GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Institut für Informatik

PRONAYA PROSUN DAS

Anomaly detection using unsupervised algorithms

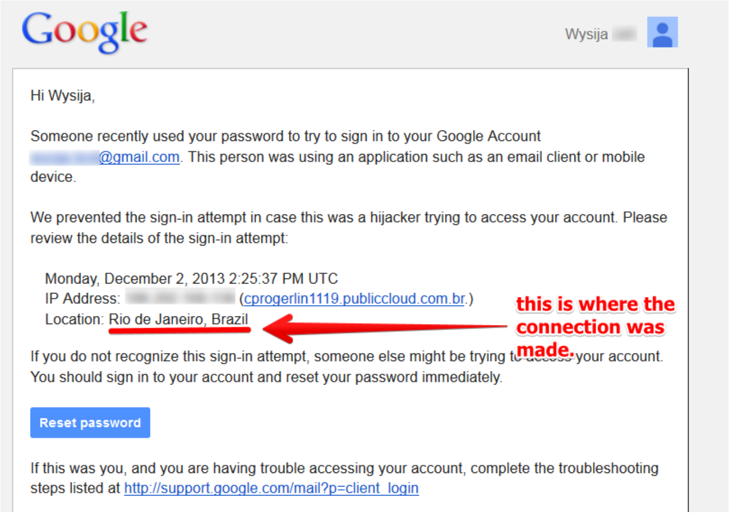
Outline

- Introduction
- Datasets
- Traffic Anomaly Detection Using Auto-encoder
- Intrusion Anomaly Detection with target encoding and Auto-encoder
- Intrusion Anomaly Detection with Data Engineering and Isolation Forest
- Conclusion
- Future Work

Introduction

- Anomaly detection is the process of identifying suspicious events or observations from the data.
- Suspicious events or observations differ from the majority of the data.
- Examples:
  - Unexpected very high traffic in a network
  - Intrusion into a System

Introduction(I)



Google

Wysija

Hi Wysija,

Someone recently used your password to try to sign in to your Google Account @gmail.com. This person was using an application such as an email client or mobile device.

We prevented the sign-in attempt in case this was a hijacker trying to access your account. Please review the details of the sign-in attempt:

Monday, December 2, 2013 2:25:37 PM UTC
IP Address: (cprogerlin1119.publiccloud.com.br.)
Location: Rio de Janeiro, Brazil

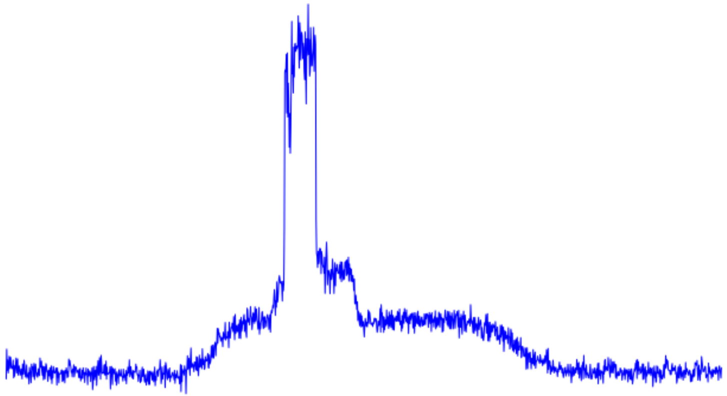
If you do not recognize this sign-in attempt, someone else might be trying to access your account. You should sign in to your account and reset your password immediately.

Reset password

If this was you, and you are having trouble accessing your account, complete the troubleshooting steps listed at http://support.google.com/mail?p=client_login

Introduction(II)

- Anomaly is a pattern in Data that is not confirmed the behavior of the system



Anomaly detection methods

- Supervised Learning Techniques.
- **Unsupervised Learning Techniques.**
- Semi-Supervised Learning Techniques.

