Anomaly detectionn

Artem Panchenko
artempanchenkotop@gmail.com
Oleksandr Deineha
jesse.pinkman.d@gmail.com

March 8, 2020

Abstract

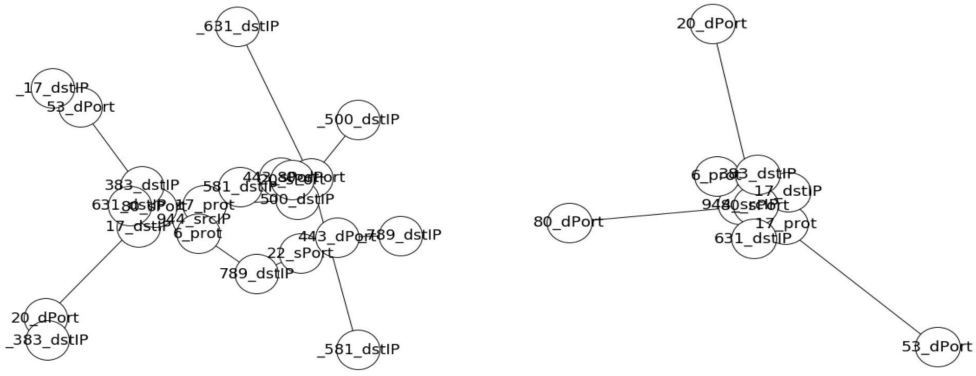
The main goal of our final project is to develop the method to carry out the anomaly detection in IP traffic. For this purpose we will use the Profile graphlet approach (1). Also we will build the model using SVM to distinguish normal from malicious end host from an annotation trace. The last step will be to try to detect the attack in a not annotated trace.

1 GRAPHLETS BUILDING

The firs step of this project is to convert the network flows from the dataset in the graphlet form. In our project we will work with the Profile graphlets. We will build them according to the algorithm form the (1). The one difference between the approach from the paper and what we did - we delete the copy of the Destination ip. We did it for reasons of optimizing the work of Random walk kernel that we will use in the future. It was completely safe operation because:

- We already take into account the destination ip.
- This is just a leaf of our tree.

Another specialty of our implementation of Profile graphlet building is that when we have a empty graphlet (this situation is possible) we create a graphlet with with one node - source ip. We do it to avoid the problem with the Random walk kernel - the exeption occurs when we try perform it on the empty graph.



(a) Activity graphlet (b) Profile graphlet

Figure 1: Activity and Profile graphlets

So, in Figure 1 we see the Activity and Profile graphlets for the same source ip that were building by the our algorithm.

2 BUILDING OF THE MODEL SEPARATING THE NORMAL FROM THE MALICIOUS END HOSTS

We will use the two approaches to build it:

- Using the kernel trick to avoid the mapping of the graphlets in the high dimensional space.
- With do the explicit mapping, then build the model.

We will the evaluate the complexity of the algorithms in terms of the execution time. We will perform all computation on the remote machine (Google collaboratory), so the time will be depend only on the computations.

2.1 USING THE KERNEL TRICK

We need to notice that in this subsection we will work with 100 random normal graphlets and all malicious graphlets. Perhaps we sacrifice the accuracy of the model, but our machine and no one remote from those that we tried (Google collaboration, Amazon) are not able to perform these operations for a complete dataset in adequate time.

We perform the Kernel trick and create the model via SVM, building of simularity matrix for 154x154 graphlets took us 1 hour. After that we did the Cross-validation test of our model. We have the following results:

- Average fit time of our model = 0,0033.
- Average score time of our model = 0,000938034.
- Average accuracy of our model = 62,342 %.

We performed Cross-validation and found best parameters but we can't perform prediction via our model on the test dataset, because we will again need to build the simularity matrix and it waste a lot of time.

So, our result is better that just random choice. Is it good result for situation when we are - we work only with 154 graphlets.

The problem of this approach is that in average our profile graphlet has 25 nodes, so product graph will have in average 25^2 nodes, so will need to analyze $(25^2)^2$ edges during building the Random walk kernel, So, building of the product graphs takes us the most of the time.

2.2 EXPLICIT MAPPING

In this subsection we work with all graphlets and also we return the leaf of the graphlet - node that copy the destination ip. We created the function for mapping the graph to Random walk kernel of length k, where k is the parameter k = 4, 10 ,20, 50, 100. Time needed to perform the mapping is much better that in case of Kernel trick usage, as we don't work with product graph and number of nodes we working with is much smaller and we need to perform it only on 1001 graphlets not 1000x1000 as in case of Kernel trick. So the total time to perform the mapping for any k is less than minute for whole dataset.

