Threat Intelligence Using Machine Learning Packet Dissection

Wiktor Sowinski-Mydlarz
London Metropolitan University
13 PUBLICATIONS **68** CITATIONS

Vassil T Vassilev
London Metropolitan University
60 PUBLICATIONS **238** CITATIONS

Threat Intelligence Using Machine Learning Packet Dissection

Viktor Sowinski-Mydlarz[1], Jun Li[2], Karim Ouazzane[1], Vassil Vassilev[1]
[1] London Metropolitan University Cyber Security Research Centre,
Tower Building, 166-220 Holloway Road, London N7 8DB
[2] Cranfield University Centre for Computational Engineering Sciences,
College Road, Cranfield, MK43 0AL
[1] {w.sowinskimydlarz, k.ouazzane, v.vassilev}@londonmet.ac.uk;
[2] Jun.Li@cranfield.ac.uk

Abstract— In this research we compare different methods to examine network packets using supervised learning to predict possible intrusions. Although there have been many attempts to use Machine Learning for automated packet analysis, our application simplifies the process by taking any packet data source for analysis in a container ready for deploying on a private or public cloud without the need to pre-process the packet data. The packet is dissected extracting numerical data, describing the packet numbers, the time and length of the packets. Categorical variables are the source and destination IP addresses, protocol used and packet info/flag. The use of filters allows ability to recognize any type of packet (e.g., SYN, ACK, FIN, RST). Four machine learning models, i.e., Neural Networks, Support Vector Machines, Logistic Regression and Linear Regression, are applied respectively to calculate the probability of suspicious packets. Subsequently, the outcomes are compared. During the testing against trojan malware, the models can detect the suspicious packets sent to a bogus website and attempts at downloading malware by means of packet payload analysis.

Keywords— Threat Intelligence, Intrusion Detection, Packet Dissection, Machine Learning, Containerization.

1. Introduction

EVERY activity on the Internet involves communication using packets. Every website viewed and every email sent is a series of packets. Each of them carries information such as the sender's IP address, intended destination IP address and is delivered using a specific protocol. Malware can hide its activity at operating system level (rootkits) but it usually leaves a trace on the network activity, whether it is encrypted or not. Inspecting PCAP files (packet captures) for possible intrusions is an everyday activity for security analysts. The contemporary intrusion detection systems (IDS) are unable to address the complexity and the adaptability of cyber threats. Adaptive methods of machine learning (ML) give better detection rates. They also lower false positives and save costs of processing and communicating. In this article we will examine some popular methods for ML from the point of view of their potential use for intrusion detection. Since our main goal is to use them in the cybersecurity framework we are currently developing [1] we are using applications which are containerized using Docker containers. IDS typically use two approaches for detection of an intruder – behavior-based or signature based. Behavior detection applies a profile of normal activity and compares new traffic to the profile. Signature-based detection is easy to apply, a set of signatures must be created though. ML has several applications in cyber security for intrusion detection. It can improve the malware detection by signature. It also provides better analysis of the attack vectors and intrusions. Another benefit of its use is the automation of daily activities performed by security analysts, which saves time and effort for more important work.

There are two general types of algorithms for learning from data: Unsupervised and Supervised ML. While unsupervised algorithms are used mainly for detection and classification, supervised algorithms can also apply experience from past to predict forthcoming events and trends [2]. In our research we are comparing four methods:

- Regression is used to extrapolate the trends using examples of existing data [2].
- Logistic Regression is applied to shape the probability of existing class or event. It has proven to be successful for packet classification since it recognizes the types of packets [3].
- Artificial Neural Networks (NN), which are inspired by biological neural networks in animal and human brains can identify hidden patterns and correlations. They have been particularly important mechanism for deep learning with more complex models of recognition and classification [3,4].
- Support Vector Machine (SVM) is a supervised learning model with additional processes for analysis of classification and regression data [5,6].

The organization of the paper is as follows. Section 2 reviews the existing work in the field. In Section 3 we state the problem. We introduce the data analyzed by our models and examine model's structure and implementation in section 4. The results of the analysis and the evaluation of the models are given in Section 5. In Section 6 we describe the incremental learning method and the containerization of the application. Section 7 concludes the paper and gives a brief description of the future work.

