COMP90073-Security Analytics Assignment 2 Report

Machine learning based cyberattack detection

Daniel Gil <Student Id: 905923>

Contents

1 Introduction

This report is organized as follows: In Section: 2, it is presented an overview of the data including the main characteristics and insights obtained using general Splunk queries to discover suspicious patterns and how this analysis drives features identification and extraction. From there, in section 3 the method to generate and extract the features is described based in literature review and a combination of Splunk queries and python data manipulation. The next section, 4, presents machine learning models to identify attacks, the parameters and environment used in the experiments with the score and threshold used to classify anomalies. Following that, in section 5 a discussion is presented regarding scalability for real-time detection attacks. Next in section 6 the best model is described and conclusions presented.

2 Exploratory Data Analysis

2.1 Dataset

The dataset denote network traffic from two days. In sources files, Day 2 is for the Normal Day and TestData represents traffic from attack day. The query 1 shows a summary of events in dataset. It is important to note that, when queries are referenced and results present both days, only one day query is described, to replicate the query for the other day it is necessary just to change the source to include the correct file.

```
(source="*day2_*.pcap.csv" OR source="*TestData.pcap.csv") date_year=2017
| stats count as num_packets earliest(_time) AS starttime, latest(_time) as endtime  by source
| eval start=strftime(starttime,"%Y-%m-%d %H:%M:%S.%Q")
| eval finish=strftime(endtime,"%Y-%m-%d %H:%M:%S.%Q")
| eval duration=endtime-starttime
| sort -time
| table source, start, finish, duration, num_packets
```

Listing 1: Summary of data

Table 1: List of datasets

source	start	finish	duration	num_packets
TestData.pcap.csv	2017-07-07 17:00:00.014	2017-07-07 20:02:41.169	10961.155027	3591186
day2_1.pcap.csv	2017-07-03 12:28:24.847	2017-07-03 12:34:53.041	388.194044	800000
day2_10.pcap.csv	2017-07-03 12:19:43.227	2017-07-03 12:28:24.847	521.619748	800000
day2_2.pcap.csv	2017-07-03 12:34:53.041	2017-07-03 12:37:58.680	185.638605	800000
day2_3.pcap.csv	2017-07-03 12:37:58.680	2017-07-03 12:40:57.442	178.762204	800000
day2_4.pcap.csv	2017-07-03 12:40:57.442	2017-07-03 12:49:26.095	508.653073	800000
day2_5.pcap.csv	2017-07-03 12:49:26.095	2017-07-03 13:18:37.636	1751.540234	800000
day2_6.pcap.csv	2017-07-03 13:18:37.636	2017-07-03 14:07:37.609	2939.973629	800000
day2_7.pcap.csv	2017-07-03 14:07:37.609	2017-07-03 14:52:23.189	2685.579320	800003
day2_8.pcap.csv	2017-07-03 11:55:58.598	2017-07-03 12:16:41.882	1243.284236	800000
day2_9.pcap.csv	2017-07-03 12:16:41.882	2017-07-03 12:19:43.227	181.344768	800000

The import process was made automatically by splunk and some mistakes were considered for further queries. E.g. for day1_5 splunk loaded events with _time in 2019, then a filter by year is included in queries.

Since datasets present different behavior it is convenient to understand each one of them and compare against each other to identify abnormal behavior. Splunk has a dashboard that gives a summary of important data in dataset according to packets and flow.

2.2 Top metrics

A data exploratory analysis was made from splunk to identify the most relevant changes in network traffic patterns. In figure 2, during attack day TCP Protocol accounts for the majority

of the traffic, followed by commonly used protocols like HTTP and DNS. In contrast, during normal day in figure 1 protocol HTTP has a different proportion being one of the less used. Following this pattern, the figure 3 shows a significant change of proportion during the attack day compared to figure 4 showing the normal day port traffic.

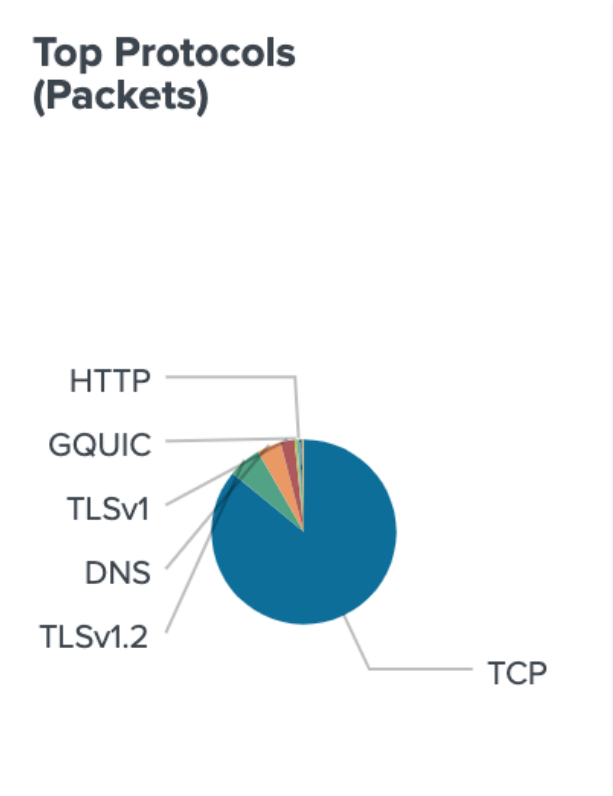


Figure 1: Top Protocols (packets) in Normal Day

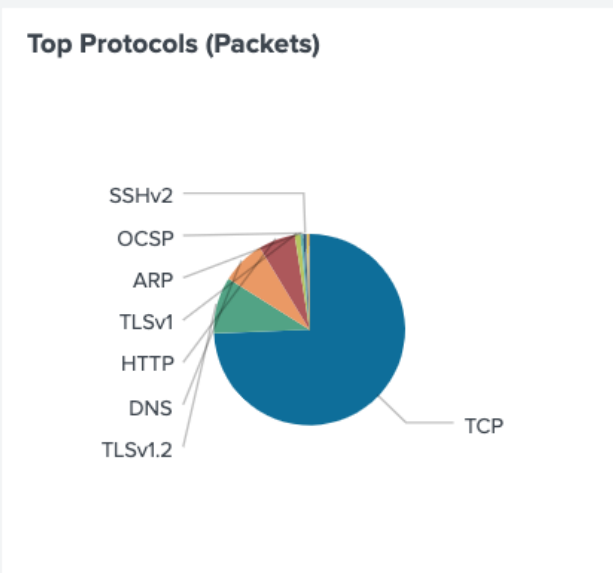


Figure 2: Top Protocols (packets) during Attack Day

During the attack day, there are significant amount of conversations involving external hosts with the host 192.168.10.50 as shown in 5 which represents a departure from normal behavior