Complexity of the Data

File Name	Size	Entries	Columns
auth.txt	73.4GB	1,051,430,459	9
dns.txt	812.7MB	40,821,591	3
flows.txt	5.2GB	129,977,412	9
proc.txt	15.4GB	426,045,096	5
redteam.txt	23.0kB	749	4

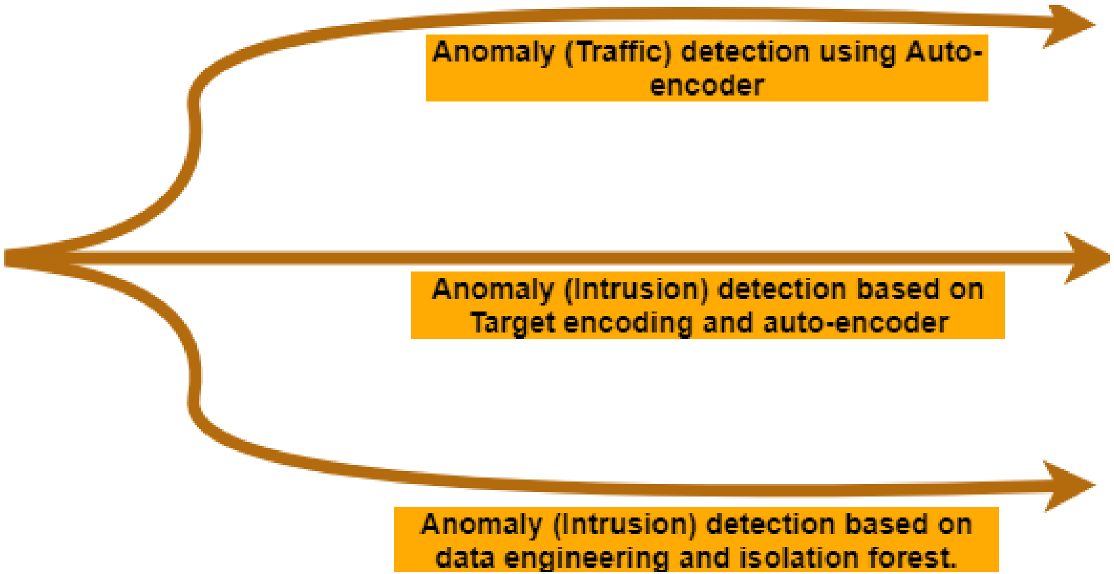
Complexity of the Data

- We choose to work with “auth.txt” file.
- Sequential read is not good for this large volume of data.
- Our approach is to split the data into several part and work with each part separately.
- Extract the features from each part and combine them at the end.
- We utilize multithreading paradigm to make data processing faster.
- All the initial jobs are handled using a cluster computer.

Splitting Data

- We create 6 threads to handle the data from 6 different position
- Each thread divide the data into 59 chucks.
- So we have 59*6=354 chunks in total.
- Each chunk contains 3000000 (3M) entries and size is around 200MB.

Our works – 3 flows



Flow 1

Anomaly (network traffic) using Auto-encoder

Extracting service counts from each chunk

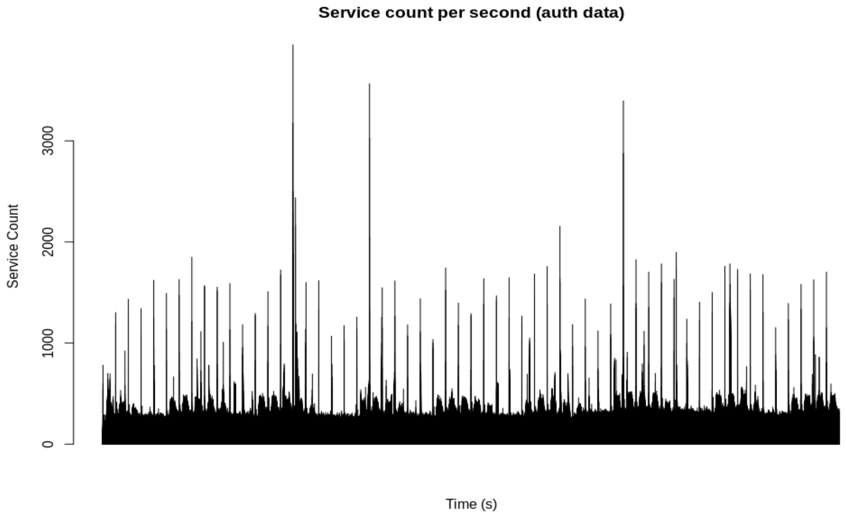
- Our goal is to find how many authentications (service counts) have been performed per second.
- We calculate service counts from each chunk of data.
- Finally merge the service count all together and save it to a file.
- The file size is reduced to 59 MB.
- Further processing is done on this file and can be processed using our personal computer.
- Extraction is also performed in a multithreaded way using python.

