DOI: 10.5604/01.3001.0010.4823

TRAFFIC ANALYSIS USING NETFLOW AND PYTHON

Vaclav Oujezsky, Tomas Horvath
Brno University of Technology, Department of Telecommunication

***Abstract**: This article presents an application that is used as NetFlow collector and analyzer. It is a console application created in Python language. A software analyzer detects and analyzes incoming NetFlow messages version 1 and 5 of devices that support them. The output file is a database of information and analysis of the overall UNIX time duration of reported traffic and analysis of NetFlow lifetime. The software is developed to work with Python version 3 and higher and is designed for the Windows operating system.*

Keywords: IP networks; Computer languages; Software tools.

ANALIZA RUCHU SIECIOWEGO Z WYKORZYSTANIEM NETFLOW I PYTHON

***Streszczenie**: W artykule przedstawiono aplikację używaną jako kolektor i analizator NetFlow. Jest to aplikacja konsoli utworzona w języku Python. Analizator oprogramowania wykrywa i analizuje przychodzące wiadomości NetFlow w wersji 1 i 5 dla urządzeń je obsługujących. Plik wyjściowy to baza danych informacji i analizy ogólnego czasu trwania zgłoszonego ruchu UNIX i analizy życia NetFlow. Oprogramowanie zostało opracowywane dla systemu operacyjnego Windows i języka Python wersja 3 lub wyższa.*

Słowa kluczowe: sieci IP; języki komputerowe; oprogramowanie narzędziowe

Introduction

NetFlow has been invented by Cisco Systems, Inc. company [1]. It is a very popular technique nowadays and it is also widely deployed. Another type is the Internet Protocol Flow Information Export (IPFIX) and it is an IETF (The Internet Engineering Task Force) [2] protocol. Both of them are used to export flow information from routers, probes and other network devices for security, accounting, and other purposes.

Typical NetFlow export datagram format for version 1, 5, 7, and 8 is shown in Fig. 1.

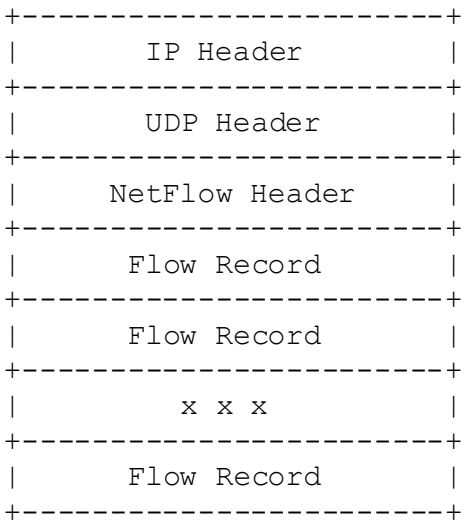

Fig. 1. NetFlow Export Datagram Format

The version 1 is rarely used nowadays. The version 5 adds Border Gateway Protocol (BGP) autonomous system information and flow sequence numbers. The version 7 adds support for Cisco Catalyst switches. When the Router-Based NetFlow Aggregation feature is enabled then the version 8 is used. The most recent version is 9 and supports template based extensible design [3].

Network devices send information about passing traffic using NetFlow to a collector. One example of such collector is Scrutinizer [4]. Collectors obtain information from network devices about duration, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) ports of a connection and so on.

Flows provide a continuous account of all network activity and detect attacks without signatures. It is possible to identify the certain types of network attacks and other incidents. This possibility depends on the quality of a collector. The flow-based analysis relies on used algorithms and behavior and provides zero-hour detection of attacks [5].

From this perspective, the deployed algorithms are very important for identifying the certain type of incidents. From this point of view, we concentrate on the possibility to develop own algorithms to detect malicious traffic. This development demands having our own application where we can test these algorithms.

1. The concept and functionality of the application

We choose Python language for the development. It is a scripting language, similar to Matlab. Python has a huge base of developers and it offers many packages for scientists as are Matplotlib, Numpy, Scipy, Panda etc.

The other very important requirement for us is the possibility to work with network integrated cards. This feature is also included in Python packages.

The name of the program is GDP [6]. Its concept is based on the possibility to process NetFlow messages version 1 and 5. Both types of messages (reports) are composed out of the header and flow body. The detailed composition is presented in [7]. As we designed the flows part, we took over the distribution of the bit stream from this source.

2. Terminal User Interface

Fig. 2 shows the terminal user interface of the GDP program. Generally, the program listens to incoming NetFlow reports on individually selected ports. The default port number `<4710>` can be changed in the configuration file `<config.txt>`. The entire socket consists out of all interfaces and addresses of a network device (computer). The part (1) shows the type of incoming NetFlow and how many flows a stream includes.

