Detection of IoT traffic anomalies based on unsupervised learning methods

STUDENT NAME: Meng Tianao

STUDENT ID: 20810553

COURSE: CS656

Introduction

Nowadays, more and more things are connected to internet. By some estimates, as of 2015 there are already 5 billion things connected to the Internet, and the number could reach 25 billion by 2020 [1]. These things include our smartphones, which already follow us around in our homes, offices, and cars, reporting our geo-locations and usage data to our ISPs and Internet applications [9]. I want to give an example about Internet connected things. Imagining a house appliance is connected to the internet, such as the washing machine, we can use our smartphones to control the washing machine to start the washing machine, to monitor washing process, and to change the washing mode. And this kind of washing machines is produced by almost all the washing machine producers, and you can see this kind of washing machines almost in most apartments.

But the proliferation of IoT devices that can be more easily compromised than desktop computers has led to an increase in IoT-based botnet attacks [2], e.g. Mirai. Because of IoT devices easily comprised, Mirai botnet has a huge impact and widely spread. The public release of its source code released in 2016 has led to a large number of Mirai variants and increased frequency of Distributed Denial of Service (DDoS) attacks [6]. As [7] mentioned, there are several operation steps involved in Mirai botnets, propagation, infection, C&C communication, and attack.

In this project, I will detect anomalies on the last step. There are two reasons why choose the last step to do detection. Firstly, easily get the data after infection in [5], and I choose the gathered from SimpleHome_XCS7_1002_WHT_Security_Camera infected by BASHLITE and Mirai. Secondly, the botnets attacks become more and more complicated, changing almost every day. It is more likely that due to these changes, those attacks could bypass those early steps detection strategy.

The dataset I will use in this project include benign traffic, and two other IoT based bonnets, i.e. gafgyt attacks and mirai attacks. These two kinds of botnets are the most common botnets. I will do a comparative analysis between different unsupervised learning algorithm (K-means and DBSCAN) to cluster the given dataset. For the given dataset clusterred with K-means, I will use 2/3 data of benign and attack data to train the K-means cluster model, and use 1/3 data to test the trained model. For DBSCAN, I will just use it to cluster the test dataset after choosing reasonable parameters of DBSCAN. Then based on the experiments results on the given dataset, to compare the performance (error rate, true positive, and false positive) difference between these two algorithms.

In the system design part, I will introduce the two unsupervised learning algorithms I choose for this project, and present the reason why I choose these two algorithms, i.e. K-means and DBSCAN; Furthermore, my project is focus on the basic unsupervised learning algorithms analysis.

Literature Review

In this part, I will focus on the machine learning algorithms and some deep learning algorithms that have been exploited in detecting the traffic anomalies these years. Many algorithms were used for detecting the traffic anomalies, such as KNN, Decision tree, MLP [4]. In [4], they use the same dataset [5]. they randomly select 20000 instances, where 10000 instances are used for training, the other 10000 instances are used for testing. Possebon et al find that the accuracy for KNN is 0.9995, for Decision Tree is 0.9996, and for MLP is 0.4948; Furthermore, the False positives for KNN, DT, and MLP are 1.04, 0.72, 1050.28 respectively. The True positives for these three algorithms are 1998.92, 1999.08, and 1029.56 respectively.

	KNN	Decision Tree	MLP
False positives	1.04	0.72	1050.28
True positives	1998.92	1999.08	1029.56
Accuracy	0.9995	0.9996	0.4948

Tabel 1: results for KNN, DT, and MLP

The number of false positives corresponds to the number of instances considered normal flows that were classified as anomalous flows. Likewise, the number of true positives corresponds to the number of instances considered anomalies that were classified as, in fact, anomalies [4]. From table 1, we can see that the decision tree achieves a relevant higher accuracy, while the MLP gets a very low accuracy results; Moreover, the number of false positives for KNN and DT is much lower than that of MLP; Furthermore, the MLP gets a low number of true positives. In my experiment part, I will give the confusion matrix to give a clearer result.