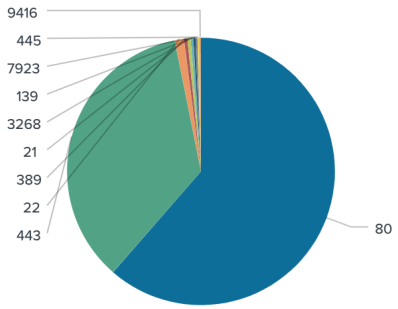


Figure 3: Top Ports during Attack Day

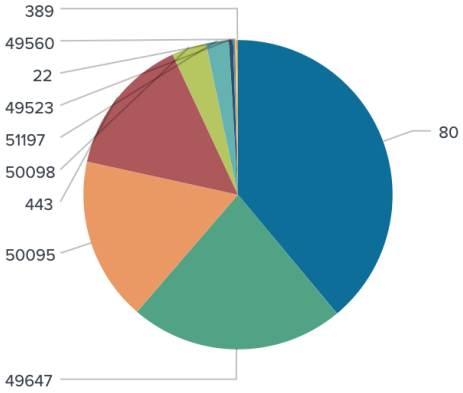


Figure 4: Top Ports during Normal Day

represented in 6. The proportion of the number of conversations with this host is also to highlight, as apparently, several hosts are constantly sending and receiving packets involving conversation with host 192.18.10.50 in a similar proportion. Based on results from Assignment 1 where conclusions led to identify a possible DoS attack to host 192.168.10.50, the attack dataset analyzed in this report suggests a distributed DoS (DDoS) as per the number of hosts interacting in similar patterns.

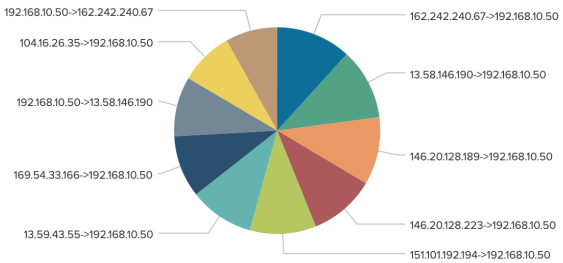


Figure 5: Top Conversations during Attack Day

During the attack day, the figures 7 and 8 show the traffic for the most prominent destinations IP and source IP respectively. As shown in conversation figures before, the host 192.168.10.50 accounts for the majority of the traffic sugesting abnormal network behaviour.

If we consider time and bytes transferred, the figure 9 shows concentrated interactions

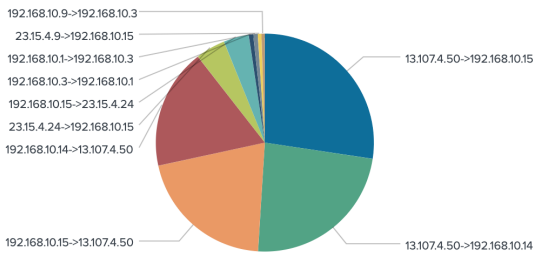


Figure 6: Top Conversations during Normal Day

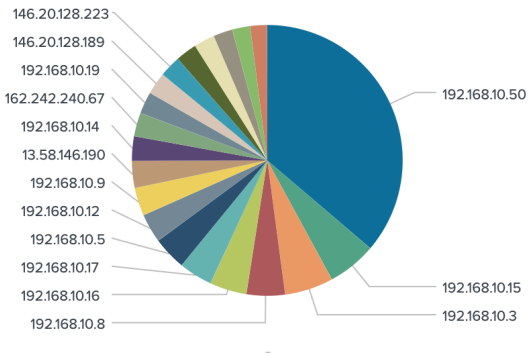


Figure 7: Top Destination during Attack Day

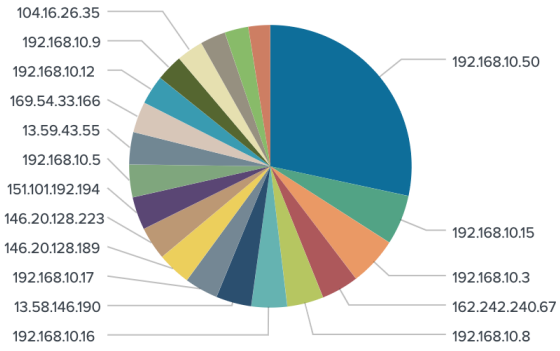


Figure 8: Top Source during Attack Day

from 6:56 PM to 7.15 PM, particularly it can be seen data transferred concentrated in host 192.168.10.50 during the same time in a constant manner as shown in figure **??** and several hosts transferring data during the same time in small intervals in scattered communications across the same time window in figure 11.

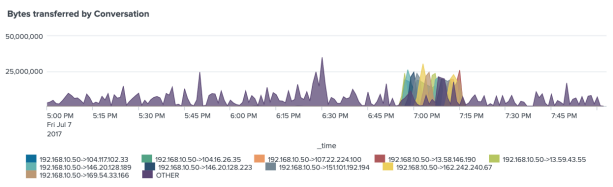


Figure 9: Bytes transferred during Attack Day by Conversation

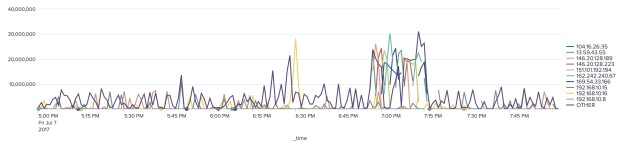


Figure 10: Bytes transferred during Attack Day by destination IP

Figure 11: Bytes transferred during Attack Day by source IP

2.3 Feature identification

In summary, the test dataset shows significant deviations in:

- Traffic between hosts,
- data transferred,
- ports used,
- protocols involved and,
- intensity of conversations.

Considering this deviations, in this report is formulated a set of features to use with existing Machine Learning models to identify possible attacks motivated by how features related to this deviations can be sufficent to describe the data in such a way that anomalies can be identified. Some features are generated directly from splunk queries while the rest are generated using transformations and more complex data processing algorithms in python.

The feature set is comprised in four categories:

- **Basic features:** Collected from the packet header incuding protocol types, services, TCP flag, source and destination flags
- **Time based features:** An example could be number of data transferred during a time a window.
- **Connection-based features:** Features that are computed over an historical packet transmission, for example, the number of packets from source to destination.

From Assignment 1, part of basic features were previously used as shown in table 2. The intuition behind feature selection for Assignment 1 was based on the results given by exploratory analysis. Increasing traffic involving two particular hosts and the amount of data transferred led the conclusions of DoS attack. The anomaly detection method used helped to identify abnormal cases based on extreme values, however, since just few features were used there were numerous false positives that got eventually discarded by comparing the results with previous analysis.