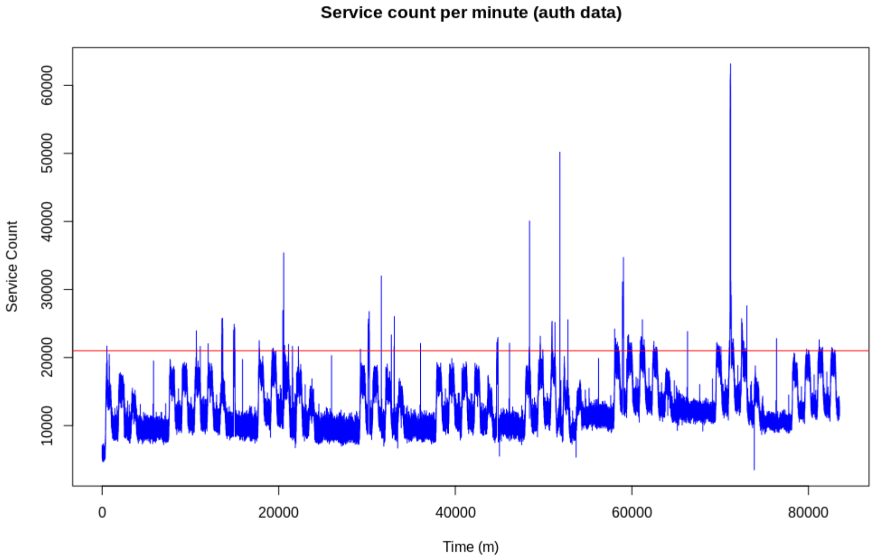
Further Feature Extraction

- We calculate service counts per minute.
- Also find out the standard deviation, variance, max, min and mean of the count per minute. These will be used as features.
- A snapshot of the extracted features are given below.

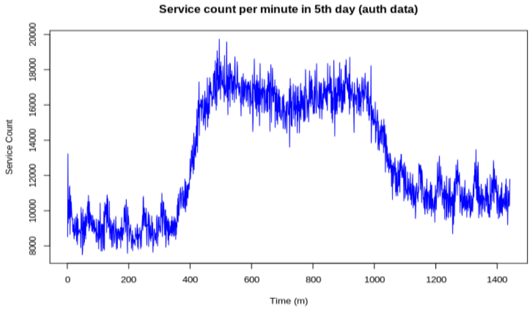
time_min	count	sd	var	mean	min	max	sec_from	sec_to
1	6625	128.35848	16475.8983	112.28814	43	781	1	59
2	5295	20.36832	414.8686	88.25000	53	133	60	119
3	6425	22.54735	508.3828	107.08333	54	152	120	179
4	5954	22.84017	521.6734	99.23333	51	169	180	239
5	6729	25.09360	629.6890	112.15000	47	173	240	299
6	6569	24.82515	616.2879	109.48333	63	172	300	359
7	5749	23.13812	535.3726	95.81667	51	156	360	419
8	7133	29.56469	874.0709	118.88333	68	190	420	479
9	6133	34.64693	1200.4099	102.21667	41	208	480	539
10	5905	23.17896	537.2641	98.41667	47	151	540	599
11	6682	24.88139	619.0836	111.36667	74	196	600	659

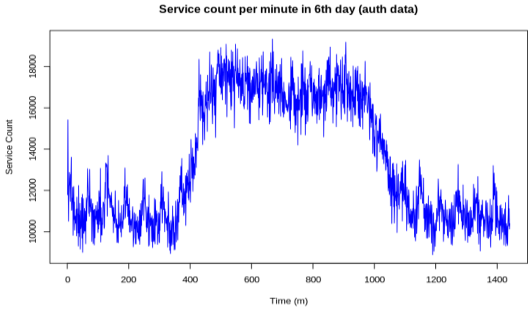
Some plots


Service count per second (auth data)
Service Count
0
1000
2000
3000
Time (s)

