project report

Recognition of similar Netflows in decentralized monitoring systems

INSTITUTE OF INFORMATION RESOURCE MANAGEMENT
UNIVERITY ULM

Jan Braitinger

supervised by Georg Eisenhart

***Abstract*—Analysis of network data in decentralized systems lead to several problems. Multiple flow exporter can export the same flows, which results in doublets. This article shows how similar flows can be detected by filtering, generating vectors and calculating the distance between them with different similarity measures. Based on flow data of the flowpipeline tool, a practical experiment is shown. It shows that the Euclidean measurement paired with the prepossessing filters delivers good results but there are still open challenges in the area of direction, mapping and threshold.**

I. INTRODUCTION

The network traffic protocol *Netflow* collects traffic information about ip-data-stream on top of a layer-3-device [1]. It is defined as a tuple with at least 5 elements. Every flow contains the following attributes: Source IP, source port, destination IP, destination port and network protocol [2]. Additionally, there could be information about packages, size in bytes and the duration of the flow. It depends on the system which records and export such Netflows.
Recording flow data on two or more devices leads to duplicate results. Accordingly, adulteration arise in the evaluation of all data.
First, this work introduces methods to detect similar flows by using filtering and distance measures theoretically. After that, it will be demonstrated on real data of a defined environment.

II. GETTING SIMILAR FLOWS

In a multi-step way, the flows of each source are filtered and preprocessed for getting similarity based on distance measures. The following subsections explain the different steps. After the introduction of the theory, a practical example, based on this methods, will be shown.

A. Protocol filter

The network-protocol of similar flows has to be the same. Therefore, large sets of flows can be separated in a UDP set and a TCP set. Every Netflow dataflow has a protocol-field, which makes it is easy to filter.
The network protocol ICMP will not be recognized, because of the absence of ports.

B. Port combination

If two flows of different sources have the same ports (source and destination), the probability for a high similarity is greater. An easy proceed is to check if the source and destination addresses of the flows are equal.
Sporadically, one data flow records the away and the other data flow of the other source the way back. As a result, the port specifications are interchanged. To solve the problem, both port addresses (source and destination) are transformed to one hash.

Algorithm 1 Generate hash value for port combination

```
$N \leftarrow (port_{source} + port_{destination})$
$L \leftarrow len(N)$
$P \leftarrow md5.hash(L + N)$
```

P stands for one port combination of exactly one port pair. Now, it does not matter in which direction the examined flows are pointing.

C. IP check

As shown in the introduction, a flow is at least a 5 tuple. There are always the IP addresses of the source and the destination. As well as the mentioned ports combination, it is easy to compare the IP addresses of the examined flows ($source - source$ and $destination - destination$).
The problem of the different directions is disregarded, for now.

D. Create vector

The next step is to transform every flow to a multiple dimensional vector. It is important to check which information each flow contains because, as already mentioned, the size of the tuple of each flow depends on its exporter. Some flows could have information about the duration ($duration = FlowTimeEnd - FlowTimeStart$) and others not. That heterogeneity leads to problems! Furthermore, it is as inevitable to search for information that is as unique as possible. For example a flow can be transformed in a three-dimensional vector, consisting of the fields *Bytes*, *Packets* and *FlowTimeReceived*.

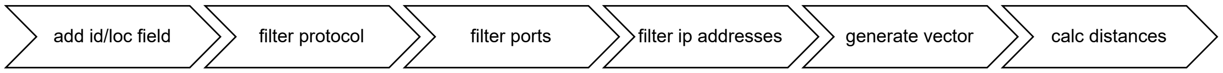

Figure 1. Steps to detect similar Netflows.

After choosing meaningful fields, a vector can be build. Now, each flow is defined as:

$$\vec{v} = \begin{pmatrix} Bytes \\ Packets \\ FlowTimeReceived \end{pmatrix} \tag{1}$$

Now it is possible to display each flow in a coordinate system and in the best case, similar flows are directly recognizable.

