TCP/IP communication flows into sentence-like transcriptions

Allan Kálnay

November 28, 2021

Abstract

The goal of this work was to design a suitable schema that transforms TCP/IP flows into sentence-like transcriptions and implement a software in Python that does such transformation from *pcap* files and to evaluate its suitability for attack detection.

The schema that we implemented uses printable ASCII characters in the range from the exclamation mark character (*!*) up to the tilde character ($\sim$) which makes 93 characters altogether. Our schema utilizes three network flow features to express the final transcription – *packet length*, *flow direction* and *inter packet times*. With a specific configuration that we used for these four features, we were able to convert huge pcap files (approximately 10GB) into few kB big files containing the sentence-like transcriptions of the network flows from those pcap files.

At the same time we were able to produce quite decent results for a binary classification problem distinguishing between attack and non-attack flows. For this purpose we used Random Forest Classifier on top of a basic bag of words data set constructed from the sentence-like transcriptions comparing it to performances of baseline models.

Contents

1 Background

1.1 pcap2transcription

pcap2transcription [3] is a preliminary project that tries to solve the same problem as our project tries. The project uses a similar technique for creating the transcription schema. It deals with IP packet lengths and flow directions the same as our project does. Uppercase symbols describe a communication in one direction and the lower case symbols describe the communication in the other direction. The dash character (-) stands for 10ms without packet exchange (communication gap/silence). Also, the project does not work efficiently when it comes to feature extraction as well as memory. We improved these two in our project. The pcap2transcription project is basically a base for our project.

2 Methodology

2.1 Data

For the purpose of our experiments we worked with two data sets:

- CIC-IDS 2017 [7],
- MAWI [4].

2.1.1 CIC-IDS 2017

First data set that we used is the synthetic CIC IDS 2017 data set [7]. The data set contains pcap files and labels of the network flows. The labels are binary – attack and non-attack.

There are 5 different pcap files in this data set. Each of them represents a different day in the week from Monday to Friday. The size of all of them vary from 7GB up to 14GB.

Table 1 describes the data sets contained in the CIC IDS 2017 data in terms of network flows, attacks and non-attacks

Day	# network flows	# non-attacks	# attacks
Monday	96 784	96 784	0
Tuesday	80 898	80 887	11
Wednesday	80 716	80 662	54
Thursday	77 069	75 948	1 121
Friday	73 726	73 534	192

Table 1: CIC IDS 2017 data description table

Labeling

The official page of CIC IDS 2017 data[1] guides you on how to do the data labeling of all days contained in the data set. However, we were provided with a labeling script by Félix Iglesias Vázquez and Fares Meghdouri which we highly appreciate. We double-checked the rules for labeling in the script and approved it for our use.

2.1.2 MAWI

Another data that we used in our experiments is the MAWI data set. This data consists of several sub data sets – days of different months and years. All of these are packet captures (pcap files), so they are perfectly suitable to our data processing process.

Table 2 describes the MAWI data that we used in our project in terms of number of network flows and number of attacks. Since we experienced insufficient RAM problems during feature extraction from this data, we could not construct all flows contained in this data. That's why `# network flows` column shows the minimal number of flows contained. The `# attacks` column captures only the number of attacks contained in the number of flows indicated in the `# network flows column`.

Date	# network flows	# attacks
2018-06-23	$>$ 3 857 573	58

Table 2: MAWI data description table

Labeling

In short, there are 4 different classes of network flows in the MAWI data:

- anomalous,
- suspicious
- notice,
- benign.

MAWI webpage documents how the data set labeling exactly works [2]. Each of these classes gets assigned based on certain technique.

Every specific data day contains its own labels in a CSV file. Every such CSV file contains an entry per attack communication. Each entry contains a source IP address and a destination IP address. From the data web page, it is not clear how to correctly label the data. From an e-mail communication with the authors, we found out that in order to label network flows of a certain date, we have to simply assign an *attack* label to every flow whose source IP and destination IP address pair is contained in a CSV file of labels.

[1]`https://www.unb.ca/cic/datasets/ids-2017.html`

2.2 Data Exploration

Since the beginning we decided to work with several network flow features – protocol identifier, packet length, communication gaps (communication silence), communication direction, packet timestamp. Based on this decision we researched our data set to find out more about these features and to drive our decisions later based on our findings.