In [2], Y. Meidan et al use deep autoencoders, which is a deep learning method to detect botnet attacks. And they do the comparation with isolation forest, LOF, and SVM methods, and get that Autoencoder get the lowest FPR and TPR; Furthermore, achieve the least average detection time. And the real traffic data [5] is also the main contribution by Y. Meidan et al, gathered from nine commercial IoT devices infected by authentic botnets from two families.

Based on the previous effort on supervised learning method and deep learning method, I select the unsupervised learning methods, for which there are not enough paper to do the experiments on the real traffic data. In other words, I will not use the label in the training stage, just clustering the given dataset, then use the label of test dataset to check the error rate, and draw the confusion matrix.

System design

Firstly, let's consider the difference between the supervised learning algorithms and unsupervised learning algorithms. In supervised learning algorithms, we need the true labels are available, which acts as an expert to train our model. As for unsupervised learning algorithms, the true labels are not available; Therefore, in my training stage for k_means, I will not use the labels for training although we can get the label for those samples, because my report is focus on the unsupervised learning algorithms; therefore, we just pretend to not know about the labels for each sample.

After we know the difference between supervised learning algorithms and unsupervised learning algorithms, we need to consider why we use the unsupervised learning algorithms?

Firstly, give the large dataset labels for each sample is a very costly procedure. As I mentioned above, the botnets attacks become more and more complicated, changing almost every day; therefore, as time goes, there could be a large number of different types of bonnets, it will be much more costly to label all of the sample to train the model; Secondly these changing of different types of bonnets could also lead to that class label may not be known beforehand. The company could need to hire more experts to do this kind of labeling, and this is also a cost of human resources.

In this term project, I will use k-Means algorithm, which is one of most famous unsupervised learning algorithms, and has been used in a lot of applications, such like dimension reduction, vector quantization.

The k-means method is a widely used clustering technique that seeks to minimize the average squared distance between points in the same cluster [10]. It is very simple to use and has a high speed of convergence. These properties appeal me to use K-means to cluster the dataset; However, k-means does not offer accuracy guarantee, and another main problem is that we need to determine the number of k, which is the number of clusters beforehand. But in practice, we could just choose two clusters, one for normal, another for malicious. In my term project, because I have some types of dataset, benign, gafgyt, and mirai, the number of k is three in my experiment. In Kmeans application in other areas, although there are some limitations of choosing this k, we can solve it by using some optimization method, such as PSO, to get the best number of K. In my project, I will not include those optimization strategies, and I only do some comparation with another famous unsupervised learning algorithm, DBSCAN.

Before I introduce DBSCAN algorithms, let's firstly summarized the drawbacks of distance-based clustering, e.g. K-means. In distance-based clustering method, we only consider one representative point for each cluster. And distance-based clustering method is only good for convex shape, similar size, and similar density problem; another is the limitation mentioned in the last paragraph, that is choosing the best k. Based on these drawbacks, I will introduce DBSCAN now.

DBSCAN stands for density based spatial clustering of application with noise. It discovers regions of high density separated by-not-dense region; furthermore, it can discover clusters with arbitrary shape, handle noisy data and outlier. Last but not least, it does not need the number of clusters beforehand. The density at point P defined in DBSCN is Nr(P) = number of points within a circle with radius r. And the dense region is A circle with radius ϵ that contains at least Minpts point. And this Minpts is what we need for implementation DBSCAN.

