

Сопроводительная документация

Содержание

- [Описание](#)
- [Сборка](#)
 - [Сборка исходных файлов](#)
 - [Сборка debian пакета](#)
- [Запуск сенсора](#)
- [Максимальная нагрузка программы](#)
 - [Демонстрация программы под нагрузкой](#)
 - [Определение максимальной нагрузки](#)
- [Авторство и лицензия](#)
 - [Автор](#)
 - [Лицензия](#)
 - [Сторонний исходный код](#)

Описание

В данном проекте реализуется Netflow-сенсор, поддерживающий Netflow export protocol версии 9.

Сенсор ведёт учёт трафика на одном указанном интерфейсе и отправляет собранные потоки в коллектор. Учёт ведётся только по пакетам, содержащим IPv4 и протоколы ICMP, TCP и UDP.

В данном Netflow-сенсоре поток определяется следующими полями:

- IP адрес источника;
- IP адрес получателя;
- Для TCP/UDP пакетов:
 - порт источника;
 - порт получателя;
- Для ICMP пакетов: ICMP код и тип;
- IP Protocol;
- IP ToS.

При старте программы коллектору посылается пакет, содержащий Template Flowset со следующими полями (см. [RFC 3954, 8. Field Type Definitions](#) и [NetFlow Version 9 Flow-Record Format, Table 6](#)):

- IN_BYTES,
- IN_PKTS,
- FLOWS,
- PROTOCOL,
- SRC_TOS,
- TCP_FLAGS,
- L4_SRC_PORT,
- IPV4_SRC_ADDR,
- INPUT_SNMP,
- L4_DST_PORT,
- IPV4_DST_ADDR,
- LAST_SWITCHED,
- FIRST_SWITCHED,
- ICMP_TYPE,
- FLOW_ACTIVE_TIMEOUT,
- FLOW_INACTIVE_TIMEOUT,
- IPV4_IDENT,
- IN_SRC_MAC,
- IN_DST_MAC,
- IF_NAME.

Примечание: этот пакет посылается только один раз, поэтому необходимо, чтобы коллектор был запущен до запуска сенсора.

После этого отправляются пакеты с Data Flowset, в которых передаются несколько потоков. Потоки экспортируются согласно 2 (экспорт неактивных потоков) и 3 (экспорт долгосрочных потоков) пунктам из [RFC 3954](#), раздел 3.2. Flow Expiration. Соответствующие таймауты равны 15 и 30 секундам.

Сборка

Сборка исходных файлов

В этом проекте используется система сборки GNU Autotools, поэтому необходимо установить пакет autoconf:

```
$ apt-get install autoconf
```

Для сборки необходимо выполнить:

```
$ autoreconf --install
$ mkdir build && cd build
$ ../configure
$ make
```

В результате выполнения команд будет создан исполняемый файл nfexp в /netflow-expoter/build/src.

Сборка debian пакета

TBD

Запуск сенсора

Для прослушивания трафика на интерфейсе используется [Raw Socket](#), требующий привилегии CAP_NET_RAW. Чтобы её установить, необходимо выполнить команду:

```
$ sudo setcap cap_net_raw+ep ./nfexp
```

Программа nfexp имеет два обязательных аргумента:

- имя интерфейса,
- адрес коллектора в формате IP:порт.

Команда для запуска сенсора:

```
$ ./nfexp -i lo -c 127.0.0.2:9995
```

Максимальная нагрузка программы

Демонстрация программы под нагрузкой

Для демонстрации программы будем использовать следующие утилиты:

- `nfcapd` - выступает в роли коллектора;
- `tcpreplay` - воспроизводит трафик из pcap файла.