The protocol identifier was purely used for identification whether a packet is a TCP/IP packet or not. If it is not a TCP/IP packet, then, as the name of the project suggests, we do not care about the packet. If it is a TCP/IP packet, we work with that.

2.2.1 Packet Size

As [5] states, the minimum size of a TCP/IP packet is 21 bytes and the maximum size of such packet is 65 535 bytes. We analysed the cumulative distribution function (CDF) of packet sizes of the packets of all CIC-IDS 2017 days. The output of our analysis is the figure 1. The figure provides us with information of what percentage of packets have size less than or equal to a certain value. The figure shows that almost all of the packets of all days are less than or equal to 5kB.

2.3 Transcription Schema Description

2.3.1 Feature extraction

Our transcription schema utilizes several network flow features. Among these are also the features that determine how output transcription schema will look like. These are:

- *packet length*,
- *flow direction*,
- *inter packet time.*

These three are basic features that can be easily extracted from both encrypted and non-encrypted network communication. Moreover, the results from the research [6] show that packet length based features are a decisive factor for identifying malicious activity.

2.3.2 Transcriptions

A transcription is a string of characters describing a network flow. We use 93 characters for constructing a transcription string. These are printable ASCII characters in the range from the exclamation mark character (*!*) up to the tilde character ($\sim$) where each character represents a captured packet length. Each character represents different packet lengths. Transcription string characters are ordered in the same chronological order as the packet captures.