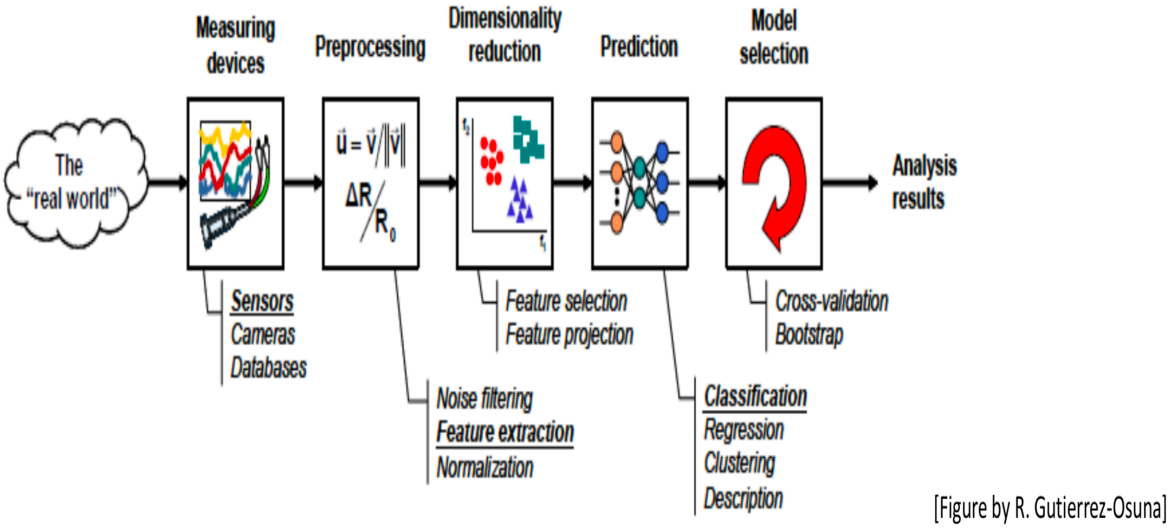

Figure1: pattern recognition system [10] used in my project

As Figure 1 shows, I will start by feature dimensionality reduction, by using PCA. Then I will do the normalization to scale the features to a range between 0 and 1, which is very convenient for us to visualize the dataset because after PCA, the order of the feature is up to 16; furthermore, we can more clearly see the difference of the two algorithms when we scale the data. I will use the two clustering methods, which are mentioned above K-means and DBSCAN, to do the prediction part shown in Figure 1. The cross-validation method is shown in the last step of the designing, which is always used for avoiding overfitting; however, those avoid overfitting method is not always used in unsupervised learning. Instead, I will try different parameters for DBSCAN to cluster the test dataset based on the number of samples and their distribution. By visualizing the data, do some fine-tuning to get better results for DBSCAN.
As for K-means, the only parameter we need choose is the number of clusters. Because we know the number of clusters beforehand, this is much easier in this project.
In conclusion, there are several experimental steps involved in my project:
1. Dataset processing (dimension reduction)
2. Normalization
3. Parameter chosen (k for k-means, radius r and Minpts for DBSCAN)
4. Visualize their clustering on test dataset
5. Get the performance of our model on the test dataset

Results And Discussion

1.Dataset for this experiment:

I choose the gathered from SimpleHome_XCS7_1002_WHT_Security_Camera infected by BASHLITE(gafgyt) and Mirai. The total number of samples included in this dataset is around 1 million. To simplify my experiment and reduce the running time of the DBSCAN algorithm, I choose to use a smaller size of dataset. I randomly choose 5000 benign data ,5000 BASHLITE(gafgyt) data, and 5000 Mirai data; Furthermore, in the visualization and error calculate part, I label both Mirai data and BASHLITE(gafgyt) data as malicious.

2.Dataset preprocessing:

I use the PCA (principal component analysis) to do the dimension reduction, which makes the accuracy of the processed data is 98%. The number of dimensions after PCA processing is 2.

```
original number of features for each sample:  115
number of features for each sample after PCA processed:  2
```

3. Normalization:

```
before normalization:  [[-3.13492809e+16 -5.91499717e+13]
 [-3.20128802e+16  1.08557068e+14]
 [-3.16820878e+16  1.48891516e+13]
 ...
 [-3.03882580e+16 -1.06589444e+14]
 [-3.20128802e+16  1.08557068e+14]
 [-3.20128802e+16  1.08557066e+14]]
```

```
after normalization:  [[6.56229130e-04 1.85101065e-01]
 [1.43635104e-15 1.85279738e-01]
 [3.27118489e-04 1.85179946e-01]
 ...
 [1.60657844e-03 1.85050524e-01]
 [1.43635104e-15 1.85279738e-01]
 [5.35895633e-12 1.85279738e-01]]
```

As we can see, it is not convenient for me to analyze the difference between DBSCAN and Kmeans from the algorithm principle perspective because the order of the value is up to 16 and it is very hard for us to choose the parameters of DBSCAN. So I use MinMaxScaler to scale the feature to the range (0,1)

4. Parameter Chosen and Visualize their clustering on test dataset:

For Kmeans algorithm, I choose n_clusters = 3, because there are total three different type of

data. And this is the knowledge we know beforehand.