For Assignment 2 it is considered a more complex set of features that comprises not only basic but time-based and connection-based features. Two methods were used to generate the

Table 2: List of features Assignment 1

Feature name	Type	Description
Basic features		
start	String	Time when the connection started
finish	String	Time when the connection finished
src_ip	String	Source IP
dst_ip	String	Destination IP
num_dst_port	Number	Number of ports in destination
total_bytes	Number	Total bytes in the conversation
num_packets	Number	Total number of packets in the conversation
mbps	Number	Number of bytes per second

features, both of them use an extended query in Splunk to extract the data in conversations but a more fine-grained level. The query shown in 2 generates conversations in more fine-grained way than the one used in the first assignment. The list of features generated are shown in table 3. A second method is shown in section 3 to expand the set of features more than the basic level using the result of 2 as input and using Python to generate the rest.

```
source="/opt/splunk/etc/apps/SplunkForPCAP/PCAPcsv/TestData.pcap.csv" date_year=2017
| rex field=info "HTTP\/\d\.\d\s(?<status>\d{3})\s"
| rex field=info "(?<failure>) Fail*"
| rex "^[^\[\n]*\[(?P<flags>[^\]]+)"
| stats earliest(flags) as first_flag, latest(flags) as last_flag, count as total_packets, sum(tcp_length) as total_bytes
count(eval(if(status>=400, dst_ip, NULL))) as total_http_errors count(failure) as total_failures earliest(_time) as start_time,
latest(_time) as finish_time,sum(ack) as flag_ack sum(fin)
as flag_fin sum(psh) as flag_psh sum(rst) as flag_rst sum(syn) as flag_syn
by tcp_stream, protocol, src_ip, src_port, dst_ip, dst_port
| eval total_duration=finish_time-start_time
| eval flow_start = strftime(start_time, "%Y-%m-%d %H:%M:%S.%Q")
| eval flow_finish = strftime(finish_time, "%Y-%m-%d %H:%M:%S.%Q")
| eval flow_id = src_ip+":"+src_port+"->"+dst_ip+":"+dst_port
| eval pps=total_packets/total_duration
| eval bps=total_bytes/total_duration
| eval bpp=total_bytes/total_packets
| fillnull tcp_stream, src_ip, src_port, dst_ip, dst_port, protocol, start, finish, total_duration, total_packets, total_bytes, pps, bps,
bpp, total_http_errors, total_failures, flag_ack, flag_fin, flag_psh, flag_rst, flag_syn
| table tcp_stream, protocol, flow_id, flow_start, flow_finish, src_ip, src_port, dst_ip, dst_port, total_duration, start_time,
finish_time, total_packets, total_bytes, pps, bps, bpp, total_http_errors, total_failures, flag_ack, flag_fin, flag_psh, flag_rst, flag_syn,
first_flag, last_flag
```

Listing 2: Splunk feature generation

3 Feature generation/extraction

Several approaches can be considered to generate features for machine learning models. Alaidaron and Mahmuddin discuss techniques to identify intrusion detection based on packets and flows [1]. A packet-based Intrusion Detection System (IDS) analyzes the packet header and payload to detect possible threats based on predefined signatures. The method compares each packet and payload to a specific database of signatures and then classify the packet as malicious or not. The use of well-known signatures represent one of the key advantages of this method as it can be used to leverage a lower false positives rate. From table 1, the amount of packets to be analyzed using this approach is shown for both datasets, particularly for the test dataset the number of packets is 3.591.186. This number of packets requires significant computing power to analyze that makes not feasible for the scope of this report. In general for packet-based approach, the computational resources to analyze such volume of data, the speed of which packets can arrive to the system and the risk of loosing packets during transmission represent difficulties to perform inspections. Consider a DoS attack which generates more traffic than usual in a particular time frame, the increasing of traffic can overload the network in such a

Table 3: Uni-directional feature set (Extension of basic features from Assignment 1)

Feature name	Type	Description
Basic features		
tcp_stream	Number	Wireshark flow identifier
protocol	String	The protocol used in the conversation .
flow_id	String	Identifies IP addreses and ports
flow_start	String	Time when the connection started
flow_finish	String	Time when the connection finished
src_ip	String	Source IP
dst_ip	String	Destination IP
src_port	String	Source port
dst_port	String	Destination port
total_duration	Number	Total in seconds
start_time	Number	Timestamp when the connection started
finish_time	Number	Timestamp when the connection finished
total_packets	Number	Total number of packets in the conversation
total_bytes	Number	Total bytes in the conversation
pps	Number	Number of Packets per second
bps	Number	Number of bytes per second
bpp	Number	Number of bytes per packet
total_http_errors	Number	Number of http requests with status in error
total_failures	Number	Number of failures described in the packet info
flag_ack	Number	Total number flags ACK in the conversation
flag_fin	Number	Total number flags FIN in the conversation
flag_psh	Number	Total number flags PSH in the conversation
flag_rst	Number	Total number flags RST in the conversation
flag_syn	Number	Total number flags SYN in the conversation
First_flag	String	Flag of the first packet in conversation
Last_flag	String	Flag of the last packet in conversation

way that the current Intrusion Detection Systems (IDS) will not be able to capture all packets and therefore, the ability to identify an attack [2]. To overcome this issue, Papadogiannakis et al. present a method to select packets to discard in an effort for IDS to predict overloads and minimize the impact on attack detection. In contrast, a flow-based approach is discussed as an alternative by researchers to achieve similar accuracy with small amount of data. Instead of relying on signature-based attack patterns, it uses statistical techniques and aggregated network data to identify malicious flows. Hofstede et al. focus on the study of flows instead of packets to monitor networks [3], according to RFC 7011, a flow can be defined as "a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties". A first approach was already defined by the analysis of conversations previously in section 2. The splunk query 3 shows the number of conversations, 16,507 in total. Conversations represent packets flowing from a source IP to a destination IP, however, following the definition of flow, Alaidaron et al, Ring et al and many authors have used a set of properties shared by the packets as the five tuple [4]: Source IP address, source port, destination IP address, destination port and transport protocol. Using the splunk query 4 we can study the flows detailed in 655,216 data points instead of using packets. By default, the 5-tuple will capture unidirectional flows, since packets are flowing in both directions one can use a bidirectional flow to reduce the total amount of data. The impact of data to be analyzed is not the only motivation, using a bidirectional flow one can understand better the properties of the conversations and attacks such as the client/server or

attacker/victim [5]. In figure 12 a simple TCP flow is shown, from an observation point one can see two separate flows representing each of the directions as shown in figure 13. If we consider an environment where contextual information is not provided then a bidirectional flow can describe details of the attack like the amount of packets or bytes flowing in each of the directions as shown in figure 14. Using unidirectional approach, many researchers have used features similar like the ones used for assignment 1 and have been able to identify malicious activities [1]. In contrast, Minarik et al. have used bidirectional flows and compared results with unidirectional flows to improve host profiling quality. Bidirectional flows standard [5] can improve also the use of statistical methods in general because it allows to distinguish requests from replies, demonstrating advantages and benefits for security analysis [6] like scanning [7] and intrusion detection [8].

```
1 source="*TestData.pcap.csv*" | eval conversation=src_ip+"->"+dst_ip |stats count by conversation | stats count
```