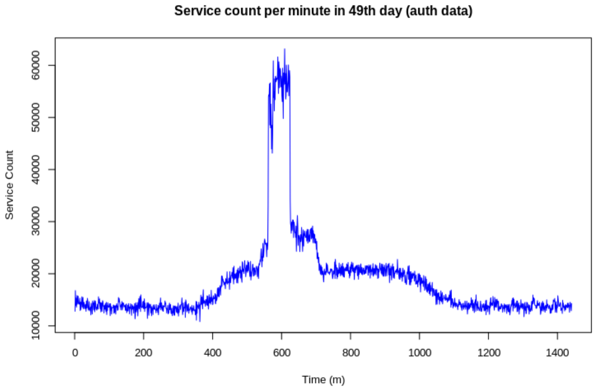

Service count per minute (auth data)
Service Count
10000
20000
30000
40000
50000
60000
0
20000
40000
60000
80000
Time (m)

Some plots





- Normal pattern.
- High counts during day time.
- Counts are between 8000 to 20000



- Could be anomaly.
- Sudden pick of counts, more than 50000.
- For a certain amount of time.

Typical Auto-encoder

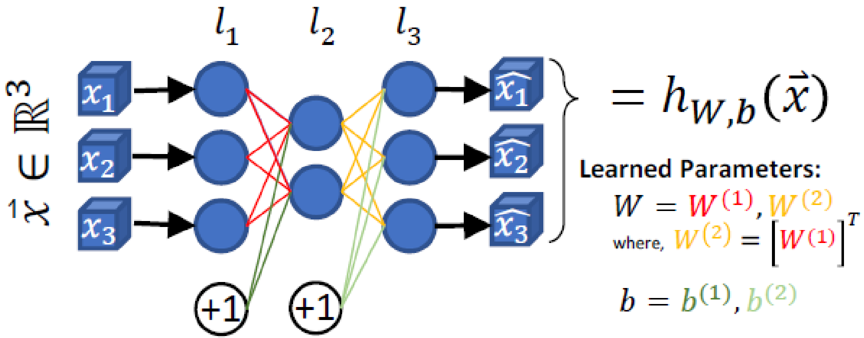


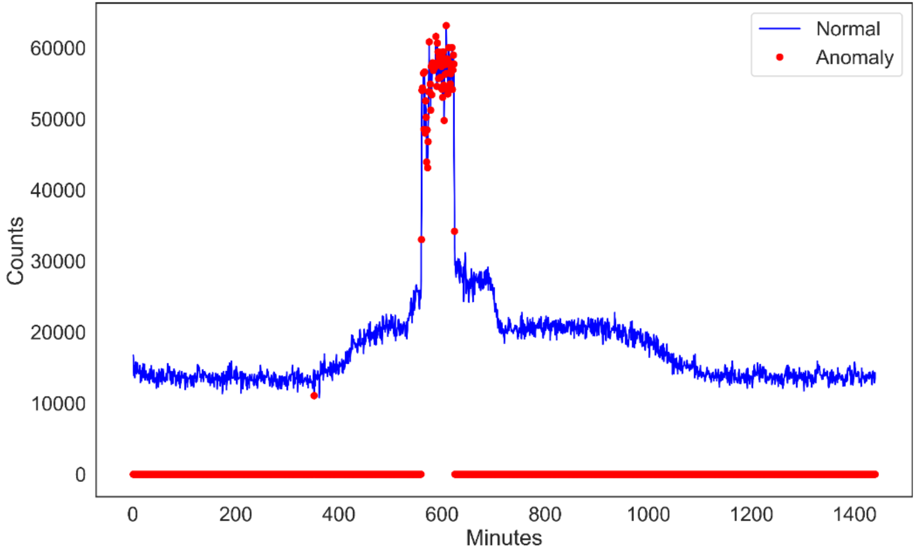
Fig.: An example autoencoder with one compression layer, which reconstructs instances with three features.

- We implemented auto-encoder using keras.