E. Distance measures

Feature selection for Intrusion Detection Model can use measures for structural equivalence [3]. There exist many distance measures which can be used in data science. In this work, three are tested to get the similarity between two vectors. The Euclidean distance is the most common way for calculating the distance between vectors.

$$\sqrt{\sum_{i=0}^{n} (\vec{q_i} - \vec{p_i})^2} \tag{2}$$

Grootendorst describes it as the length of a segment connecting two points [4]. A similar distance measure is the Manhattan distance by also calculating the distance between two vectors.

$$\sum_{i=0}^{n} |\vec{q_i} - \vec{p_i}| \tag{3}$$

The third measure is the cosine similarity which is the angle between two vectors.

$$cos(\theta) = \frac{\vec{p_i} \cdot \vec{q_i}}{||\vec{q_i}|| \, ||\vec{p_i}||} \tag{4}$$

If both vectors are exactly the same, the cosine similarity is 1. In contrast, if both are completely contrary, the similarity is -1.
The next step is to calculate the distances between every flow of both sources, as displayed in 2. Each vector $\vec{p_i}$ has to check which vector q_i^* is its most similar vector by creating a stack with all sorted distances. If multiple vectors $\vec{p_i}$ point to the same q_i^*, the vector with the least distance is the most similar. If some vectors should remain due to this, these are evaluated as noise.

A problem that still needs to be solved is how the threshold should be set. At what distance are two Netflows not similar anymore? It will be mentioned in the next section.

F. Further challenges

During the work on this project, several problems arose. First, the direction problem. It does matter at which time the flow is recorded, because the flow is bidirectional and each direction has its own pair of port and IP addresses.

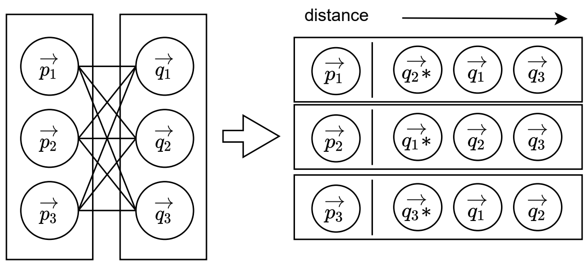

Figure 2. Get lowest distances of $\vec{p_i}$ to $\vec{q_i}$.

A further problem is about mapping. If all devices are configured equal, one flow of $\vec{p_i}$ will mapped (in case of similarity) to one flow of $\vec{q_i} \rightarrow$ 1:1 mapping. But in real world networks it is not the case. Every device is configured differently and this may result that one flow of $\vec{p_i}$ can mapped to multiple flows of $\vec{q_i}$ (1:N relationship).
Calculating the distances between the flows gives information about the similarity. But how can the upper limit of the distances be detected? From a certain distance, a flow is not anymore similar to the other flow. It is more a case of coincidence. Accordingly it is important to set a certain threshold which with great probability must be recalculated each time.

III. Practical experiment

In laboratory condition, based on two Raspberry Pis (Kirk and Spock as shown in figure 3) and two layer-3 switches, Netflows are examined. All devices are configured with static IP addresses and there is no connection to other networks. This is to avoid unnecessary connections, packets and flows while recording test scenarios. Both switches have the same Netflow-settings. That is important, because both save their flow after the same time period. Thus, the probability for a 1:1 flow mapping is higher.

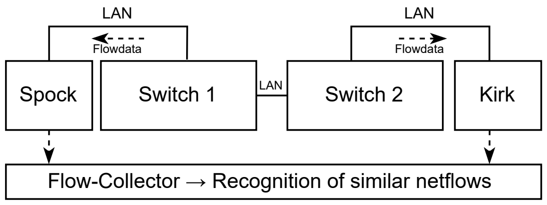

Figure 3. Laboratory setup

By sending larger files (>1GB) with the netcat tool from one Raspberry Pi to the other, traceable data flows are generated under defined conditions and are sent from the switches to the Raspberry Pis. That later simplifies the evaluation by hand. On both Raspberry Pis run the tool flow

pipeline which records the flows and saves them in a json file.
The result are multiple flows of each device. All flows are send to the flow collector which in this case is just a normal computer with the "recognition of similar flows" - tool. Similar to the mentioned theory, figure 1 displays the steps which are run through.

Each flow gets a unique ID (md5 hash of the whole json string of each Netflow) and based on its exporter, a location tag (which clarifies which switch the flow comes from). In this case, the flows are separated by the locations "kirk" and "spock". The next step is to filter the data based on protocol and ports and IP addresses. As mentioned before, the problem with the different directions is not directly considered here. It is just a simple check if the IP pairs of both locations are equal.

Table I
Result of shortest Euclidean distances between each flow pair.

Euclidean measurement		
$\vec{p_i}$	$\vec{q_i}$	distance
4	22	2832412.631243195
5	23	21219.92273784238
8	25	1045341.2417627078
9	24	61911425.75990718
10	26	1024121.3190364704
11	27	59079013.12866521

Further, a vector is generated by using the fields *Bytes*, *Packets* and *FlowTimeEnd*.
Beginning with the Euclidean measure, the distances of all vectors of kirk to spock are determined. The results of the most similar pairs (lowest distances) of vectors is shown in table I.