Listing 3: Unidirectional flows

```
1 source="*TestData.pcap.csv*" | eval conversation=src_ip+"->"+dst_ip |stats count by src_ip,src_port,dst_ip,dst_port,protocol
2 | stats count
```

Listing 4: Unidirectional flows

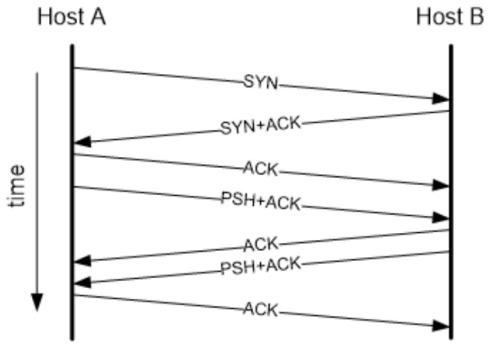


Figure 12: TCP Flow

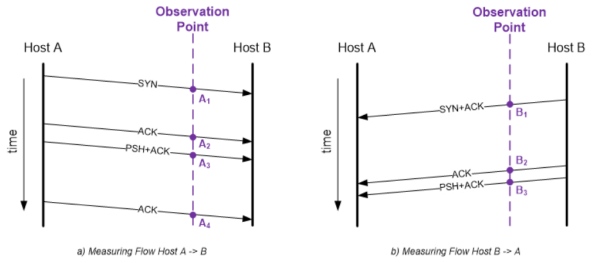


Figure 13: Unidirectional TCP Flow

The feature generation presented in this report uses both type of flows, first Splunk is used to generate unidirectional flows using the 5-tuple plus and an additional flow identifier generated by Wireshark (used to capture the packets). The query shown in 2 is used to generate features for uni-directional flows presented in table 3, then data is manipulated with python to extract bidirectional flows and create a new feature set capturing data for each of the flows.

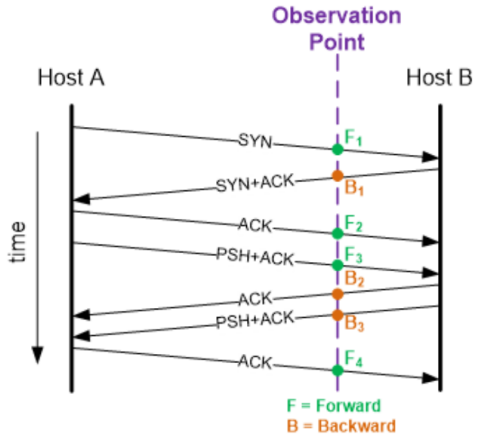


Figure 14: Bidirectional TCP Flow

Bhuyan et al. [9] [10] use time and connection based features to analyze TUIDS, NSL-KDD and KDD99cup99 datasets. They propose a new clustering technique (TreeCLUSS) with no requirement of labelled or signature-based data. Gogi [11] also presents two unsupervised clustering algorithms for anomaly detection using the same approach. In the experiments for anomaly detection presented in this report it is used a similar approach extending the feature set with additional statistics and rates. The feature for bidirectional flow is presented in table 4.

4 Anomaly Detection

It is considered an anomaly a data point that presents a deviation from the expected behavior [10]. It can be defined as a single event or a set of data points / events. Anomalies in network traffic data can show complex patterns which occur because of malicious activities in the network, overloading of networking infrastructure, malfunctioning of network devices or compromises when setting network parameters. According to the learning method, anomaly detection algorithms can be classified in two categories [12]: Unsupervised and supervised. When anomaly detection is performed using supervised learning, normal behavior and malicious behavior is labelled in datasets allowing machine learning models to classify new network connections as normal or abnormal [11]. In practice, unlabeled datasets predominate therefore unsupervised anomaly detection approaches became useful to analyze network traffic data. This report present results for several clustering algorithms used as unsupervised machine learning models with the datasets described in 1 and features generated from 3 and 4.

Each of the algorithms use data processed as follows: 1. Data is loaded from exported datasets generated by Splunk query 2 for uni-directional flow features or from object files (dataframes) in python generated previously to represent bi-directional flows, 2. Data is scaled and centered to avoid differences in variances because of the different units present in the datasets, and 3. A representation in two-dimensional space is created to create plots and visualize clusters using Principal Components Analysis. (This visualization is not used to identify outliers since it is not capturing a significant portion of the variance in data)

Table 4: Bi-directional feature set

Feature name	Type	Description
Basic Features		
src_ip	String	Source IP address
dst_ip	String	Destination IP address
src_port	String	Source port
dst_port	String	Destination port
fwd_packets	Numeric	Number of packets src ->dst
bwd_packets	Numeric	Number of packets dst ->src
fwd_bytes	Numeric	Number of bytes src ->dst
bwd_bytes	Numeric	Number of bytes dst ->src
fwd_duration	Numeric	Duration of the flow
flow_start	Datetime	Start time of start flow
fwd_bps	Numeric	Flow src->dst bytes per second
fwd_pps	Numeric	Flow dst>src packets per second
fwd_bpp	Numeric	Flow src>dst bytes per packet
fwd_total_http_errors	Numeric	Flow total http errors
fwd_total_failures	Numeric	Flow failures
tcp_stream	Numeric	Flow ID from wireshark
protocol	String	Protocol associated with flow
fwd_flag_syn	Numeric	Number of SYN flag in flow
fwd_flag_ack	Numeric	Number of ACK flag in flow
fwd_flag_fin	Numeric	Number of FIN flag in flow
fwd_flag_psh	Numeric	Number of PSH flag in flow
fwd_flag_rst	Numeric	Number of RST flag in flow
first_flag	String	First flag of the flow
fwd_avg_bytes	Numeric	Average bytes in flow
fwd_stdev_bytes	Numeric	Std deviation of bytes in flow
fwd_min_bytes	Numeric	Minimum bytes in flow
fwd_max_bytes	Numeric	Maximum bytes in flow
bwd_duration	Numeric	Duration of bytes in flow
flow_finish	Datetime	Finish time
bwd_bps	Numeric	Flow dst->src bytes per second
bwd_pps	Numeric	Flow dst->src packets per second
bwd_bpp	Numeric	Flow dst->src bytes per packet
fwd_http_errors	Numeric	Flow total http errors
fwd_failures	Numeric	Flow total failures
bwd_flag_syn	Numeric	Number of SYN flag in flow
bwd_flag_ack	Numeric	Number of ACK flag in flow
bwd_flag_fin	Numeric	Number of FIN flag in flow
bwd_flag_psh	Numeric	Number of PSH flag in flow
bwd_flag_rst	Numeric	Number of RST flag in flow
last_flag	String	Last flag seen in the flow
bwd_avg_bytes	Numeric	Average bytes in flow
bwd_stdev_bytes	Numeric	Std deviation of bytes in flow
bwd_min_bytes	Numeric	Minimum bytes in flow
bwd_max_bytes	Numeric	Maximum bytes in flow
flow_duration	Numeric	Total Duration of flow
Time-based features		
count_dest	Numeric	Number of flows to unique destination IP addresses in the last T seconds from the same source
count_src	Numeric	Number of flows from unique source IP addresses in the last T seconds to the same destination
count_serv_src	Numeric	Number of flows from the source IP to the same destination port in the last T seconds
count_serv_dst	Numeric	Number of flows to the destination IP using same source port in the las T seconds
Connection-based features		
count_dest_conn	Numeric	Number of flows to unique destination IP addresses in the last N flows from the same source
count_src_conn	Numeric	Number of flows from unique source IP addresses in the last N flows to the same destination
count_serv_src_conn	Numeric	Number of flows from the source IP to the same destination port in the last N flows
count_count_serv_src_conn	Numeric	Number of flows to the destination IP using same source port in the last N flows