Also we need to notice that we use RBF to increase the quality pf the model. So, we tuned SVM with the RBF kernel for each k and got following results:

k	F1 - score
4	0.053
10	0.02
20	0.02
50	0.02
100	0.02

So, the best result in terms of the F1-score is when we set k = 4.

3 DETECTION OF POTENTIAL ATTACKS

do it we use the model that we build with Explicit method and fited with the best parameter that we tuned in Section 2.2. We perform our model on the not annotated trace and tried to predict the anomaly traffic. And we got one:

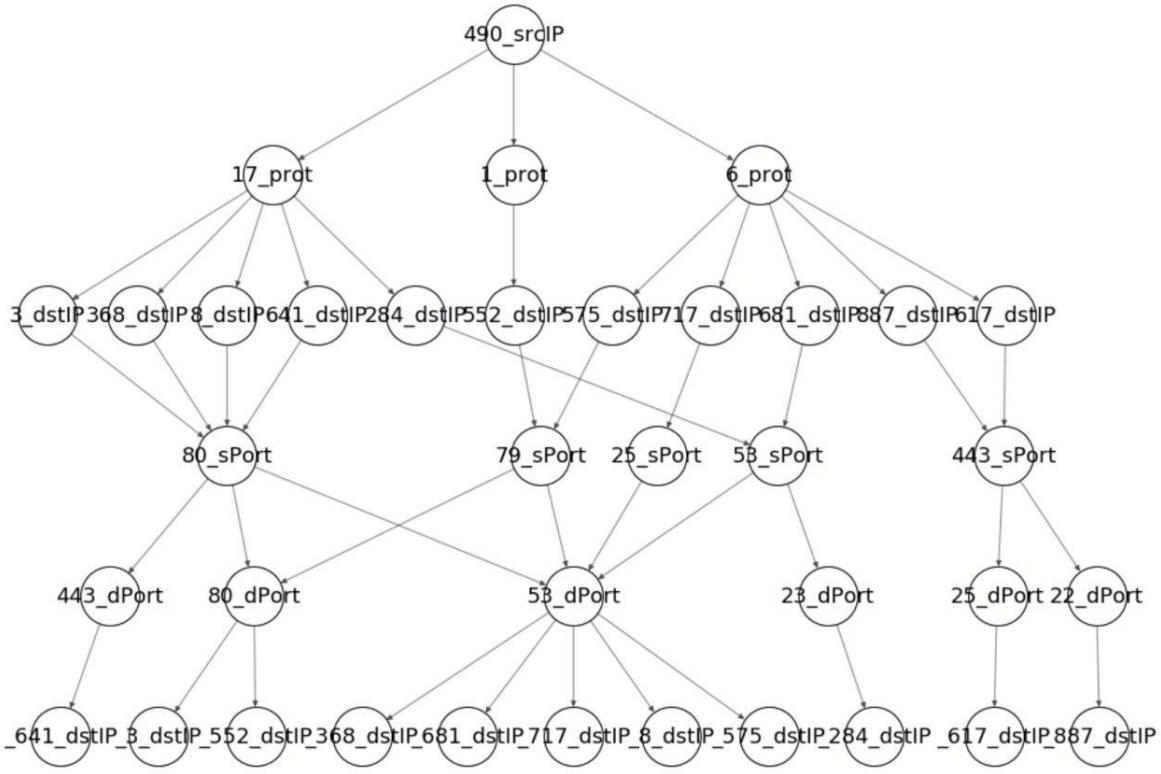


Figure 2: Anomaly Profile graphlet

It is extremely hard to define the type of attack via this Figure. But there one strange thing - lets look closely in the Destination port 53. We see that different protocol and different source port is connected to the one destination port 53. It is possible that corrupted user in Source ip 490 start the many programs that push request on this 53 port to different machines. Port 53 is used by the Domain Name System (DNS). Because port 53 is usually open, malicious programs may attempt to communicate on it

4 DISCUSSING THE RESULTS

We had the bad F1-score value because the percentage of anomaly graphlets in our dataset is extremely low. Also as we saw from the plots, created via PCA (available in the source code) there no clear separation between classes, anomaly graphlets distributed uniformly.

We can have False positive result in case of using this approach because the anomaly graphlet doesn't mean that it is attack, False negative results are occurred because of very bad dataset in terms of percentage of class partitioning,

5 IDEAL KERNEL

For this experiment we take the library `https://github.com/ysig/GraKeL`. We perform the our model with this kernel for 50 normal graphelets and all malicious for the decreasing

performing time and got the following F1-score via Cross-validation (it is impossible to do the test because of extremely long performing time): 0.639

REFERENCES

[1] Nina Taft Thomas Karagiannis, Konstantina Papagiannaki and Michalis Faloutsos. *Profiling the End Host*.