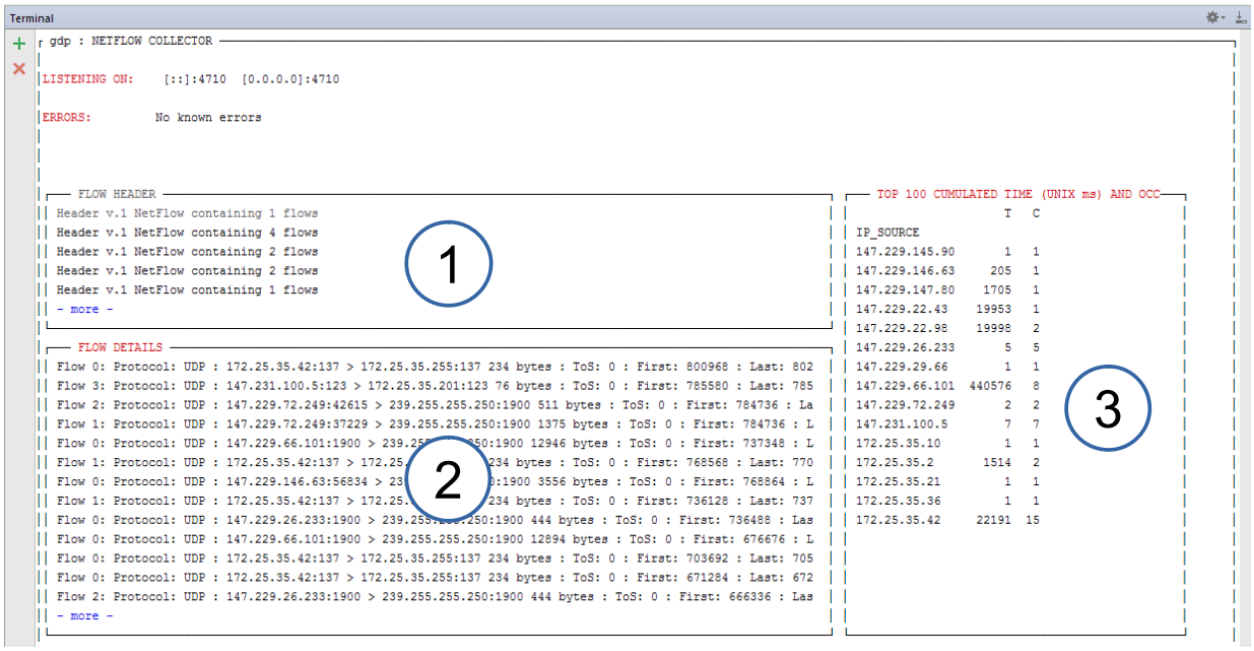

Fig. 2. GDP Terminal User Interface

Part (2) shows detailed information about each included flow, as are flow number, protocol, source IP address, destination IP address, Type of Service, first and last time in UNIX format. An analysis of traffic is shown in the last part (3). This analysis presents summary UNIX duration time for each IP source address (column T) and also how many times this connection was observed (column C).

All of this information is taken from a database file, in which all incoming traffic is written in a proper format. The string format is shown in Fig. 3. Items listed here are: ID, SYS_UPTIME, UNIXS, FIRST, LAST, IP_SOURCE, IP_DEST, PROTO, SOURCE_PROTO, DEST_PROTO and timestamp. Its names come from NetFlow. The program automatically stores information to the file `<dataset.sqlite3>`. The database file is deleted by default from the program's beginning. The preservation of historical data in the database can be changed with the configuration file. The value `<YES>` is necessary to change to the value `<NO>`.

The Terminal User Interface (TUI) is developed upon the npyscreen package [8]. We also had to use the threading package [9]. The program runs with two separate threads and with one general lock. First of all, the TUI starts after that the socket listener starts. It is necessary to run two independent loops of specified program parts. It is because socket listener is continuously listening to incoming frames and the TUI is doing a separate calculation at the same time.

SYS_UPTIME	UNIXS	FIRST	LAST	IP_SOURCE	IP_DEST	PROTO	OURCE_PROT(	DEST_PROTO	timestamp
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
245084	1474708790	217424	218984	172.25.35.42	172.25.35.255	17	137	137	1474708784....
245084	1474708790	177040	229052	147.229.66.101	239.255.255....	17	1900	1900	1474708784....
245084	1474708790	223580	223580	147.231.100.5	172.25.35.201	17	123	123	1474708785....
434084	1474708979	362656	422044	147.229.66.101	239.255.255....	17	1900	1900	1474708973....
434084	1474708979	387920	407872	147.229.22.43	239.255.255....	17	53206	1900	1474708973....
434084	1474708979	411840	413372	172.25.35.42	172.25.35.255	17	137	137	1474708973....
434084	1474708979	415572	415572	147.231.100.5	172.25.35.201	17	123	123	1474708974....
434084	1474708979	418388	418388	147.229.29.66	239.255.255....	17	49654	1900	1474708974....
448084	1474708993	422012	422012	147.229.26.233	239.255.255....	17	1900	1900	1474708987....
448084	1474708993	425452	425452	172.25.35.2	172.25.35.255	17	138	138	1474708987....

Fig. 3. DB Browser for SQLite – NetFlow

3. Flow analysis

Accumulated time duration analysis is implemented as a first type of individual analysis. The output of the analysis is presented in previous chapter. The value is taken from saved flows in the database. The value is calculated by <SysUptime> the last package of the flow was received minus <SysUptime> at the start of the flow.