Training and Testing:

- Training Phase:
  - Train an auto-encoder on clean (normal)
  - We select the days which has same kind of pattern.
  - **Threshold = mean(reconstruction error for train set)/4**
  - Here, reconstruction error is the MSE between predicted and normal data.
- Testing Phase:
  - For the testing, we choose day 49, we get predicted data from auto-encoder.
  - We calculate reconstruction error (MSE between predicted and anomaly data).
  - **If(reconstruction error for test set > Threshold) –> labeled as anomaly**

Result of Flow 1:



- Tested on 49th day
- Red points are indicating traffic anomaly.
- As, there no label for this kind of anomaly, So exact accuracy can’t be measured.
- But our goal is to implement an algorithm that can detect this kind of anomaly.

Flow 2

Anomaly (intrusion) detection based on direct pattern and Auto-encoder

Data reduction

- Self authentication is removed.
  - Example: If the source and destination has the same computer, then it is removed.
- We remove columns named **authentication type, logon type and authentication orientation**.

time_sec	src_dom	des_dom	src	des	auth
691201	ANONYMOUS LOGON@C1208	ANONYMOUS LOGON@C1208	C3076	C1208	1
691201	ANONYMOUS LOGON@C457	ANONYMOUS LOGON@C457	C13130	C457	1
691201	ANONYMOUS LOGON@C586	ANONYMOUS LOGON@C586	C10251	C586	1
691201	ANONYMOUS LOGON@C586	ANONYMOUS LOGON@C586	C13709	C586	1
691201	ANONYMOUS LOGON@C586	ANONYMOUS LOGON@C586	C16581	C586	1
691201	ANONYMOUS LOGON@C586	ANONYMOUS LOGON@C586	C6631	C586	1
691201	ANONYMOUS LOGON@C586	ANONYMOUS LOGON@C586	C7298	C586	1
691201	ANONYMOUS LOGON@C625	ANONYMOUS LOGON@C625	C13406	C625	1
691201	C10208$@DOM1	C10208$@DOM1	C10208	C492	1
691201	C10251$@DOM1	C10251$@DOM1	C10251	C586	1
691201	C111$@DOM1	C111$@DOM1	C111	C625	1

Target-based Encoding

Source	Count	Source encoded (Probability)
A1212	3	3/(1+1+1+1+3)=0.429
C1234	1	1/(1+1+1+1+3)=0.143
A1212	3	0.429
B1111	1	0.143
D2222	1	0.143
A1212	3	0.429
A2222	1	0.143

After Encoding

time_sec	src_dom	des_dom	src	des	auth
1081855	0.000109691	0.000109691	0.00018584	0.0744736	1
1065386	0.000184026	0.000184026	0.000365333	0.0161309	1
1101757	0.000123289	0.000122382	0.000260175	0.014646	1
1059193	3.80744e-05	3.80744e-05	6.6177e-05	0.0730078	1
1057365	4.26071e-05	4.26071e-05	6.88966e-05	0.0816525	1
1108412	0.000106065	0.000106065	0.000135074	0.0191088	1
1103476	4.44202e-05	4.44202e-05	5.2579e-05	0.0744736	1
1043335	0.00646087	0.00646087	0.0077273	0.00135618	1
1072738	1.99438e-05	1.99438e-05	6.07378e-05	0.0816525	1
1112286	2.44764e-05	2.44764e-05	6.6177e-05	0.0236923	1
1040511	8.43077e-05	8.79338e-05	0.000118756	0.0744736	1

Selected algorithm

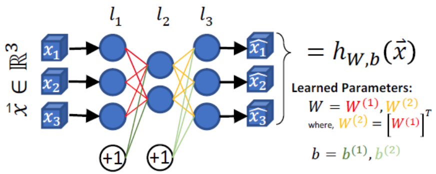


Fig.: An example autoencoder with one compression layer, which reconstructs instances with three features.

- We use auto-encoder same as Flow 1.
- Auto-encoder is trained on normal data.
- We took 10% from a day data to train the auto-encoder (Only normal data).
- Test data contains mix of normal and anomaly data.
- We did the same test for five different draw and then find the average.
- Threshold = mean(reconstruction error for train set)

Result of Flow 2

Day	Avg. Accuracy (%)	Avg. True Positive Rate (%)	Avg. False Positive Rate (%)	Avg. False Negative Rate (%)	Number of actual anomaly
8	79.49	100	20.58	0	273
12	82.09	98.55	18.94	1.45	209
13	80.97	88.12	19.29	11.88	81
26	82.12	100.0	17.87	0	26