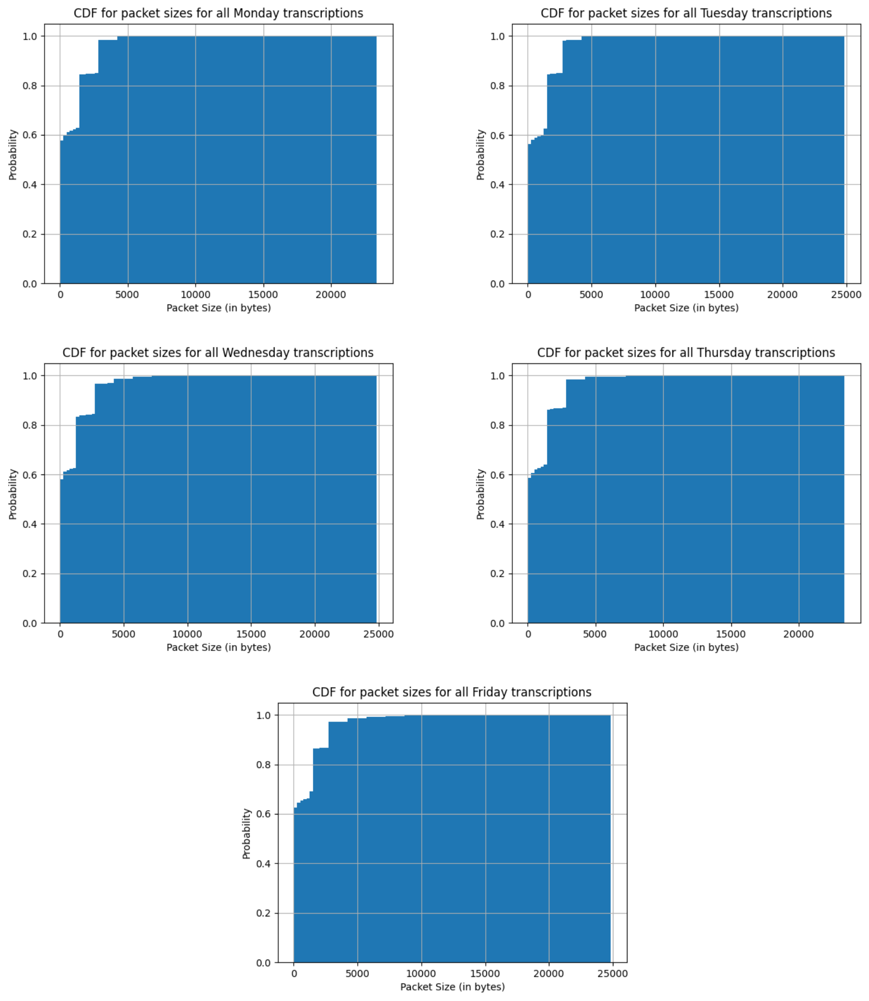

Figure 1: CDF plots for packet sizes of all CIC-IDS 2017 days

The very first character, exclamation mark, is used to represent a silence in a communication. One exclamation mark represents a silence of 1 second. The 46 characters starting from the quotation mark included up to the O character included are used to describe one direction of a communication. The 46 characters starting from the P character included up to the tilde character included are used to describe the other direction of a communication. The meaning of transcription characters is captured in a tabular form in the table 3.

Character	Meaning
!	Silence in a communication of 1s time period.
" up to O	A packet sent from host A to host B. Each of these characters represents different packet length bucket.
P up to $\sim$	A packet sent from host B to host A. Each of these characters represents different packet length bucket.

Table 3: Meaning of transcription characters

Based on the research in the section 2.2 we utilized the characters in such way that the characters represent different bins of packet lengths. We reasonably chose binning based on the analysis of the figure 1. We encoded the packets of lengths $<= 2^{14}$ with first 30 characters, the packets of lengths $(2^{14}; 2^{16}]$ with the next 15 characters and the packets of lengths $> 2^{16}$ with the last (46th) character. Naturally, each communication direction utilizes its own character set as described in the section 2.2. In every category the bins are of the same size. Meaning, in the first category there are 30 bins of the same size, in the second category there are 15 bins of the same size and in the last category there is just one bin.
An example transcription may look like this:

$$CCn!D!n$$

The meaning of the transcription character by character is the following:

1. C – host A sent a packet of length between 8752B up to 9298B
2. C – host A sent a packet of length between 8752B up to 9298B
3. n – host B sent a packet of length between 15863B up to 16384B
4. ! – 1s of communication silence
5. D – host A sent a packet of length between 9299B up to 9846B
6. ! – 1s of communication silence
7. n – host B sent a packet of length between 15863B up to 16384B

2.4 Implementation

2.4.1 Data Processing Pipeline

The software for processing the *pcap* files and converting network flows into sentence-like transcriptions is a data processing pipeline consisted of two steps. The two-step process consists of extracting the necessary features from a *pcap* file with *go-flows* software [1] and the 2nd step is the transcriptions building process itself based on the output from the 1st step. The data processing pipeline itself is a shell script that orchestrates the steps to be done in order to achieve the final output. The path to the script is `experiments/01/transcription/pipeline_general.sh`

2.4.2 go-flows Features Extraction

First step of data processing pipeline is extracting features from a pcap file using go-flows. go-flows uses a configuration file which describes what features we want to extract from a pcap. The configuration file for go-flows used in our data processing pipeline is located in `experiments/01/transcription/feature_extraction/2tuple_bidi.json`. This configuration file makes go-flows produce a CSV file containing network flows with following features:

- *flowStartMilliseconds* – beginning of a flow timestamp in ms,
- *flowDurationMilliseconds* – flow duration in ms,
- *sourceIPAddress* – source IP address of a flow,
- *destinationIPAddress* – destination IP address of a flow,
- *accumulate(ipTotalLength)* – accumulated lengths of packets – array,
- *accumulate(_interPacketTimeMilliseconds)* – accumulated inter packet times in ms – array,
- *accumulate(flowDirection)* – accumulated flow directions related to packets – array.

Each row of the output CSV file represents a single network flow.

2.4.3 Transcriptions Maker

As mentioned above, the 2nd step of the data processing pipeline is the transcriptions building process. This is the core software of this project implemented in Python expecting a CSV file of a certain structure as an input and producing a TSV file containing the sentence-like transcriptions as the output among other features described in the section 2.4.4. Let's call this software the transcription maker.

The input CSV file for this core software is an output produced from go-flows. All the necessities for this input CSV file are described in the section 2.4.2.

Transcription maker read the input CSV file line by line (a.k.a. flow by flow) and traverse the array features in order to produce the output characters building the final transcription strings.

The final structure of the output TSV file produced by the transcription maker software is described in the section 2.4.4.