4.1 K-Means

K-Means algorithm allows to generate clusters of data and identify outliers using a distance based approach. In a first step, K-Means is applied on normal data to train the model and identify k clusters using euclidean distance. Since the objects in the same cluster are similar to each other one can expect that abnormal data will be separated from normal data with a particular distance threshold, hence in step two test data is feeded into the model and outliers data points or clusters of outliers can be identified.

4.1.0.1 Experimental setup

It is usually a good practice to establish the optimal number of clusters for K-Means using a calculated metric in combination with business domain knowledge. A range of possible values were established between 2 and 5 and a silhouette analysis was performed with normal data (training). The figure 15 shows the silhouette score across the range of number of clusters. The silhouette score measures the separation distance between the clusters and how close each point in one cluster is to points in the neighboring clusters. It has a range of [-1, 1] where values

close to 1 indicate the sample is away from neighboring clusters, 0 denotes the sample is close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster [13]. The experiment will use a combination of the score and domain-knowledge to establish the optimal number of cluster for the k-means algorithm.

Let define k=2 as shown in 15

Note: As an alternative, MiniBatchKMeans can be used with similar results using less computational resources.

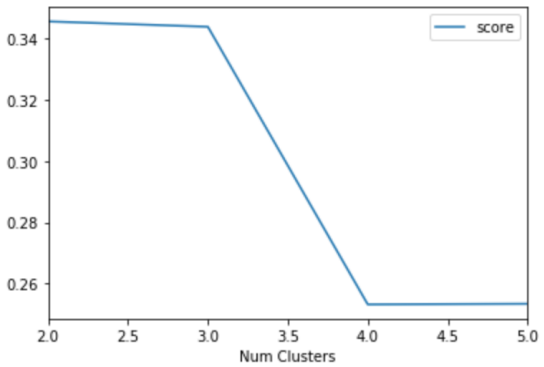


Figure 15: silhouette Score for cluster range 2 - 5.

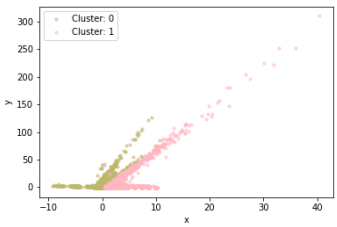
```
1 Exporting BIFLOW_KMEANS...
2 Number of clusters: 2
3
4 Cluster 0: 416708 data points
5
6 Cluster 1: 23415 data points
```

Listing 5: K-Means result

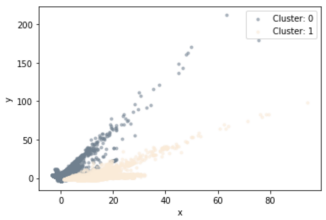
Using dimensionality reduction with Principal Components, it can be seen the a representation of the clusters in figure 16. Although the representation is not significant it is useful to visualize datapoints associated with clusters.

4.1.0.2 Scoring and threshold technique

According to anomaly detection methods, it is necessary to find data points or instances that differ from the rest of the data considering that anomalies only occur very rarely and their features differ in a significant way with respect to the rest the instances. Two methods have been selected for k-means:

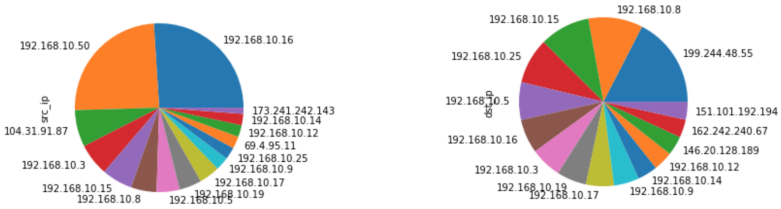


(a) Unidireccional Flow



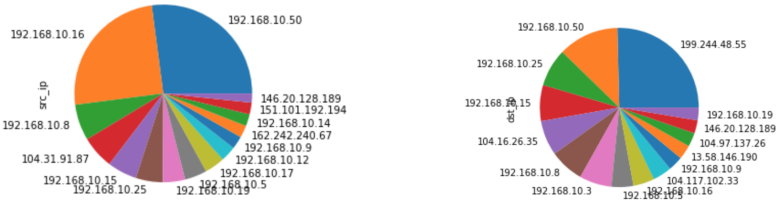
(b) Bidireccional Flow

Figure 16: K-Means using 2 Principal Components



(a) TOP Source IP Address (b) TOP Destination IP Address

Figure 17: K-Means Top Anomalies in Unidirectional Flow with Extreme Value Analysis



(a) TOP Source IP Address (b) TOP Destination IP Address

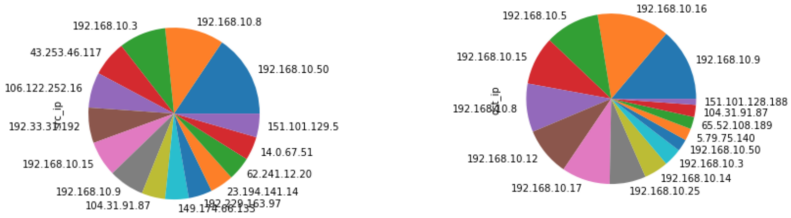
Figure 18: K-Means Top Anomalies in Bidirectional Flow with Extreme Value Analysis

- **Extreme Value Analysis:** It is assumed distances to each of the cluster to be normally distributed. It is calculated the z-score of the distances to each cluster and applied a threshold where score is less than -3 or greater than 3:

  $$anomaly\_threshold = (zscore < -3)\, OR\, (zscore > 3)$$

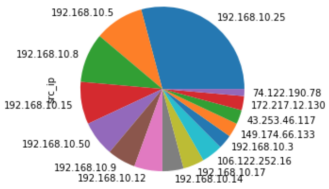
  Using the zscore to filter out outliers it is detected 2,513 for the Bidirectional flow and 2,936 for the UniDirectional flow. If we analyze the top source and destination IP addresses by frequency in the list of 15 top anomalies (IPs) as shown in figure 17 for uni-directional flows and 18 for bidirectional flows, it can be seen the host 192.168.10.50 which was detected as a victim in Assignment 1 .