Flow 3

Anomaly (Intrusion) detection based on data engineering and Isolation forest

Data reduction

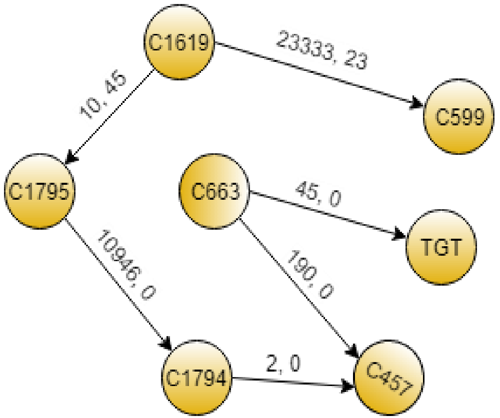
- Self authentication is removed.
    - Example: If the source and destination has the same computer, then it is removed.
- By this way, 60% reduction is achieved.
- Further, for our approach, we don't need **source domain, destination domain, authentication type, logon type, authentication orientation**. So these columns are removed.
- At the end, each chunk is reduced to around 34-40MB.

time_sec	src	des	auth	id
691201	C3076	C1208	1	1
691201	C13130	C457	1	2
691201	C10251	C586	1	3
691201	C13709	C586	1	4
691201	C16581	C586	1	5
691201	C6631	C586	1	6
691201	C7298	C586	1	7
691201	C13406	C625	1	8
691201	C10208	C492	1	9
691201	C10251	C586	1	10
691201	C111	C625	1	11
691201	C1115	C1114	1	12

Data mining (engineering) step 1

- We try to fine the relation between source and destination.
- The number of times a computer is connected to another computer and the number of failed attempts.

src	des	times	failed	id_list
C1619	C599	23333	23	202 317 349 455 523 575 …
C1795	C1794	10946	0	203 1569 2569 7257 7359 75…
C663	C457	190	0	204 205 259 8465 10631 1…
C663	TGT	45	0	207 214573 216116 26901…
C926	C528	210	0	208 285 456 1378 4008 40…



Data mining (engineering) step 2

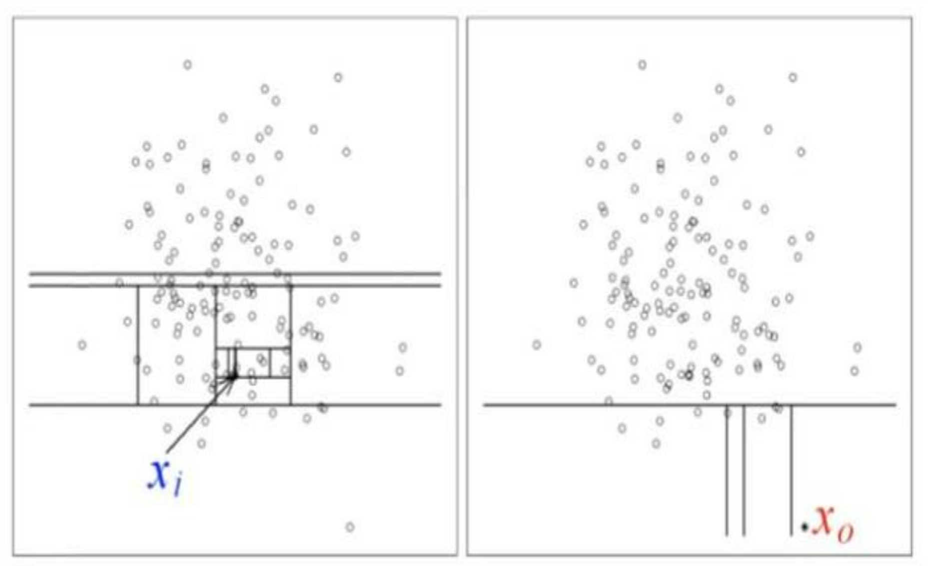
- Now we find out the number of destinations for a source computer, their total connection and total failed attempts.
- des_list contains all the destination computer, con_per_des is average connection per destination.