The source code of the transcriptions maker software is located in the project's root in the folder `experiments/01/transcription/src/main/python/transcription`.

2.4.4 Output Data Set

The output data set containing transcriptions derived from pcap files is a TSV file. The output file contains 5 columns. These are:

- *flowStartMilliseconds* – flow start in milliseconds,
- *flowDurationMilliseconds* - flow duration in milliseconds,
- *sourceIPAddress* – source IP address of a flow,
- *destinationIPAddress* – destination IP address of a flow,
- *transcription* – the transcription string itself.

3 Evaluation

In the following sections, we conducted several experiments on top of different data sets. For each data set we trained and evaluated two types of models – baseline and obtained model.

Baseline Model

As a baseline model we chose to train Random Forest models. We aggregated several network flow features in order to create data sets of numeric vectors. We used the very same network flow features for constructing these numeric vectors of the data sets as the ones that we use for constructing transcriptions. These features are:

- *packet length*,
- *flow direction*,
- *inter packet time.*

For each *flow direction* (forward/backward) we used each of these aggregation functions on top of the *packet length* and *inter packet time* features:

- *min*,
- *max*,
- *median*,
- *mean*,
- *mode*,
- *stdev.*

Obtained Model

In the each experiment, we decided to train a Random Forest models on top of bag of words for the transcription data sets comparing them to baseline Random Forest models. We will call the non-baseline models *"obtained models"*.

In the experiments we work with the same schema of transcription data sets as described in the section 2.4.4 and the same transcriptions schema as described in the section 2.3.2.

We used a bag of words technique to express our transcription strings as numeric vectors. For each entry of the data set (a transcription) we counted number of each word (symbol) in the transcription. Basically, we transformed our data set of transcriptions to the data set where our dimensions represented the word counts in transcriptions. For better understanding, see an example at figure 2.

srcIP	dstIP	attack	transcription
1.1.1.1	2.2.2.2	1	AAp!D!p
2.2.2.2	3.3.3.3	0	P!PD

srcIP	dstIP	A	D	P	p	!
1.1.1.1	2.2.2.2	2	1	0	2	1
2.2.2.2	3.3.3.3	0	1	2	0	1

Figure 2: Data set of transcriptions (top table) and bag of words data set (bottom table)

3.1 CIC-IDS2017 Data Set Experiments

The experiments provided in this section were conducted on top of the CIC-IDS2017 data set described in the section 2.1. As the whole data set consists of several sub data sets, we conducted experiments on top of almost each one of these.

As Monday data do not have any malicious traffic, we skipped this one. All the data sets from Tuesday to Friday contain both malicious and non-malicious traffic, therefore we conducted an experiment on top of each of them.

From each data set from Tuesday to Friday, we derive two different data sets for training and evaluating two different models – baseline and obtained. Both of these models are trained for binary classification, meaning deciding between the two classes – *attack* and *non-attack*.

For both the baseline and non-baseline models we split both attacks and non-attacks in the 70:30 train set to test set ratio. Then we merge non-attacks with attacks for both train and test data separately in order to create a single train set and a single test set.

We didn't necessarily train and test our baseline models on exactly the same train and test network flows as we did for the models on top of transcriptions data. Nevertheless, we subsampled from the same sets of network flows which minimizes the impact of not using the same network flows for training and testing the models. The counts for train and test sets are exactly the same for both models in each day, i.e. if Thursday's baseline model has a train set consisted of 1 500 non-attacks and 100 attacks and a test set consisted of 700 non-attacks and 50 attacks, then the same counts for train and test sets are applied to the Random Forest on top of bag of words model.

The goal for these experiments was to achieve the highest possible precision for both the *attack* class and the *non-attack* class. Achieving good accuracy results on top of whole test sets is meaningless for us as our data sets are imbalanced and therefore accuracy may distort results.

3.1.1 Friday

For this experiment we used the Friday's CIC-IDS 2017 data. We sampled 5 500 non-attacks and used all attacks from this data set. This makes altogether a data set of 5 692 entries.

The predictions of a obtained model gives us very promising results as we can see at Figure 3. The precision scores for both non-attacks and attacks classification are almost the same – around 92%.