2. Related work

In [6] SVMs and NNs are compared from the point of view of their potential for enhancing threat intelligence and for data preparation. The research proves that SVMs surpass neural networks in training time and detection accuracy. The reduction of generalization errors is another benefit of SVMs. Autoclass is a classifier that can learn clusters from training data by Naïve Bayes algorithm with attributes of uncategorized instances. Autoclass method is used in [7]. In [8] a supervised Naïve Bayesian method is used to classify packets with manual input for the estimator. An advantage of the method is the high accuracy thanks to adapting the algorithm to the set of features. Some disadvantages are the processing cost and the speed of the algorithm.

A method for recognizing the application relying on examining only the first five TCP packets is proposed in [9]. It analyses the packets only at the negotiating stage, omitting the control packets. An SVM method called Enhanced SVM lowering the false positives rate in unsupervised learning is presented in [10]. In [11] an approach to deal with the shortcomings of the current hand-tuned heuristics packet classification is proposed, but it uses deep learning to build the decision tree and rule-based packet classifier. An AI-SIEM system able to disseminate true alerts and false alerts based on deep learning techniques is proposed by [12].

Several works compare different methods and/or provide a summary of a method. [13] introduces more systematic approach for assessing the performance of classification algorithms. It compares Multilayer Perceptrons (MLP), Decision Trees and Bayesian Networks.

Multilayer Perceptron (MLP) is a type of feed forward neural network. It consists of three types of layers - the input layer, output layer and hidden layer which are fully connected. The inputs are pushed forward by using the dot product of the input with the weights between the input layer and the hidden layer. We are using an MLP in this research. Please see Fig. 1 in section "ML Models Implementation". Disadvantage of MLP is that the number of total parameters can grow high (exponentially) for each layer. It also disregards spatial information (taking flattened vectors as inputs).

Decision Tree transforms the data into a tree representation. Each internal node represents an attribute, and each leaf node represents a class label. Decision Tree algorithms can solve both regression and classification problems. The advantages of this method are less effort for data preparation needed, outputs easy to read and interpret, it can be used without scaling the data. The downsides are more memory and time required, and high sensitivity to small changes (causing large changes in the tree structure).

Bayesian Networks provide a robust and mathematically coherent framework for problem analysis. They are probabilistic graphical model comprised of nodes and directed edges and can combine different sources of weak evidence producing strong hypotheses. Different types of reasoning, diagnostic, predictive and intercausal can be performed. Bayesian Networks are easier for humans to understand than Neural Networks. The drawbacks are inability to depict variables which are correlated (undirected graphical models, i.e., Markov Random Field are more suitable for that task) and high resources requirements. [14] relates them to the Naïve Bayes estimator approach.

3. Problem description

The first step for the hacker is to analyse and inspect the potential target. This is done by packet sniffing, email, malware, and social engineering. Next, they choose the best method to invade the network. It means that they encode and prepare the most efficient tools for exploiting the vulnerability. They break security and plant the malware. As soon as the system is breached, they continue to exploit the sensitive data for intended benefits. In our approach the detection, recognition, and classification of threats are all done through packet dissection. Here are some of the types of packets that are interesting from threat intelligence perspective:

- TCP RST packet is sent when remote side signifies not recognized connection on which the previous TCP packet was sent. The reason could be the port not being open or connection being closed. It forces a reset on the connection. The host does not wait for response and instantly terminates the connection. It is more aggressive way to stop the connection.
- TCP FIN is sent when there is a need for acknowledgement as the connection is about to close. The host will not accept any more packets.
- TCP SYN as the name suggests signifies exchange of synchronization packets by the two communicating sides. It is in fact the first packet sent from each side in the beginning of the connection.

If malware on a computer can spread through the LAN it will certainly try to initiate connections with other computers. Hence, we can expect other machines to receive SYN flags from the first computer. In the case when these are end user systems, it signifies an anomaly. Otherwise, first computer would try not to connect to other computers (if there is explicit business need). If we can check the number of SYN packets on the second computer, we could validate these packets for each IP versus a threshold. We can also scan the open ports on the first computer which are connected to the corresponding ports on the second computer. when there is a surge in SYN packets. If first machine receives a lot of RST packets, the target computer most likely denied initiating connections. We can deduct from the computer receiving too many RST flags, that likely the first machine is attempting to scan the neighboring system with SYN packets. Numerous SYN packets received indicate that the source is affected, and a lot of RST packets received means that recipient is infected.

Classification learning approach identifies a set of categories where new example fits, based on data training set with examples which membership is established. Signature based approach determines a unique identifier of a known threat allowing it to be recognized after training. Signatures are used for threat detection. Because of the variety of insider threats most works do not attempt to explicitly model the threat behaviour. Alternatively, Deep Neural Networks (DNNs) and

Recurrent Neural Networks (RNNs) which are trained to identify characteristics of the users can be used for learning behaviour models. Such a research is not concerned with packet dissection per se, although it uses ML.