Then I try different value for DBSCAN parameters eps and min_samples. For eps, I change from 1 to 0.0001. Firstly, I want to cluster the data into 3 clusters, I decrease eps from 1 to around 0.0001. Then to get better results, I increase from 0.0001 to 0.0005. At last, I choose 0.0005 for eps. As for min_samples, because the number of samples in the test dataset is 5000, I change the value from 120 to 80 to get better results when the DBSCAN can cluster the dataset into 3 clusters.

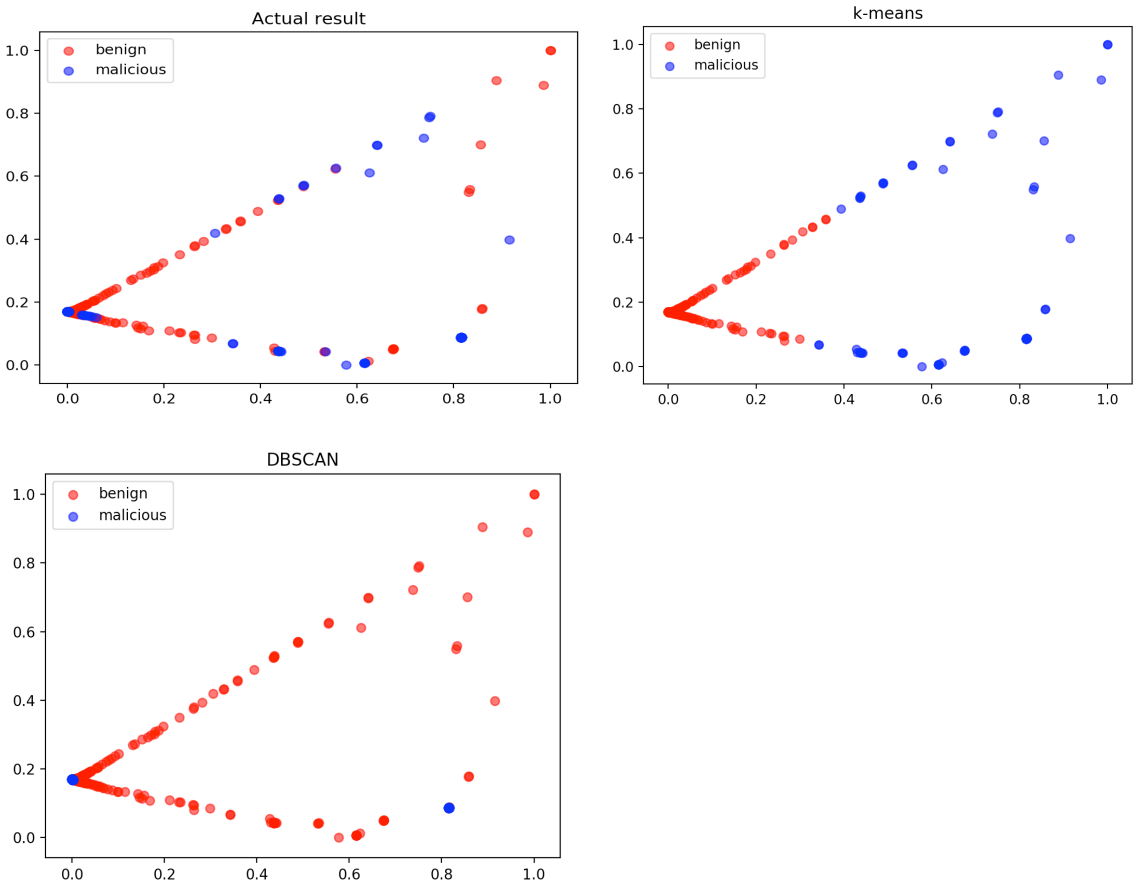

Figure 1: visualize their performance on the test dataset

When I firstly see the results, I think that the Kmeans could perform better for this test dataset; however, when we observe the data whose x range is between 0 and 0.01 and y range is between 0 and 0.5, I find that there are much more data in that range and DBSCAN clusters almost all malicious data in that xy range; therefore, to get a more obvious difference between the performance of Kmeans and DBSCAN, I zoom in the Figure1 to observe the results in x~(0, 01), y~(0,0.5)