The baseline model achieves better results here in regards to predicting non-attacks with 100% precision compared to the obtained model scoring 92.4% precision than the obtained model. In regards to attacks, the models perform exactly same well. The results of the baseline model are captured in the Figure 4.

	true non-attack	true attack
predicted non-attack	92.36%	7.64%
predicted attack	8.62%	91.38%

	true non-attack	true attack
predicted non-attack	1525	125
predicted attack	5	53

Figure 3: Confusion matrices with normalized and absolute values of the obtained model for the experiment 1

	true non-attack	true attack
predicted non-attack	99.15%	0.85%
predicted attack	6.90%	93.10%

	true non-attack	true attack
predicted non-attack	1636	14
predicted attack	4	54

Figure 4: Confusion matrices with normalized and absolute values of the baseline model for the experiment 1

3.1.2 Wednesday

Here we are working with CIC-IDS 2017 Wednesday data set. We sampled 5 000 non-attacks and we kept all 54 attacks. After splitting the data set to train and test sets, we obtained 3 500 non-attacks and 37 attacks in the train set and 1500 non attacks and 17 attacks in the test set.

The baseline model was able to score slightly better results here than the obtained model. Concretely, classifying two more attack correctly.

	true non-attack	true attack
predicted non-attack	100.0%	0.0%
predicted attack	23.53%	76.47%

	true non-attack	true attack
predicted non-attack	1500	0
predicted attack	4	13

Figure 5: Confusion matrices with normalized and absolute values of the obtained model for the experiment 2

	true non-attack	true attack
predicted non-attack	99.93%	0.07%
predicted attack	11.76%	88.24%

	true non-attack	true attack
predicted non-attack	1499	1
predicted attack	2	15

Figure 6: Confusion matrices with normalized and absolute values of the baseline model for the experiment 2

3.1.3 Tuesday

In this experiment we work with CIC-IDS 2017 Tuesday data set. We sampled 5 500 non-attacks and we kept all 11 attacks. Splitting the data set into train and test sets results in 3 850 non-attacks and 37 attacks in the train set and 1650 non attacks and 4 attacks in the test set.

The confusion matrices for the obtained model in Figure 9 again show promising results.

By comparing the baseline model results in the Figure 8 to the results of the obtained model, we conclude that the both models performed the same well.

	true non-attack	true attack
predicted non-attack	100%	0.0%
predicted attack	25.0%	75.0%

	true non-attack	true attack
predicted non-attack	1650	0
predicted attack	1	3

Figure 7: Confusion matrices with normalized and absolute values of the obtained model for the experiment 3

	true non-attack	true attack
predicted non-attack	100%	0.0%
predicted attack	25.0%	75.0%

	true non-attack	true attack
predicted non-attack	1650	0
predicted attack	1	3

Figure 8: Confusion matrices with normalized and absolute values of the baseline model for the experiment 3

3.1.4 Thursday

In this experiment we work with CIC-IDS 2017 Thursday data set.

The data set contains 77 069 entries out of which only 1 121 are attacks. We sampled 20 000 non-attacks and we kept all 1 121 attacks. Splitting the data set into train and test sets results in 14 000 non-attacks and 784 attacks in the train set and 6 000 non attacks and 337 attacks in the test set.

The confusion matrices in Figure 9 show worse results than in the previous experiments with attacks classification precision nearly 50% and non-attacks classification precision slightly above 75%.

The baseline model scores better results than the obtained model here. We can observe its results in the Figure 10. The precision of non-attacks classification is about 9% higher than in the obtained model and the precision of attacks classification is about 13% higher than in the obtained model.

	true non-attack	true attack
predicted non-attack	75.40%	24.60%
predicted attack	51.93%	48.07%

	true non-attack	true attack
predicted non-attack	4524	1476
predicted attack	175	162

Figure 9: Confusion matrices with normalized and absolute values of the obtained model for the experiment 4

	true non-attack	true attack
predicted non-attack	88.41%	11.58%
predicted attack	43.03%	56.97%

	true non-attack	true attack
predicted non-attack	5305	695
predicted attack	145	192

Figure 10: Confusion matrices with normalized and absolute values of the baseline model for the experiment 4

3.2 MAWI Data Set Experiments

The experiments provided in this section were conducted on top of the MAWI data set described in the section 2.1. As the whole data set consists of several sub data sets, we conducted experiments on top of almost each one of these.