In our research we are leaving out the security policies and active protection to other parts of the security framework, looking only for fast and reliable methods for intrusion detection. Because of this we are focused on automated packet dissection to reduce any analyst involvement in the security analytics.

4. Methodology

To address the problem stated above we are using 4 different ML methods (Neural Network, Support Vector Machine, Logistic Regression and Linear Regression). In the following section we introduce the dataset used for experimentation and then describe implementation of 4 ML models focusing on Multi-Layer Perceptron Neural Network.

4.1 Data and exploratory analysis

The data for our research come from Netresec (public packet capture repository, https://www.netresec.com/? page=PcapFiles) [15]. The format of the files is PCAP and CSV and their size varies from 6MB to 318MB. We used 7 CSV files and 10 PCAP files which include Ursnif and Trickbot infected traffic. Each record has numerical data, describing the packet numbers, the time and length of the packets. Categorical variables are the source and destination IP addresses, protocol used and packet info. The distribution of the packets across different protocols in the data file is as follows:

TABLE I

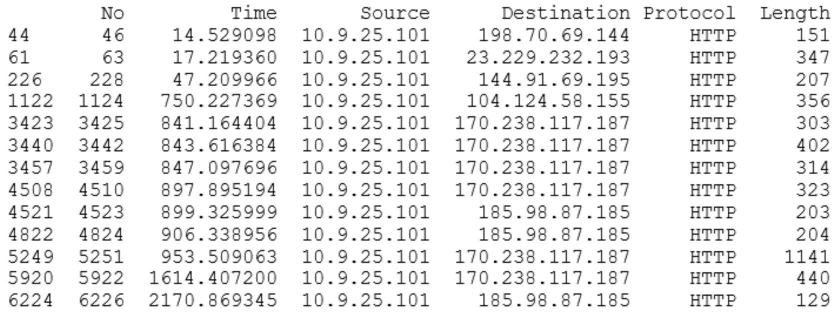
DISTRIBUTION OF PACKETS

TCP	14565	HTTP	26
TLSv1	1346	BROWSER	21
NBNS	371	SSLv3	15
DNS	201	NTP	2
LLMNR	65	SSDP	1
IGMPv3	61	MDNS	1
TLSv1.2	54	DHCP	1

The number of unique values for IP source addresses was 47 distinct IPs in an average PCAP file. The number of unique values for IP destination addresses was 51 distinct IPs in a sample PCAP file. If we apply filters to the dataset, selecting protocol as HTTP and IP address as 10.9.25.101, for example, we get the following:

TABLE II

PACKETS FROM PROTOCOL HTTP AND IP ADDRESS 10.9.25.101



```
        No         Time       Source     Destination Protocol  Length
44      46    14.529098  10.9.25.101    198.70.69.144     HTTP     151
61      63    17.219360  10.9.25.101   23.229.232.193     HTTP     347
226    228    47.209966  10.9.25.101    144.91.69.195     HTTP     207
1122  1124   750.227369  10.9.25.101   104.124.58.155     HTTP     356
3423  3425   841.164404  10.9.25.101  170.238.117.187     HTTP     303
3440  3442   843.616384  10.9.25.101  170.238.117.187     HTTP     402
3457  3459   847.097696  10.9.25.101  170.238.117.187     HTTP     314
4508  4510   897.895194  10.9.25.101  170.238.117.187     HTTP     323
4521  4523   899.325999  10.9.25.101    185.98.87.185     HTTP     203
4822  4824   906.338956  10.9.25.101    185.98.87.185     HTTP     204
5249  5251   953.509063  10.9.25.101  170.238.117.187     HTTP    1141
5920  5922  1614.407200  10.9.25.101  170.238.117.187     HTTP     440
6224  6226  2170.869345  10.9.25.101    185.98.87.185     HTTP     129
```

The data are transformed into a tabular style of labelled axes to be used in Pandas data frame, which utilizes a two-dimensional, size-mutable, potentially heterogeneous tabular data format.

4.2 ML Models Implementation

Logistic Regression and Linear Regression use functional dependence between the variables. Despite the similarities the two methods have quite distinctive use. In linear regression the dependent variable is continuous (e.g., height, weight, time), while in logistic regression it is categorical (e.g., dog/cat/horse, dead/alive, types of packets). In our research the basic output is four types of packets - RST, SYN, FIN and general. In a different configuration the four types can be interpreted as suspicious, malware download (two types), and general.