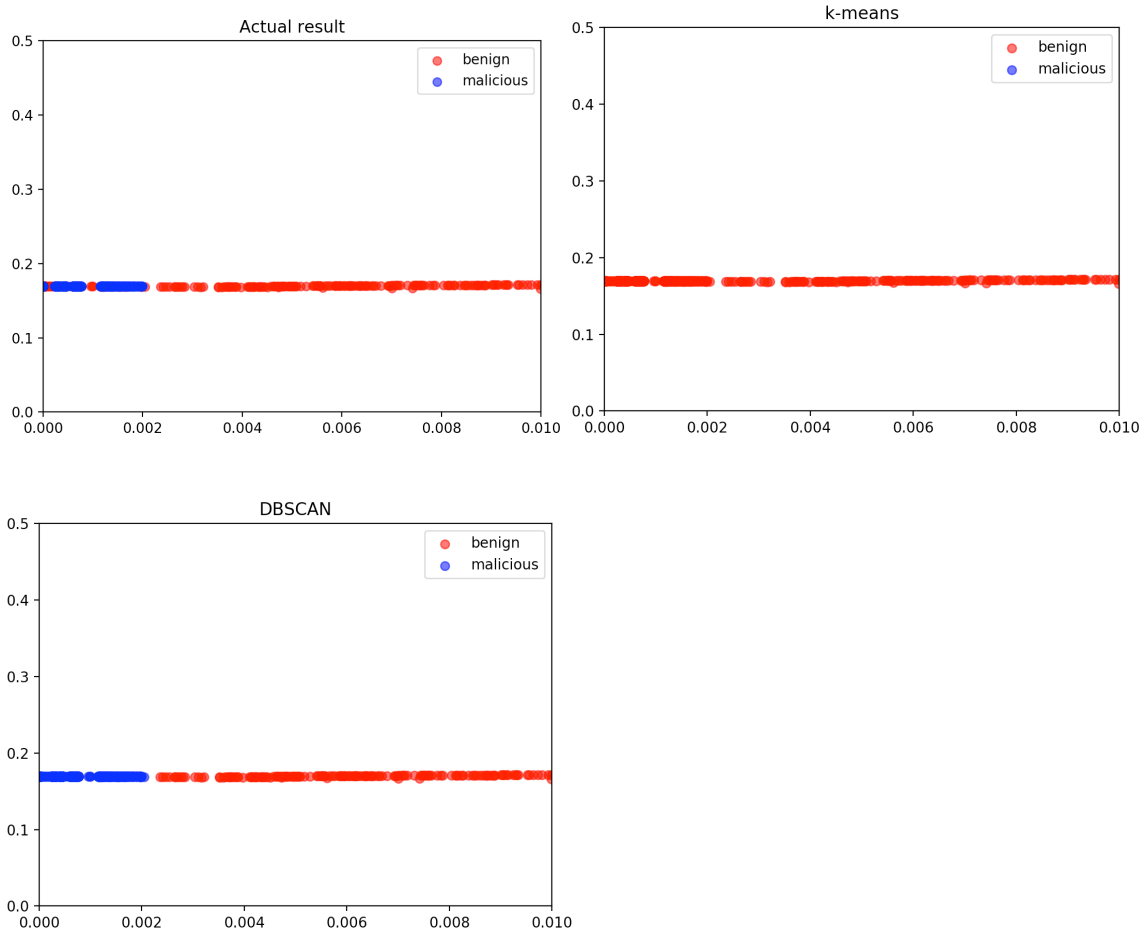

Figure 2: Zoom in results

From Figure2, we can get that DBSCAN has a better performance in this xy range while Kmeans cannot detect the malicious data in this xy range. Then from Figure 2 we can see how the density-based algorithm cluster the data because in the DBSCAN results of Figure2, there is a clear gap between the benign data and the malicious data; therefore, we can see that DBSCAN discovers regions of high density separated by-not-dense region, which is the principle of DBSCAN algorithm.