The MAWI data set contains a lot of sub data sets. We chose some data sets that contain higher number of attack flows compared to the other sub data sets.

3.2.1 23rd June 2018

Out of the whole data set consisted of 276 anomalous flows (attacks) and several millions of non-attack flows, we generated only the first 3 857 573 flows. This is the maximum number that we were able to generate with our machine. The network flows generation process was too heavy in regards to RAM memory and did not allow us to generate more flows. This way we lost remaining 218 attacks and various other non-attack flows.

Out of these almost 3.9M flows, we sampled 500 000 non-attacks and all 58 attacks. The train set contains 35 000 non-attacks and 40 attacks. Test set contains 15 000 non-attacks and 18 attacks.

In this experiment, our obtained model outperformed the baseline model. The difference between precisions for non-attacks is negligible. The difference between precisions is around 16%.

	true non-attack	true attack
predicted non-attack	99.95%	0.05%
predicted attack	16.67%	83.33%

	true non-attack	true attack
predicted non-attack	14992	8
predicted attack	3	15

Figure 11: Confusion matrices with normalized and absolute values of the obtained model for the MAWI experiment 1

	true non-attack	true attack
predicted non-attack	99.94%	0.06%
predicted attack	38.89%	66.67%

	true non-attack	true attack
predicted non-attack	14991	9
predicted attack	6	12

Figure 12: Confusion matrices with normalized and absolute values of the baseline model for the MAWI experiment 1

4 Conclusions

In all experiments on top of CIC-IDS 2017 data, the baseline models scored better results than the obtained models except of one experiment where they both achieved the very same results. However, in most cases the baseline models performed just slightly better than the obtained models. This means that the very first outcomes in regards to the transcriptions data set look promising.

We provided only 1 experiment on top of MAWI data, concretely on top of the 23rd June 2018 data set. Here, the obtained model scored overall better results than the baseline model. Particularly, a huge difference (more than 22%) is in precision of attacks classification.

One potential reason why our obtained models mostly did not succeed in scoring better results than the baseline models is lack of train data and particularly attacks data. The transformation of transcription data sets into bag of words data set was failing on bigger data sets due to insufficient RAM. This is the reason why we subsampled only certain amount of non-attack samples for each experiment. Using more samples to be transformed into bag of words was crashing our scripts.

Next potential improvement may be choosing a different model instead of Random Forest on top of bag of words and evaluating its results. Using a simple bag of words may be too naive method. Researching and using a different kind of technique may score better results.

Giving more attention to train and test set splits is another kind of improvement. As mentioned in the section 3.1, the obtained and the baseline models are not necessarily trained and tested on the same network flows. On such few data points a slight shuffle in train and test data may play a big role in the final results. Therefore conducting experiments that train both an obtained model and a baseline model in each experiment on the very same network flows and evaluate their performance on the same test sets may give us more higher quality results.

Regarding the implementation of transcription maker software itself, one may try experimenting with IP packet lengths binning. Would be worth reconsidering the binning approach and assigning more characters to express the packets of lengths <= 5 000 in order to better express variability among these.

References

[1] go-flows · pkg.go.dev. `https://pkg.go.dev/github.com/CN-TU/go-flows?utm_source=godoc`. (Accessed on 08/09/2021).

[2] Mawilab - documentation. `http://www.fukuda-lab.org/mawilab/documentation.html`. (Accessed on 11/28/2021).

[3] felixig. py_pcap2transcription. `https://github.com/CN-TU/py_pcap2transcription`, 2020.

[4] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *ACM CoNEXT '10*, Philadelphia, PA, December 2010.

[5] Eric Hall. Internet core protocols: The definitive guide. `https://www.oreilly.com/library/view/internet-core-protocols/1565925726/re04.html`. Accessed: 2021-07-18.

[6] Fares Meghdouri, Tanja Zseby, and Félix Iglesias. Analysis of lightweight feature vectors for attack detection in network traffic. *Applied Sciences*, 8(11):2196, 2018.

[7] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.