The other two methods - NN and SVM - use supervised ML. NN is parametric, using hyper parameters tuning in the training, whilst SVM is nonparametric, using linear vector for separating the classes. The main benefits of using SVM are - they are less likely to be trapped in local minima and can easily see “the bigger picture”, they cannot be overfitted with small number of samples or long training. A selection of kernels gives flexibility and adaptability to various types of problems. SVMs can handle nonlinear cases with the help of RBF (Radial Basis Function) kernel, in fact with a specific kernel they can tackle any problem. They handle non-prepared, non-labelled and unstructured data particularly well, scaling to deal with more complex data. Multi-Layer Perceptron NN is sequential with three layers, where each layer has precisely one input and one output tensor. The output of one layer is the input of the next layer similarly to neuron connections in the organic brain. Our NN model has a signal receiving input layer, a hidden layer making the calculations and a predicting output layer as shown in Fig. 1.

We will go into more detail regarding MLP NN now, discussing the model architecture:

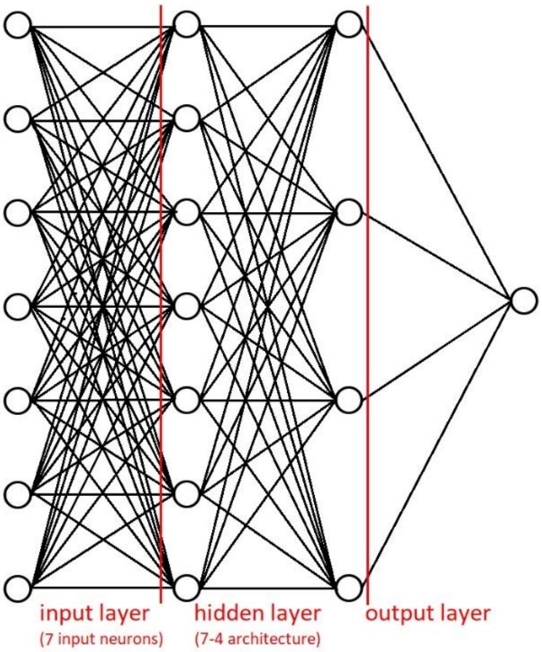


Fig. 1 NN Model Layers

For activation function we are using ReLU (Rectified Linear Unit), the most popular function in convolutional NNs and deep learning. Both the function and its derivative are monotonic. As we are performing a classification, for output layer we are using Softmax activation function because the more often used sigmoid function can handle only two classes. Softmax, on the other hand, calculates distribution of multiclass probability over target classes.

We specify the input shape in the first layer using argument input_dim=5. The layer is dense, it implements operation output = activation (dot (input, kernel) + bias). The kernel is the layer's weight matrix and bias is not applicable in the case of this work (a vector created by the layer, see [16]). The parameters of the input layer are output dimension and number of hidden units. The 7 input nodes take the data fed into the NN and pass it to the hidden nodes.

The input of the hidden layer is also an array of size 7 and the layer has 4 outputs. The output layer uses Softmax as activation function and has an input dimension of 4, and 1 output with 4 results (identified packets). The model is configured with stochastic gradient descent (SGD) as an optimizer, which updates parameters for each training example and label. Adam is another popular optimizer, which is faster than SGD, but the latter has a better convergence with longer training time and momentum, which helps accelerating gradient vectors.

We use sparse categorical cross entropy as the loss function to optimize the parameter values. It calculates cross entropy loss between labels and predictions. Categorical means having more than two classes (as opposed to binary) and sparse implies using an integer ranging from 0 to quantity of classes minus 1. True labels format is the only distinction between sparse categorical cross entropy and categorical cross entropy. In multiclass classification problem like here each data entry belongs to a single class. [17]

The implementation relies on four Python libraries for ML. TensorFlow is an end-to-end open-source system which includes a thorough, adaptable environment of tools, libraries, and community resources. Keras is an open-source NN library running on top of TensorFlow, intended for rapid experimentation in deep NNs. For reading in PCAP files into Pandas data frame we use a network protocol analyzer called TShark. SKLearn package is an open-source library featuring classification, regression, and clustering algorithms. The goal is to identify an attack vector before the intrusion happens. The models can not only detect healthy and unhealthy packets in the network traffic but possibly also search for other patterns like remote access session duration, backdoors listening ports, unauthorized communication channels, exploit kits. Based on the data given, it has a potential to detect uploads, tunnelling, and injections. DOS attacks can be discovered by abnormal increase in NON 2xx/3xx codes, namely 5xx errors. 404 errors signifying directory brute-forcing, and 401 errors for bypassing the authorization.