As for the Kmeans, it is a distanced based algorithm. From Figure1, because Kmeans seeks to minimize the average squared distance between points in the same cluster, we can see that when x is in range(0.4, 1), it can cluster some data into correct clusters, which performs better than DBSCAN in this range.

```
error_kmeans_rate:  0.6514
error_DBSCAN_rate:  0.2692
```

```
kmeans false positives:  25
kmeans true positives:  166
kmeans confusion matrix
[[1577   25 1602]
 [3232  166 3398]
 [4809  191 5000]]
```

```
DBSCAN false positives:  1259
DBSCAN true positives:  3311
DBSCAN confusion matrix
[[ 343 1259 1602]
 [  87 3311 3398]
 [ 430 4570 5000]]
```

Figure 3: Error rate and confusion matrix

From Figure 3, we can see that Kmeans method gets a very high error rate because it cannot cluster correctly the data with a very low x value, while DBSCAN can cluster this part of data correctly; however, Kmeans algorithm performs better than DBSCAN when x is greater than around 0.4. Seen from the confusion matrix of the Kmeans and DBSCAN, we can get that DBSCAN gets a higher false positives 1259 compared with Kmeans method is 25; therefore, we can get that DBSCAN clusters more normal flows as anomalous flow. But Kmeans has a higher true negatives which is 4809. So Kmeans clusters more anomalous flow into the normal cluster, and this has a more negative on our detection of the anomalous flow.

Also we can discuss why Kmeans has a such bad performance with 65% error rate. Firstly, this dataset we cannot see a convex shape for these two classes of data distribution, and this dataset is obviously not the similar density problem. And because I split more malicious data than the benign data, this is also not a similar size problem. These contribute to this bad performance.

Conclusions and Future

From the above discussion, we conclude that for the given dataset, when the x lower than around 0.4, DBSCAN has a better clustering result, and when the x higher than around 0.4, Kmeans performs better. Then from the perspective of the whole dataset, DBSCAN has a much lower error rate than Kmeans. Until now analysis, supervised learning method is much better performance than the unsupervised learning method; However, from the results I get, I come up with a future model using unsupervised learning algorithm, which is combine the Kmeans and DBSCAN algorithm because DBSCAN performs better when x value is small, while Kmeans perform better when x values is large; Because the pages limit and time limit, I will not conduct this experiment in this term project. I only propose this for feature, then we can get a model with the benefit of unsupervised learning methods,which achieves a relative higher accuracy.

References:

1. Gartner report on Internet of Things, http://www.gartner.com/ technology/research/internet-of-things
2. Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," in IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, doi: 10.1109/MPRV.2018.03367731.
3. Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, and Y. Elovici 'N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders', IEEE Pervasive Computing, Special Issue - Securing the IoT (July/Sep 2018).
4. Improved Network Traffic Classification Using Ensemble Learning. / Possebon, Isadora; da Silva, Anderson; Granville, Lisandro; Schaeffer-Filho, Alberto ; Marnerides, Angelos. IEEE Symposium on Computers and Communications (ISCC) 2019. IEEE, 2020. p. 1-6.
5. https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT#
6. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al., 2017. Under- standing the Mirai botnet. In: 26th USENIX Security Symposium, vol. 17. USENIX Security, pp. 1093e1110.
7. C. Kolias et al., “DDoS in the IoT: Mirai and Other Botnets,” Computer, vol. 50, no. 7, 2017, pp. 80–84.
8. E. Bertino and N. Islam, “Botnets and Internet of Things Security,” Computer, vol. 50, no. 2, 2017, pp. 76–79.
9. Computer Networking: A Top-Down Approach, Jim Kurose & Keith Ross, Addison-Wesley, 7th.
10. Arthur, David, and Sergei Vassilvitskii. “k-Means++: The Advantages of Careful Seeding.” Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2007. 1027–1035. Print.
11. R. Gutierrez-Osuna, "Pattern analysis for machine olfaction: a review," in IEEE Sensors Journal, vol. 2, no. 3, pp. 189-202, June 2002, doi: 10.1109/JSEN.2002.800688.