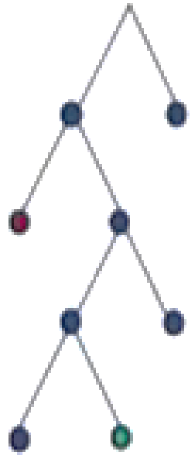
src	total_con	num_of_des	total_failed	des_list	id_list	con_per_des
C1798	91581	6865	108	C718 C18401 C558 C19217 …	69 478939 992997 15905…	0.074961
C1521	58575	5703	3286	C13246 C625 C1065 C586 C…	18 3137643 3202856 3403…	0.0973624
C17693	527	311	199	C10171 C1654 C4747 C1893 …	2875444 3686937 3182…	0.590133
C5802	1793	241	265	C21838 C21857 C21174 C1899…	20416 20607 20672 20778 …	0.134412
C5808	1785	240	243	C20347 C8705 C2689 C10508…	3144 4546 4720 75202 7…	0.134454
C10	1080	207	0	C586 C528 C625 C2106 C…	24344 26618 103748 17807…	0.191667

Isolation Forest as an unsupervised algorithm

- Isolation Forest (IF) is used to split the table into 2 part, where 1 part contains intruder data and another normal data.
- IF is built on the basis of decision trees.
- In IF partitions are created by first randomly choosing a feature.
- And then choosing a random split value between the min and the max value of the chosen feature.
- IF builds an ensemble of random trees for the given data set
- Anomalies are the points with shortest average path length from the root.
- No profiling of normal instances, No point based distance calculation.

Anomaly vs Normal





Isolation Forest as an unsupervised algorithm

- In this method an anomaly score is required for decision making:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Where, h(x) is the path length of observation x,

c(n) is the average path length of unsuccessful search in a Binary Search Tree,

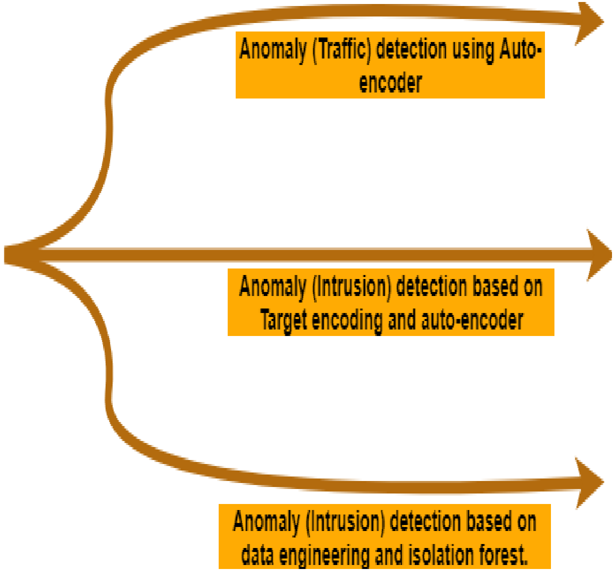
n is the number of external nodes.

- Threshold = 0.0005
- if(S(x, n)>Threshold) -> Anomaly

Result of Flow 3

Day	Accuracy (%)	True Positive Rate (%)	False Positive Rate (%)	False Negative Rate (%)	Number of actual anomaly
8	99.27	95.60	0.72	4.396	273
12	99.32	99.04	0.68	0.96	209
13	98.41	100.0	1.59	0	81
26	93.72	100.0	6.27	0	26

Conclusion



Detecting Traffic Anomaly that can help the administrator to examine a period of time that system has an irregular behavior.

With the help of Auto-encoder, and vectorised selected column(without special Feature Engineering) to detect anomalous users.

Maximum Accuracy with the help of Data mining and Isolation Forest Algorithm.

Conclusion

Flow 1 (Traffic anomaly):

- Auto-encoder is applied on normal data (authentication per minute).
- Exact accuracy cannot be measured due to lack of labeled data.

Flow 2 (Intrusion anomaly):

- Auto-encoder is applied directly on encoded features.
- Average result is measured.

Flow 3 (Intrusion anomaly):

- Specific information are extracted .
- We use isolation forest.
- Higher accuracy is achieved.

Future Work

- Currently Working on: LSTM Auto-encoder.
  - Expected to have High Accuracy based on similar Models.
  - Minimize the effort for feature engineering ,
  - Real time processing.
- Test few other encoders in Flow 2.

Thank You For your Attention.

Questions?