Также создадим dummy интерфейс, чтобы изолировать процесс считывания пакетов:

```
$ sudo ip link add name dum0 type dummy
$ sudo address add 10.0.0.1/32 dev dum0
$ sudo link set dum0 promisc on
$ sudo link set dum0 up
```

Отладочный трафик подаётся из файла [bigFlows.pcap](#). Для более наглядного отображения количества отправленных/принятых пакетов оставим только записи с ipv4 пакетами, содержащими протоколы ICMP, TCP и UDP. Полученный отладочный набор пакетов находится в файле `test_pcap/bigFlows_ipv4only.pcap`. (Количество пакетов изменяется от изначальных 791615 до 790893.)

Для запуска демонстрации необходимо выполнить три команды из корневого каталога проекта.

1. Запустим коллектор по адресу 127.0.0.2:9995, который будет сохранять информацию о потоках в директории `test_pcap/`:

```
$ nfcapd -l test_pcap/ -b 127.0.0.2 -p 9995
Add extension: ...
...
Bound to IPv4 host/IP: 127.0.0.2, Port: 9995
Startup.
Init IPFIX: Max number of IPFIX tags: 68
```

2. Запустим сенсор:

```
$ build/src/nfexp -i dum0 -c 127.0.0.2:9995
```

Коллектор nfcapd при запуске сенсора должен вывести сообщения:

```
Process_v9: New exporter: SysID: 1, Domain: 127, IP:
127.0.0.1
```

```
Process_v9: [127] Add template 256
```

3. Отправим отладочный трафик на интерфейс:

```
$ sudo tcpreplay -i dum0 --topspeed
test_pcap/bigFlows_ipv4only.pcap
Actual: 790893 packets (355329063 bytes) sent in 1.15 seconds
Rated: 307493791.3 Bps, 2459.95 Mbps, 684421.04 pps
Statistics for network device: dum0
Successful packets:      790893
Failed packets:          0
Truncated packets:       0
Retried packets (ENOBUFS): 0
Retried packets (EAGAIN): 0
```

Когда tcpreplay завершит выполнение, необходимо подождать 15 секунд, чтобы сенсор экспортировал оставшиеся (неактивные) потоки. После этого можно завершить выполнение сенсора и коллектора.

Как видно по выводу tcpreplay, было отправлено 790893 пакетов (т.е. все пакеты из отладочного трафика) со скоростью 2459.95 Mbps.

Количество учтённых сенсором пакетов можно увидеть в сообщении, которое выводит коллектор при завершении работы:

```
^CIdent: 'none' Flows: 40273, Packets: 773969, Bytes: 337134367,
Sequence Errors: 0, Bad Packets: 0
Total ignored packets: 0
Terminating nfcapd.
```

либо с помощью команды nfdump:

```
$ ./nfexp -i lo -c 127.0.0.2:9995
...
Summary: total flows: 40273, total bytes: 337.1 M, total packets:
773969, avg bps: 1.4 G, avg pps: 390302, avg bpp: 435
```

Определение максимальной нагрузки

Процент потерянных пакетов определяем по формуле: $(1 - \text{получ/отправ}) * 100$.
Потеря пакетов в демонстрационной процедуре составила $(1 - 773969/790893) * 100 = 2.14\%$.

Для установления максимальной нагрузки эта процедура была проделана для других значений скоростей (путём вызова `tcrcplay` с опцией `--mbps N`), с небольшим количеством повторений. Данные в таблице примерные.

Speed, Mbps	Loss
2000	до 3.1 %
1000	до 1.4 %
500	до 0.6 %
250	до 0.2 %
100	до 0.04 %

Максимальная нагрузка, которую может выдержать данная реализация - до 100 мбит/с.

Авторство и лицензия

Автор

Copyright (c) 2022 Доленко Дмитрий <dolenko.dv@yandex.ru>

Лицензия

Исходный код распространяется под лицензией MIT (см. прилагаемый файл LICENSE или <http://www.opensource.org/licenses/mit-license.php>).

Сторонний исходный код

В проекте используется сторонний исходный код:

- `hash_funcions/lookup3`: `lookup3.c`, by Bob Jenkins, May 2006, Public Domain.
- `hash_funcions/murmur3`: by Austin Appleby and Peter Scott, Public Domain.