Alg. 1. Python code of duration analysis

```
import warnings
import numpy as np
from database import Database as db

def survival_max_time_per_ip():
    with warnings.catch_warnings() as s:
        warnings.filterwarnings("error")

        try:
            f = db("./database/dataset.sqlite3")
            df = f.survival_read()
            daf = df.head(n=100)
            grouped = daf.groupby("IP_SOURCE")
            return "%s" % (grouped.aggregate(np.sum))
        except RuntimeWarning:
            return s
```

The next step of data processing is shown in Alg. 1. The function <read_sql_query> from panda package reads the data from the database and after that the data are stacked by <head>, where 100 inputs of IP addresses are read. The data are then grouped by IP source and the function <survival_max_time_per_ip()> returns the sum of aggregated values. In Alg. 2 is presented the function <survival_read()> mentioned above. It uses sqlite3 database import and sql panda import. As is shown, the sql reads previously saved information from the database.

Alg. 2. Python code of duration analysis

```
def survival_read(self, table="Flow", ):
   db = sqlite3.connect(self.database_name)
   c = db.cursor()

  # read by panda>
   sql = """SELECT IP_SOURCE AS "IP_SOURCE", IP_DEST,
(LAST - FIRST) + 1 AS "T", (LAST IS NOT NULL) AS "C"
                    FROM Flow ORDER BY ID DESC"""

   p = pd.read_sql_query(sql, db)
   c.close()
   return p
```

4. Traffic Lifespans

The second algorithm is used to find lifespans of each communication and compares their similarity. Survival analysis is used for this purpose. The survival analysis was originally developed to measure lifespans of an individual. This analysis can be applied to any process duration. To estimate the survival function, we used Kaplan-Meier estimator. Mantel-Cox test is used to test each traffic and observe its conformity. This research is presented in [10]. This test is not fully implemented yet in the final version of GDP application.

5. Conclusion

In this paper, we presented our developed application GDP used to collect and analyze network traffic from NetFlow messages. Its benefit is the possibility to add own algorithms in the source code. This program creates a base for the intended following research. The duration traffic analysis is the first of the algorithms which were implemented. The second algorithm which is partly implemented is the lifespans. Other algorithms will follow to test the theoretical conclusions of analytical capabilities of NetFlow reporting.

In the near future, we want to expand our application with NetFlow version 9 and with the IPFIX format and to deploy genetic's algorithms.

Acknowledgment

Research and development described in this paper was financed by the National Sustainability Program under grant LO1401. For the research, infrastructure of the SIX Center was used.

References

[1] Cisco Systems, Inc., Introduction to Cisco IOS NetFlow – A Technical Overview. CISCO, 2012
[2] IETF, Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. IETF Tools, 2013.
[3] Cisco Systems, Inc.: NetFlow Services Solution Guide. http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/netflow/nfwhite.html [23.09.2016].
[4] Plixer International, Inc., Flow Analytics, Plixer–Malware Incident Response, 2016.
[5] Plixer International, Inc.: Top 5 Uses of NetFlow for Network Security. https://www.plixer.com/blog/netflow/top-5-uses-of-netflow-for-network-security/ [24.09.2016].
[6] Network Security Research.: GDP–NetFlow Collector, BUT, 2015.
[7] Plixer International, Inc., NetFlow packet Version 5 (V5), 2016.
[8] Cole N.: An introduction to npyscreen. http://npyscreen.readthedocs.io/introduction.html [24.09.2016]
[9] Python Software Foundation, Threading — Thread-based parallelism. Python 3.5.1 documentation, 1990-2016.
[10] Oujezský V., Horváth T., Škorpil V.: Modeling Botnet C& C Traffic Lifespans from NetFlow Using Survival Analysis. In Proceedings of the 39th International Conference on Telecommunication and Signal Processing, TSP 2016. International Conference on Telecommunications and Signal Processing (TSP). Vienna, Austria, 2016, 50–55.

M.Sc. Vaclav Oujezsky
e-mail: vaclav.oujezsky@phd.feec.vutbr.cz

Vaclav Oujezsky (MSc) was born in Brno, Czech Republic. Post graduate student at Brno University of Technology, Department of Telecommunications Senior Network Engineer at T-Mobile CZ and currently at IBM CZ. Working actively on projects of security and transport networks at laboratory SIX. His research interests include implementation of evolutionary algorithm, Cisco, Python, VHDL and converged networks. His topic of dissertation thesis is Converged Networks and Traffic Tomography by Using Evolutionary Algorithms.

M.Sc. Tomas Horvath
e-mail: horvath@feec.vutbr.cz

Tomas Horvath (MSc) was born in Havirov, Czech Republic. He received his MSc. degrees in Telecommunications from the Brno University of Technology, Brno, in 2013. His research interests include passive optical networks (xPON) and optoelectronics. Currently, he is post graduate student at Brno University of Technology, Department of Telecommunications. His topic of dissertation thesis is Optimization Services in FTTx Optical Access Networks.

otrzymano/received: 05.02.2017 *przyjęto do druku/accepted: 01.06.2017*