- **Proximity based:** Analysis is based on that anomalies proximity to its neighbours significantly deviates from the rest. In this analysis, we calculate the most distant data points to the cluster centers. Figures 19 and 20 show the top 15 source and destinations IP addresses for both type of flows.

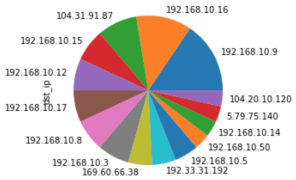


(a) TOP Source IP Address (b) TOP Destination IP Address

Figure 19: K-Means Top Anomalies in Unidirectional Flow with Proximity-based Analysis



(a) TOP Source IP Address



(b) TOP Destination IP Address

Figure 20: K-Means Top Anomalies in Bidirectional Flow with Proximity-based Analysis

4.2 OPTICS (DBSCAN based)

Although source code of the Assignment 2 implementation presents a helper function to run DBSCAN, it was becaused of computational resources limitations that a similar approach was used. OPTICS (Ordering Points To Identify the Clustering Structure) finds core samples of high density and expands clusters from them [14]. OPTICS creates an ordering of the data representing its density-based clustering structure. Clusters are then extracted using a DBSCAN-like method [13].

4.2.0.1 Experimental setup

Two parameters were used to set up OPTICS. Due to time constraints and computational resources these parameters are not tuned but were chosen based on previous analysis.:

- **Eps:** The maximum distance between two samples for one to be considered as in the neighborhood of the other. The average distance from a point to the center of cluster in K-Means is approximately 4 as shown in figure . This value is chosen as a first attempt to identify a valid approach. Eps value will impact the number of outliers identified by the algorithm, if it is too small then a large portion of data will not be labeled inside a cluster and will be considered as outliers. In contrast, if the value is too high then clusters will be merged and datapoints will fall inside the cluster. The threshold for datapoints to be identified as outliers is influenced by this parameter.

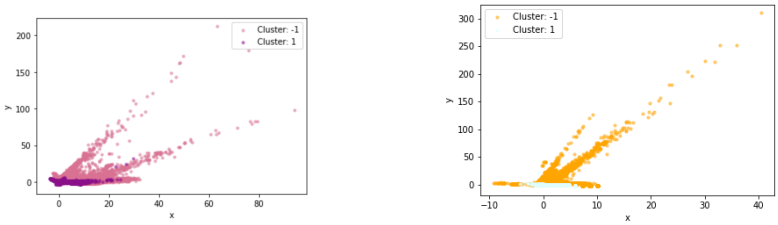
```
[152]: d=kmeans.transform(X_Attack)
       dfd=pd.DataFrame(d,columns=list(set(labels)))
       dfd.describe()
```

[152]:

	0	1
count	440123.000000	440123.000000
mean	4.609646	6.861602
std	4.855068	4.399189
min	2.247108	2.486054
25%	3.194263	5.260209
50%	3.407380	6.270189
75%	4.587872	7.449095
max	648.713210	648.754326

Figure 21: K-Means distances statistics

- **Min Samples:** The number of samples in a neighborhood for a point to be considered as a core point. Some techniques to establish the value consider the number of dimensions in dataset and suggest to define a value greater than the number of dimensions (P): $Min\ Samples > P$.



(a) IForest Bi-directional flow (b) IForest Uni-directional flow

Figure 22: IForest cluster visualization using two Principal Components

4.2.0.2 Scoring and threshold technique

The score is derived from the parameters used to run the clustering algorithm. As described during experimental setup, an Eps close to the mean distance in k-means clustering and a minimum samples greater than the number of parameters influence how datapoints will be classified as outliers by the algorithm: $cluster = -1$.

Note: Although some experiments were conducted with this algorithm, there was no final output with the latest set of features and therefore results are not shown.

4.3 IForest

The IForest algorithm separate observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature [13]. The method tries to isolates anomalies instead of profiles normal data points [15] and therefore it does not use the concept of distance discussed in the previous methods. Using PCA dimensionality reduction technique, it can be seen in figure 22 two clusters one of them identifying possible anomalies according to the parameters chosen. Summary of data points is shown in 6 for bi-directional flow.

```
1 Exporting BIFLOW_IFOREST...
2 Number of clusters: 2
3 Cluster -1: 124248 data points
4 Cluster 1: 315875 data points
```

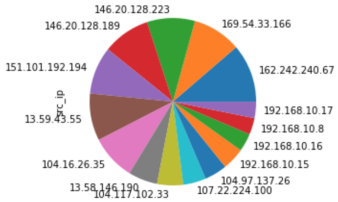
Listing 6: IForest Clusters and Datapoints (Bi-directional flow)

4.3.0.1 Experimental setup

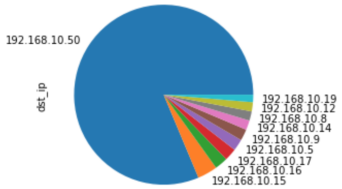
IForest builds an ensemble of iTrees and identify anomalies as the instances which have short average path lengths on the iTrees [16]. For the purpose of this analysis only one parameter is passed to the algorithm, the rest will remain as default by sklearn implementation [13]. The parameter indicates the proportion of outliers and it is used to define a threshold in the decision function. For the purpose of simplify the analysis the parameter is set by default in 1% considering the size of testing dataset.

4.3.0.2 Scoring and threshold technique

The threshold is determined by the contamination parameter which influence the decision function. The score is -1 to instances identified as outliers and +1 otherwise.

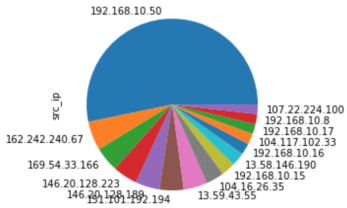


(a) TOP Source IP Address

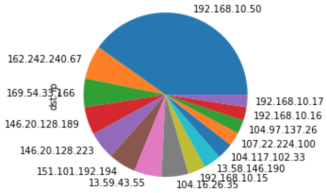


(b) TOP Destination IP Address

Figure 23: IForest Top Anomalies in Bidirectional Flow



(a) TOP Source IP Address



(b) TOP Destination IP Address

Figure 24: IForest Top Anomalies in Unidirectional Flow

Establishing a threshold equal to the top most frequent 15 source IP and destinations IP in the anomalies results, it can be seen in figure 24 the IForest top 15 source IP and destination IPs, it is clearly shown a dominate behaviour from several IP addresses particularly outside the local network and a destination in the host 192.168.1.50 identifying this host as a possible victim of attack. On the other hand, figure 24 shows a similar pattern but it is not clear which IP is a potential attacker or victim due to the nature of the flow. Most of the times, the source IP will play the role of destination IP when the other host reply back.