Malicious websites and encrypted communication channels can be found by recognizing the SSL certificates server's names. All of these can be detected by means of packet/frame dissection. More specific algorithms of this kind are a work in progress, which depends only on the type of data provided for analysis. Next part of this paper is concerned with the performance of the 4 models.

5. Modelling results and evaluation

In this section we are going to assess the precision and speed of a NN, SVM, Linear Regression and Logistic Regression using sample packet capture files. We are also going to discuss Class Weighting as a way of handling imbalanced datasets in Deep Learning. This part of the paper also includes a description of hyperparameters tuning experiments and evaluation of impact of input variables number on predictions.

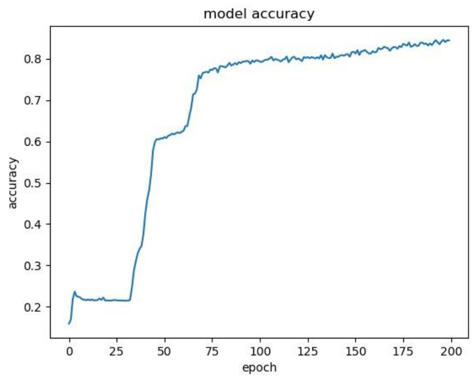
5.1 Performance of different models. (PCAP 1)

The first model is a NN implemented in Tensorflow/Keras. For a PCAP file of size 6,221,590 bytes the accuracy on training data is 88% and the accuracy on test data is 88% as well.

Number of total regular packets: 15608
Number of total ACK packets: 4774
Number of total SYN packets: 511

Number of predicted regular packets: 4163
Number of predicted ACK packets: 1936
Number of predicted SYN packets: 169

Number of regular packets in test set: 4722
Number of ACK packets in test set: 1387
Number of SYN packets in test set: 159



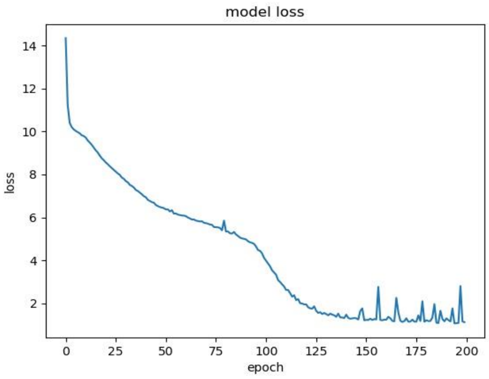
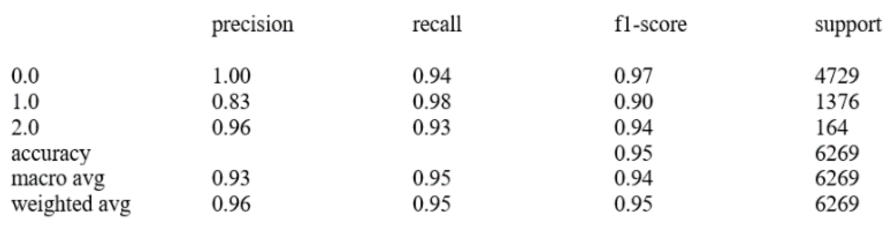


Fig. 2 NN Model Accuracy Over Epochs, Fig. 3 NN Model Loss Over Epoch

As we can see, the prediction for ACK packets is off by 549. The time to train the model is 02 minutes 16 seconds for 200 epochs. Adding some Gaussian Noise as a layer with same input and output shapes, between the input and hidden layer, reduces accuracy only as far as the model is not overfitted. The standard noise deviation is set to 0.1 and mean is zero.

An alternative to NN is SVM. Here is a report on the performance of this solution:

TABLE III
SVM CLASSIFICATION REPORT (Y_TEST, Y_PRED)



	precision	recall	f1-score	support
0.0	1.00	0.94	0.97	4729
1.0	0.83	0.98	0.90	1376
2.0	0.96	0.93	0.94	164
accuracy			0.95	6269
macro avg	0.93	0.95	0.94	6269
weighted avg	0.96	0.95	0.95	6269

In the above table the criteria are as follows:

Precision is classifier's capacity to not categorize an instance positive whilst it is in fact negative. It is a ratio of true positives to sum of true and false positives as a percentage of correct positives.

