Brno University of Technology
Faculty of Information Technology

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Network Applications and Network Administration

2022/2023

Project documentation

Generování NetFlow dat ze zachycené síťové komunikace

Vladyslav Kovalets (xkoval21)

Brno, November 14, 2022

Project task

Create a NetFlow exporter that creates NetFlow records from captured network data in *pcap* format and then sends it to the collector.

Introduction

NetFlow is a network protocol system that collects active IP network traffic as it flows in or out of an interface. This creates the ability to see the network and is therefore critical to maintenance and security.

NetFlow v5

The NetFlow version specifies which packet data (or which packets) will be included in the statistics, and how individual statistics items will be arranged when exported to the collector. It consists of a header and a record. What each part contains can be found on the Cisco Systems website.

For our purposes, version 5 will be implemented. It only handles IPv4 network protocol packets. The statistics are transmitted to the collector in a UDP packet in the format *header+n*record*, where *n* has a maximum value of 30.

Implementation

To implement an analyzer of captured network communication in *PCAP* format, which creates NetFlow records and send them to the collector, I first divided the project into several tasks: Argument Parser, Packet Processing, Flow Creation and Export to Collector.

Time

All timing data needed to manage a scenario or calculate statistics will be taken from individual packages. That is, the analyzed communication won't take place in real time, but will be read from a file or from STDIN.

Thus, time will flow discretely, from value to value, based on the timestamp of the packet currently being processed.

Argument Parser

This part of the code was implemented very quickly with the *getopt_long()* function, which makes it easy to process command line arguments.

Further, all arguments are written to the structure *argv_options*, which initially stores the default settings.

Packet Processing

To implement this part of the code, first I had to capture network communication and thereby create a file in *PCAP* format. The tcpdump program helped me with this.

To work with these files, I use the *pcap.h* library. First I open the file with *pcap_open_offline()* function, and then I parse each packet using a loop with the *pcap_next()* function.

While parsing each packet, I skip IPv6 packets because NetFlow v5 doesn't support them. The remaining packets, I send to switch, where I determine which packet belongs to which protocol (TCP / UDP / ICMP).

The program should first check with each new packet if the export interval has expired and if some active threads should expire based on inactive timer. Only then will they be able to process the given packet and possibly include it in the flow, and not in reverse order. Having decided that this package doesn't belong to any of the active threads, but installs a new one, it is necessary to compare the number of all active threads with the maximum possible number.

Flow Creation

The program stores all packets that have the appropriate protocol in *std::list<>flows* and then sorts them into *std::map<>sorted_flows*.
The *std::map<>sorted_flows* is where I store the created flows and the packets are merged into flows according to the mentioned petition which will be the key.
The *sort_flows()* function is responsible for sorting, which works with the *std::list* and with the *std::map*.

Export to Collector

The *UDP_export()* function is responsible for this part. The protocol for sending NetFlow packets is UDP.
In order to send flows to the NetFlow collector, I use the address and port from the arguments. In the constructor of the object class I use to connect to the connector, I use the *gethostbyname()* function from the *netdb.h* library, which takes the hostname, queries the DNS server, and returns a structure that contains the desired IP address.

Usage

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

-f	Name of analyzed file (STDIN, if not specified).
-c	IP address or hostname of the NetFlow collector. Optionally also UDP port (127.0.0.1:2055, if not specified).
-a	Interval in seconds after which active records are exported to the collector (60 if not specified).
-i	Interval in seconds after which inactive records are exported to the collector (10 if not specified).
-m	Flow-cache size. When the maximum size is reached, the oldest entry in the cache is exported to the collector (1024, if not specified).

All parameters are taken as optional. If any of the parameters is not specified, the default value is used instead.

Example: *./flow -f input.pcap -c 192.168.0.1:2055*

Testing

Testing was done with the nfdump program, which allowed me to get a collector capable of accepting the NetFlow protocol nfcapd *-p <port> -b <ip> -l <folder>*. To make sure the server was running, I used the netstat *-n --udp --listen* command. Next, I launched the exporter, which sent the data and using ndfump *-r <file>* looked through the flows that I managed to send.
In the end, to compare the result, I used the softflowd program.

Applications

Information from the NetFlow protocol can be used in a number of applications:

- Network security analysis

Allows you to identify and classify denial of service attacks (DDOS attacks), detect traffic generated by viruses and Trojans in real time. By analyzing network behavior, you can quickly identify anomalies in the network and take measures to eliminate them.

- Accounting for used network resources

Provides accurate information about transmitted/received traffic. On the basis of which it is possible to issue invoices to users if the user does not use an unlimited package.

- Network monitoring

With the help of programs that process the accumulated data from NetFlow collectors, you can evaluate traffic on individual network devices, identify problems that arise in the network (overload or failure of some network devices, unauthorized actions on the network, etc.) and, accordingly, quickly make decisions to fix these problems.

- Application monitoring.

NetFlow data provides accurate information about which applications are used on the network at what time, to identify which applications are using the network the most. This can be used to schedule new services such as voice over IP (VoIP), IPTV, video on demand, and the like.

- User monitoring.

NetFlow protocol data allows you to collect detailed information on each network user, about what network resources they use at a particular point in time, what applications are used, and so on. This information can be used to effectively plan the network and allocate user access to network resources, determine the bandwidth they need, and detect potential security issues.

- Network development planning.

Allows you to minimize the total cost of network operations and at the same time increase network performance, its capabilities and reliability.

It also allows you to detect unwanted external traffic, monitor the quality of service (QOS) and analyze the consequences of introducing new network applications to the network.

References

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

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

https://linux.die.net/man/3/getopt_long

https://man7.org/linux/man-pages/man3/gethostbyname.3.html

https://www.tcpdump.org/manpages/tcpdump.1.html

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

https://manpages.ubuntu.com/manpages/trusty/man7/netdb.h.7posix.html

https://nfdump.sourceforge.net/

https://manpages.ubuntu.com/manpages/bionic/man8/softflowd.8.html

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

https://en.cppreference.com/w/cpp/container/map

https://manpages.ubuntu.com/manpages/bionic/man1/nfcapd.1.html

https://linux.die.net/man/8/netstat