4.4 LOF (Local Outlier Factor)

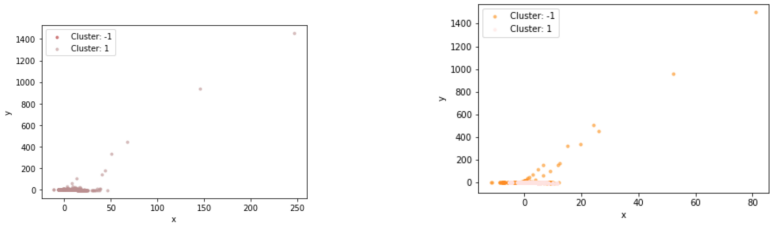
This method pretends to identify deviations on density with respect to data point neighbors. The outliers are detected by comparing the local density of the instances and densities of its neighbors. It is local in that the degree depends on how isolated the object is with respect to the surrounding neighborhood [17]. Using PCA dimensionality reduction technique, it can be seen in figure **??** two clusters one of them identifying possible anomalies according to the parameters chosen. Summary of data points is shown in 7.

```
1 Exporting BIFLOW_LOF...
2 Number of clusters: 2
3
4 Cluster -1: 5939 data points
5 Cluster 1: 586088 data points
```

Listing 7: LOF result

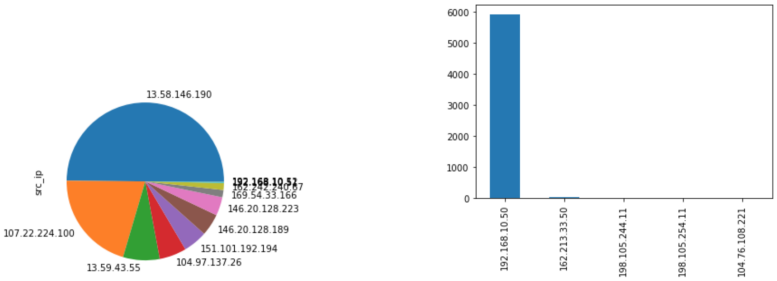
4.4.0.1 Experimental setup

Since other density-based algorithms were used previously, a similar approach to establish the parameters value can be used for LOF. The LOF of an instance is based on the parameter MinPts [17], which is the number of nearest neighbors used to identify the local neighborhood



(a) LOF Bi-directional flow (b) LOF Uni-directional flow

Figure 25: LOF cluster visualization using two Principal Components



(a) TOP Source IP Address (b) TOP Destination IP Address

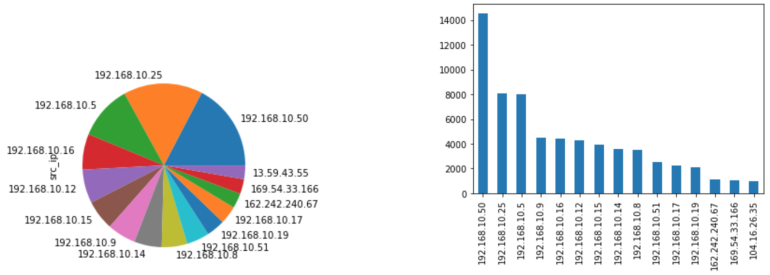
Figure 26: LOF Top Anomalies in Bidirectional Flow

of the instance. It is used a similar approach that the OPTICS algorithm to determine the number of neighbors: $MinPts > P$, where P is the number of dimensions in dataset.

4.4.0.2 Scoring and threshold technique

Outliers are identified by instances with a lower density compared to its neighbors. The value -1 is given to each of the instances classified as outliers and 1 otherwise. Although many heuristics are described in [17] to select the value for parameter MinPts and therefore, influence the threshold used for the algorithm to classify outliers, most of them are focused on selecting reasonable values. It is defined in this report a threshold given by parameter MinPts greater than the number of dimensions (P): $MinPts > P$.

The figure 26 shows the most frequent Top 15 source and destination IP addresses using bi-directional flow. In a similar way, the figure 27 shows the results in the uni-directional case.



(a) TOP Source IP Address (b) TOP Destination IP Address

Figure 27: LOF Top Anomalies in Unidirectional Flow

(a) OCSVM Bi-directional flow (b) OCSVM Uni-directional flow

Figure 28: IForest cluster visualization using two Principal Components

It is clearly shown a top destination IP in the host 192.168.10.50 and several source hosts with labelled flows as anomalous. This clustering technique also shows a potential victim the host 192.168.10.25 and 192.168.10.5.

4.5 One-class Support Vector Machines (OCSVM)

OCSVM defines a decision boundary capable to predict a data point as belong to a class (inside the trained distribution) or outlier (not belong to the distribution) [18]. Previously unsupervised methods have suggested 2-classes, which in the context of this report could be related to malicious activity or normal activity. Using PCA dimensionality reduction technique, it can be seen in figure 28 two clusters one of them identifying possible anomalies according to the parameters chosen for each one of the type of flows. Summary of data points is shown in 8.

```
Exporting BIFLOW_OCSVM...
Number of clusters: 2

Cluster -1: 124598 data points
Cluster 1: 315525 data points

------------------------------
Variability explained by the PC: 0.40361135838123563

Exporting FLOW_OCSVM...
Number of clusters: 2
Cluster -1: 74556 data points
Cluster 1: 782278 data points
```

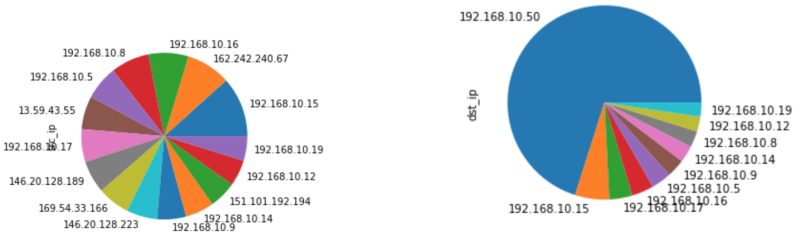
Listing 8: OCSVM result

4.5.0.1 Experimental setup

Since the actual label of attack dataset is unknown it cannot be assumed a balanced class. Therefore it needs to be considered the current testing dataset as imbalanced. The parameter *nu* is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Then we define the value previously used as contamination to establish the parameter. $nu = 0.01$.

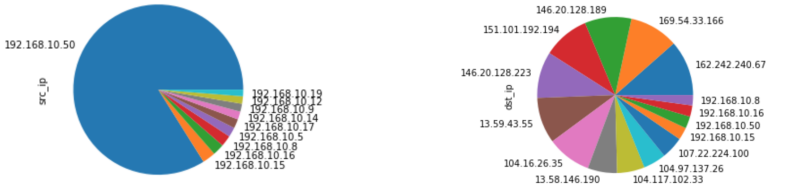
4.5.0.2 Scoring and threshold technique

Outliers are identified by instances classified with the class -1. Each data point in cluster -1 is a potential outlier for the context of this report. The figures 30 and 29 show the top 15 source and destination IP most frequent in uni-directional and bi-directional flows respectively. Here it can be seen that outliers are most frequent having as destination IP the host 192.168.10.50 and several source IP addresses en similar proportions as shown in previous results.