Recall is classifier's capability to identify positives. It is a ratio of true positives to sum of true positives and false negatives as a percentage of correctly classified true positives.

F1- score has value between 0 (worst) and 1 (best). It is used for classifier models comparison as a weighted harmonic mean of precision and recall.

Support is the sum of actual instances of class. Imbalanced values in training set suggest flaw in the classifier's scores and can be resolved by class weighting and oversampling. Our data set is class weighted.

Here is a summary of packets detected by SVM:

```
Number of predicted regular packets: 4470
Number of predicted ACK packets: 1641
Number of predicted SYN packets: 158
Number of predicted FIN packets: 0

Number of regular packets in test set: 4729
Number of ACK packets in test set: 1376
Number of SYN packets in test set: 164
Number of predicted FIN packets: 0
```

As we can see, SVM delivers prediction accuracy better than NN. In fact, its accuracy is at 92% and the model executes in 28 seconds. This is a major increase in performance and efficiency over the NN. This model is also very resilient to noise tolerance, since introducing noise to dataset does not seem to have any effect on accuracy.

The third model is Logistic Regression. Originally, it was built with PySpark Python API. However, since there were numerous problems with the implementation due to data leakages, we changed it to Sklearn. This Python module integrates typical ML algorithms with scientific Python packages (numpy, scipy, matplotlib). Here are the results for Logistic Regression:

```
Number of predicted regular packets: 4020
Number of predicted ACK packets: 1739
Number of predicted SYN packets: 510
Number of predicted FIN packets: 0

Number of regular packets in test set: 4699
Number of ACK packets in test set: 1413
Number of SYN packets in test set: 157
Number of FIN packets in test set: 0
```

Although this model is extremely fast (it executes in 5 seconds), the results are not impressive as the accuracy is 78%. Introducing noise does not have much effect either.

Finally, we can evaluate Linear regression. The execution time is only a few seconds, but the predictions are inaccurate. Probability estimations are only for classification and not for regression (i.e., numeric prediction).

5.2 Class Weighting

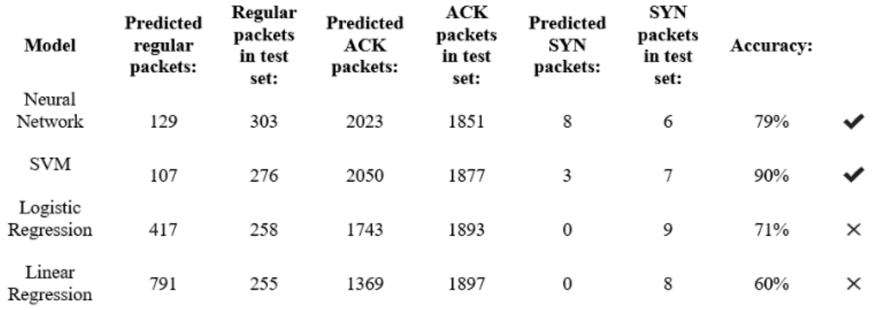
The data in our experiments is imbalanced as we have only a small number of RST, SYN and FIN packets in the dataset. We need the classifier to weight the available samples of these packets more than regular ones. To do that, we pass weights to underrepresented classes allowing the model to recognize and predict them more easily. In this case the percentage of regular packets is 74.7% and percentage of ACK packets is 22.8% of total. The weight of ACK packets class is calculated as percentage of regular divided by percentage of ACK and is 32.7. SYN packets are only 2.45% of all packets. Their weight is percentage of regular packets divided by percentage of SYN. The ratio for SYN is 305.4. These values are automatically calculated based on numbers of packets in the dataset and entered into a dictionary which is then fed into the model during fitting by class_weights parameter.

5.3 Impact of Input Variables Number on Prediction

Some of the inputs may not make a significant contribution to the prediction. In this section we compare the results with reduced inputs for each model. The following table gives the baseline (with all inputs):

TABLE IV

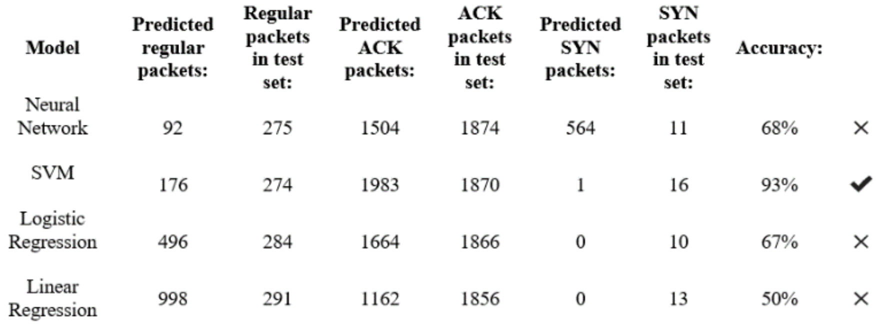
PACKET STATISTICS (BASELINE)