Table II
Result of shortest Manhatten distances between each flow pair.

manhattan measurement		
$\vec{p_i}$	$\vec{q_i}$	distance
4	22	2834304
5	23	21625
8	25	1065252
9	24	61952694
10	26	1043623
11	27	59118386

Next was testing the Manhattan distance measurement which is displayed in table II. The results are kind of similar to the Euclidean distance and also the pairs ($\vec{p_i}$ to $\vec{q_i}$) of the similar flows are equal.

Although the distances are very big, the results of flow pairs are correct. For the manual check, all bytes of flows of kirk are added up and compared with the sum of bytes of the spock. Both have the exact same sum of bytes. The same applies to the number of packages. Each $\vec{p_i}$ is connected to $\vec{q_i} \rightarrow$ 1:1 mapping. That can be considered as the highest similarity possible, because no flow remains.
By calculating the similarity using the cosine measurement, incorrect results have been produced. Table III shows that 5 of 6 flows pointing to the same vector.

Table III
Result of shortest cosine similarity between each flow pair.

cosine measurement		
$\vec{p_i}$	$\vec{q_i}$	angle
4	26	0.9544707791787631
5	22	0.9552817120138634
8	22	0.9547486836509638
9	22	0.9954712696918332
10	22	0.9544406940138226
11	22	0.9874416013623379

As well, the mapping between $\vec{p_4}$ and $\vec{q_{26}}$ is incorrect. Because of it, the systems recognize only 2 similar flow pairs. Compared with the other measurements and the manually rechecked pair of similar flows it can be said that the cosine similarity measurement is not purposeful here.

The similarity of the flows (based on Euclidean distances measurement) is well recognizable in figure 4. All doublet-pairs are connected with a line and have the same color.

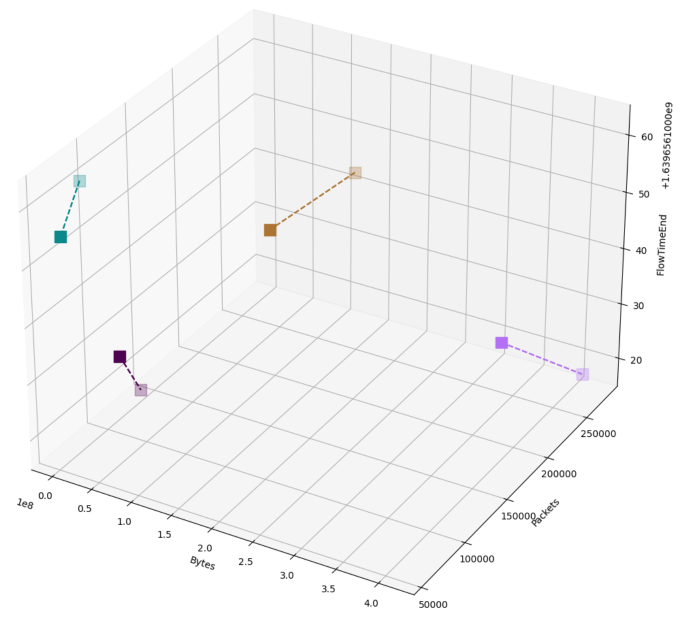

Figure 4. Displays similar flows in a 3-dimensional coordinate system.

IV. Conclusion

This work shows how to detect similar Netflows of multiple exporters by filtering for the single tuple objects, converting them into vectors and calculating the distances between them.

Further projects, based on this one, should address the problems of 1:N mapping and distances threshold. When solving how to handle Netflows from exporters with different settings one flow can be mapped to multiple other flows. One idea for an approach is to compose similar flows and calculating the distance of a group of vectors. The problem of distances threshold is to determinate at what maximum distance the flows are still similar. For that an algorithm should be developed which calculates a threshold for each run.

References

[1] Duygu Sinanc Terzi, Ramazan Terzi, and Seref Sagiroglu. Big data analytics for network anomaly detection from netflow data. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 592–597, 2017.

[2] Peichao Wang, Yun Zhou, Cheng Zhu, and Ruiqi Yue. Role classification with netflow data in intranet. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pages 279–282, 2018.

[3] Anirut Suebsing and Nualsawat Hiransakolwong. Feature selection using euclidean distance and cosine similarity for intrusion detection model. In *2009 First Asian Conference on Intelligent Information and Database Systems*, pages 86–91, 2009.

[4] Maarten Grootendorst. Distance measures in data science. In *Towards Data Science*, 2021.