(a) TOP Source IP Address (b) TOP Destination IP Address

Figure 29: OCSVM Top Anomalies in Bidirectional Flow



(a) TOP Source IP Address (b) TOP Destination IP Address

Figure 30: OCSVM Top Anomalies in Unidirectional Flow

5 Scalability in real-time detection

Anomaly detection has been becoming increasingly important for many applications in the security landscape. From computer networks to sensors and IoT devices, the way data is flowing between devices has increase in volume and velocity, characteristics of streaming data which represent a challenge for existing techniques in monitoring inconsistent or irregular data to assure security and stability in the systems [19]. Many researchers have focused the attention to discuss important characteristics of streaming data in the context of anomaly detection [20]. First, packet capturing deals with capturing infinite and high volume data as it the streams are flowing constantly to be analyzed. The approach described in this report is not relevant anymore as it is not possible to dump continuously data to be analyzed offline due to computational and network constraints. The capturing and analysis process makes impractical an approach where offline process interact. Therefore, packet capturing has to be aligned with the anomaly detection process and in order to be scalable it will need to consider continuous packets capturing and processing on high volume data. Secondly, feature generation needs to consider not only packets but flows, although current devices like routers are able to extract flows, the feature extraction process needs to be able to describe the data in a way that models can work with unbalanced data generated in streams. Finally, to deploy machine learning models successfully the time variable has to be treated carefully, an important consideration is the concept drift, where the patterns change compared to the previous set of data used for training and the current model will not be a good fit or consistent. If the model cannot be retrained timely the overall performance will be negatively affected. The anomaly detection models for streaming data are a generalization of the classical model but with the data size growing infinitely [19]. For example, Ding and Fei have proposed a method based on IForest [16] adapted to use in streaming data taking advantage of the lower computational complexity of IForest and that uses the concept drift to retrain the model when necessary.

6 Conclusion

Anomaly detection is a complex task, however, many anomalies can be easily identified executing simple queries and analysing network traffic data. This report provides a good glimpse on how a unsupervised machine learning techniques can lead to intrusion detection using basic clustering algorithms. In general, most of the algorithms gave similar results. However, some of them show more stable outputs in terms of the consistency of outliers. The Isolation Forest, LOF and OCSVM perform reasonable better to identify DDoS attackers and victim whereas distance-based models like K-means shows good results but not as precise. Density-based algorithms like DBSCAN and OPTIC were a good choice in the exploratory phase when sampling can be done, but as data increases the computational resources required are significantly higher at the point that experiments could not finish on time. IForest was chosen to perform the final evaluation which identified a DDoS attack to the victim host 192.168.10.50 and a total of 11 attackers: 162.242.240.67 169.54.33.166 146.20.128.223 146.20.128.189 151.101.192.194 13.59.43.55 104.16.26.35 13.58.146.190 104.117.102.33 107.22.224.100 104.97.137.26

77,711 flows were identified as anomalous, the attack started at 17:14:05.497 and finished on 19:16:12.201 on 2017-07-07. To establish how good this model could be as an Intrusion Detection System (IDS) there has to be an extensive evaluation in terms of 1. Detection accuracy or true positive rate (TPR), 2. False positive rate (FPR), 3. Precision and accuracy, 5. F-measure, and 6. Receiver operating characteristic (ROC) curves. Using the model as input for a classification task. This evaluation is considered future work and out of the scope of this report.

References

[1] H. Alaidaros and M. Mahmuddin, "Flow-Based Approach on Bro Intrusion Detection," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 2-2, pp. 139–145–145, Jun. 2017. [Online]. Available: http://journal.utem.edu.my/index.php/jtec/article/view/2234

[2] A. Papadogiannakis, M. Polychronakis, and E. P. Markatos, "Improving the accuracy of network intrusion detection systems under load using selective packet discarding," in *Proceedings of the Third European Workshop on System Security*, ser. EUROSEC '10. New York, NY, USA: ACM, 2010, pp. 15–21. [Online]. Available: http://doi.acm.org/10.1145/1752046.1752049

[3] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[4] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, Sep. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016740481930118X

[5] E. Boschi and B. Trammell, "Bidirectional Flow Measurement, IPFIX, and Security Analysis," Jan. 2006.

[6] P. Minarik, J. Vykopal, and V. Krmicek, "Improving Host Profiling with Bidirectional Flows," in *2009 International Conference on Computational Science and Engineering*, vol. 3, Aug. 2009, pp. 231–237.

[7] R. R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. New York, NY, USA: ACM, 2004, pp. 187–200. [Online]. Available: http://doi.acm.org/10.1145/1028788.1028812

[8] D. Watson, M. Smart, G. R. Malan, and F. Jahanian, "Protocol scrubbing: network security through transparent flow modification," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 261–273, Apr. 2004.

[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards an Unsupervised Method for Network Anomaly Detection in Large Datasets," *COMPUTING AND INFORMATICS*, vol. 33, no. 1, pp. 1–34–34, Feb. 2014. [Online]. Available: http://www.cai.sk/ojs/index.php/cai/article/view/909

[10] M. Bhuyan, D. K. Bhattacharyya, and J. Kalita, *Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools*, 01 2017.

[11] P. Gogoi, "Network Anomaly Detection using Unsupervised Model," p. 13, 2011.

[12] L. Portnoy, "Intrusion detection with unlabeled data using clustering," Ph.D. dissertation, Columbia University, 2000. [Online]. Available: https://doi.org/10.7916/D8MP5904

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '99. New York, NY, USA: ACM, 1999, pp. 49–60. [Online]. Available: http://doi.acm.org.ezp.lib.unimelb.edu.au/10.1145/304182.304187

[15] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 3:1–3:39, Mar. 2012. [Online]. Available: http://doi.acm.org.ezp.lib.unimelb.edu.au/10.1145/2133360.2133363

[16] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*, Dec. 2008, pp. 413–422.

[17] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-based Local Outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00. New York, NY, USA: ACM, 2000, pp. 93–104, event-place: Dallas, Texas, USA. [Online]. Available: http://doi.acm.org/10.1145/342009.335388

[18] P. Winter, E. Hermann, and M. Zeilinger, "Inductive Intrusion Detection in Flow-Based Network Data Using One-Class Support Vector Machines," in *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, Feb. 2011, pp. 1–5.

[19] Z. Ding and M. Fei, "An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, Jan. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016314999

[20] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast Anomaly Detection for Streaming Data," in *Twenty-Second International Joint Conference on Artificial Intelligence*, Jun. 2011. [Online]. Available: https://www.aaai.org/ocs/index.php/IJCAI/IJCAI11/paper/view/3229