Model	Predicted regular packets:	Regular packets in test set:	Predicted ACK packets:	ACK packets in test set:	Predicted SYN packets:	SYN packets in test set:	Accuracy:	
Neural Network	129	303	2023	1851	8	6	79%	✔
SVM	107	276	2050	1877	3	7	90%	✔
Logistic Regression	417	258	1743	1893	0	9	71%	×
Linear Regression	791	255	1369	1897	0	8	60%	×

We start by removing source and destination IP addresses from the input set. That leaves 'No', 'Protocol' and 'Length' as predictors. These are results from that reduced dataset:

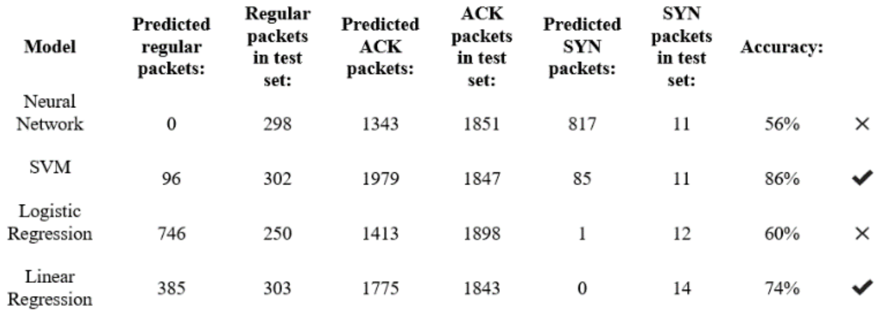
TABLE V
PACKET STATISTICS (REDUCED INPUT 1)



Model	Predicted regular packets:	Regular packets in test set:	Predicted ACK packets:	ACK packets in test set:	Predicted SYN packets:	SYN packets in test set:	Accuracy:	
Neural Network	92	275	1504	1874	564	11	68%	✕
SVM	176	274	1983	1870	1	16	93%	✓
Logistic Regression	496	284	1664	1866	0	10	67%	✕
Linear Regression	998	291	1162	1856	0	13	50%	✕

Next, we will remove 'Protocol' and 'Length' inputs, leaving 'No', 'Source' and 'Destination' IP.

TABLE VI
PACKET STATISTICS (REDUCED INPUT 2)



Model	Predicted regular packets:	Regular packets in test set:	Predicted ACK packets:	ACK packets in test set:	Predicted SYN packets:	SYN packets in test set:	Accuracy:	
Neural Network	0	298	1343	1851	817	11	56%	✕
SVM	96	302	1979	1847	85	11	86%	✓
Logistic Regression	746	250	1413	1898	1	12	60%	✕
Linear Regression	385	303	1775	1843	0	14	74%	✓

This shows that the protocol and the length of packet are crucial for predictions. Source and destination IP addresses do not seem to have much influence on the accuracy, as can be suspected from the type of data they represent, namely ordinal categorical variables. Numerical variables are packet numbers (discrete), time and length (continuous). The length of the packet has a considerate impact on predictions as well as the protocol used (nominal categorical data). Interestingly, Linear Regression algorithm seems to perform better with just ordinal variables. We need to emphasize that the accuracy of the model is based on the strength of relationship between independent variables and dependent variables (type of packet in this case), since the independent variable affects the dependent variable**.**

5.4 Hyperparameters Tuning. (PCAP 2)

5.4.1 Neural Network Parameters

The method we are using for finding optimal parameters is Hand Tuning (Trial and Error). Let us begin with the **number of epochs** in training the NN. For 50 epochs and batch size set to 16 the test set accuracy is only 30% so we need to increase the number of epochs to 100. This results in 42% accuracy. Increasing epochs to 250 gives 89% accuracy. Increasing **batch size** to 128 speeds the training. The optimization algorithm we used is **Adam** (adaptive moment estimation). It is a substitute for stochastic gradient descent for training deep learning models. **LazyAdam** is an upgraded version of Adam designed to be more efficient at handling sparse updates. However, using LazyAdam optimization does not affect the results. **Stochastic Gradient Descent** (SGD) is a simple optimization algorithm using a fixed learning rate for parameters in the training phase. SGD is not suitable for this research either, as it results in drastic accuracy drop. **AdaGrad** algorithm gives similar results with a main difference in using adaptive gradients. With AdaGrad accuracy drops to 48%. Another alternative for Adam optimizer is **RmsProp** which uses the magnitude to normalize gradients. This helps with balancing the momentum, reducing it for large gradients and expanding for small gradients and the accuracy increases to 75%. As we can see Adam optimizer works best for our purposes. There are 4 arguments of the algorithm:

