask bits
46-47	pad2	Unused (zero) bytes

Tabulka 2: Flow záznam

V samotném záznamu neznáme hodnot více, například SNMP indexy nebo masky. I zde je všechny nastavíme na nulu.

Každý příchozí packet je zařazen k patřičnému flow. Ten je poté aktualizován na základě nového packetu. Exporter má nastavitelné hodnoty dvou časovačů – aktivní a neaktivní. Poté, co přijde nový packet, jsou podle nich všechny flows zkontrolované. Pokud do flow nepřišel nový packet po delší dobu, než je určena v neaktivním časovači, je flow exportován. Stejně tak pokud je flow aktivní déle, než je povoleno aktivním časovačem. Tyto kontroly se odehrají před přiřazením packetu do některého z flows. Pokud neexistuje aktivní flow, do kterého by bylo možné nově příchozí packet zařadit, je vytvořen nový flow.

Po přečtení všech packetů z `.pcap` souboru je zbytek flows vyexportován. Každý exportovaný flow je odeslán na collector, jehož adresa je nastavitelná přes argument programu. V našem případě pomocí UDP packetů.

3 Implementace

Jako implementační jazyk jsem si zvolil C++. Primární knihovny, které jsem využil, jsou `pcap` pro čtení ze vstupního souboru a několik `netinet` knihoven pro práci s hlavičkami packetů. Pro čtení argumentů jsem využil knihovnu `getopt`.

Podle manuálů k `pcap` knihovně[3] jsem vytvořil funkci, která otevře vstupní soubor a pro každý packet zavolá stejnou funkci – `callback`.

```
char errbuf[PCAP_ERRBUF_SIZE];
pcap_t *handle;
struct bpf_program fp;
char filter_exp[] = "icmp or tcp or udp";
bpf_u_int32 net = 0;

handle = pcap_open_offline(arguments.file, errbuf);

if (!handle)
    // error_exit

if (pcap_compile(handle, &fp, filter_exp, 0, net) == -1)
    // error_exit

if (pcap_setfilter(handle, &fp) == -1)
    // error_exit

pcap_loop(handle, 0, (pcap_handler)callback, nullptr);
export_remaining_flows_in_map();

pcap_close(handle);
```

Pomocí `pcap_open_offline` otevřeme soubor, který je předán z argumentů. Jelikož přijímáme podle zadání pouze ICMP, TCP a UDP packety, připravil jsem pro ně ve `filter_exp` stringu filtr. Ten se poté přes `pcap_compile` zkompiluje a následně v `pcap_setfilter` nastaví. Nakonec `pcap_loop` zavolá pro každý packet `callback` funkci.

V této funkci extrahuju z hlavičky časové údaje daného packetu a volám navazující funkci, která určí, o který typ protokolu se jedná. Následně se vytvoří klíč, podle kterého se pak packet ukládá do mapy flows. Jak

[3] https://www.tcpdump.org/manpages/

už jsem dříve zmínil, před vložením packetu se kontrolují jednotlivé časovače. K tomu jsem si připravil funkci, která projde celou mapu a exportuje flows splňující podmínky.

```
Iterator oldest = flow_cache.begin();

for (Iterator it = oldest; it != flow_cache.end();) {
    // calculations for active and inactive timers
    if (active_invalid || inactive_invalid) {
        export_flow(it->second);
        it = flow_cache.erase(it);
    } else {
        if (oldest->first > it->first) oldest = it;
        ++it;
    }
}

if (arguments.count <= flow_cache.size()) {
    export_flow(oldest->second);
    flow_cache.erase(oldest);
}
```

Počty pro proměnné `active_invalid` a `inactive_invalid` vypadají takto:

```
active_invalid = arguments.active_timer * 1000 < sys_uptime &&
it->second.first < sys_uptime - (arguments.active_timer * 1000);

inactive_invalid = arguments.inactive_timer * 1000 < sys_uptime &&
it->second.last < sys_uptime - (arguments.inactive_timer * 1000);
```

Jelikož jsou časovače v sekundách, musím je přepočítávat na stejnou jednotku, jako `sys_uptime`, tedy milisekundy. `sys_uptime` znázorňuje systémový čas právě zpracovávaného packetu. Dále `it` je hodnota typu `Iterator`, který představuje položku mapy. Každá taková položka má `first` a `second` property, kde `first` je klíč a `second` je samotný záznam. Proto přes `second` mohu přistoupit k časovačům jednotlivých flows.

4 Použití programu

Ke kódu je připravený soubor `Makefile`. Stačí tedy otevřít terminálové okno v dané složce projektu a spustit překlad pomocí příkazu `make`. Tímto se vytvoří spustitelný soubor, který lze poté spustit s několika argumenty.

```
$ ./flow [-f <file>] [-c <netflow_collector>[:port]] [-a <active_timer>]
[-i <inactive_timer>] [-m <count>]
```

- `file` – Vstupní soubor ve formátu `.pcap`. Pokud není specifikovaný, čte z standartního vstupu
- `netflow_collector` – Adresa, na kterou se exportují flows. Výchozí adresa je 127.0.0.1:2055.
- `port` – K adrese lze připojit také port. Pokud není vybraný, je 0.
- `active_timer` – Aktivní časovač v sekundách, v základu 60 vteřin.
- `inactive_timer` – Neaktivní časovač v sekundách, v základu 10 vteřin.
- `count` – Počet flows, které mohou být zároveň uložené v paměti. Pokud je tato hodnota přesažena, exportuje se nejstarší záznam. Výchozí hodnota je 1024.

Pokud se neobjeví žádné chybové hlášení, export proběhl úspěšně.