- learning-rate, beta1, beta2 and epsilon. The optimal value for learning-rate is 0.01. Large values (0.1) speed up the learning rate while small values (1.0E-5) slow it down.
- beta1 is the initial exponential decay, usually 0.9.
- beta2 is the subsequent exponential decay, normally 0.999.
- epsilon is a threshold to prevent division by zero (typically set to 1e-08). With epsilon set to 1 we have 86% accuracy.

Originally, we used **ReLU** as the activation function for input and hidden layers. Experimenting with new **Swish** function does not improve accuracy. Swish is a smooth, non-monotonic function matching and outperforming ReLU in deep networks, developed by Google. It is unbounded above and bounded below. Both **sigmoid** and **Tanh** functions yield worse results.

The number of neurons for input layer and hidden layer was initially 7. During the experiments we have increased them to 12 and added a second hidden layer. This configuration gives 88% accuracy.

5.4.2 SVM Parameters

A **linear** kernel is not suitable for our nonlinear problem. RBF (**Radial Basis Function**) kernel gives us an accuracy of 94%. **Polynomial** and **sigmoid** kernels have result at around 35% and 20% respectively. In RBF the class boundaries dissolve when they get away from support vectors. **Gamma parameter** defines the speed of this dissolving. Large number decreases the support vector influence. With gamma set to 10 we have 74% accuracy, setting it to 1000 gives 86% precision, the best value is 30000 with **94% correctness**. **C parameter** is concerned with the importance of misclassified points - higher value mean focusing more on correct classification. This parameter does not have any influence on the accuracy of our SVM.

5.4.3 Logistic and Linear Regression Parameters

We have tried different methods of oversampling - SMOTE, ADASYN and Near-Miss algorithm. With SMOTE the accuracy of Logistic Regression and Linear Regression is at 70%. The minority class is oversampled by synthetically generating additional samples. This makes a major difference in evaluation of methods. We also adopt the MinMaxScaler on the dataset, constraining the range of values. There is also a parameter C. which when set to 1 gives an increase in accuracy to 80%. With lower value of C, the regularization strength is improved (underfitting data gives simpler model). With higher value of C the regularization is lower (overfitting data leads to more complex model). Further increase in C value does not affect the performance.

6. Containerization and Incremental learning

For containerization we used **Docker**. Our application uses 5 containers: **Airflow** to create model and process tasks, **PostgreSQL** for storing metadata, **Kafka** for data streaming, **Zookeeper** for managing sessions and topics and **MLflow** for showing model statistics, comparison, and support of incremental learning. **DAGs** (Directed Acyclic Graphs) are managed by AirFlow and contain all tasks to be executed in the form of a workflow, accounting their relationships and dependencies. The containers are orchestrated through Docker Compose using two commands: build and up. The build command executes a YAML file which specifies the workflow, while up executes the constructed workflow. In our experiments on a laptop, it takes 16 minutes and 27 minutes for the execution of the two commands, respectively. Once the Docker commands are finished, we can use the Airflow interface within the browser to monitor and control the workflow. Airflow allows to schedule and manage tasks such as training initial model, streaming data through Kafka's topic and updating the model [20]. Firstly, we pre-process data, create and fit the initial ML model using TensorFlow/Keras. Then we simulate streaming data from the sensors and feed that to Kafka. Every 5 minutes we extract this data for updating the ML model (NN), evaluate its performance against other models and choose the best one. If this model outperforms older models it is set as a current model and saved. Finally, MLflow shows the statistics of the models. As the initial model is used for preliminary training, it executes only once. Streaming and updating happens over 5 minutes intervals. MLflow platform shows the number of samples used, the loss and accuracy of the current and the updated model, the number of epochs ran by each and the batch size used. It also gives the duration of the model and status of either replacing the current one or being archived. After running about 20 updates through Directed Acyclic Graphs (DAGs), the accuracy of the NN model **increased to 88%** thanks to incremental learning and class weighting. Fig. 4 shows the graphical output of the AirFlow dashboard which controls the execution.

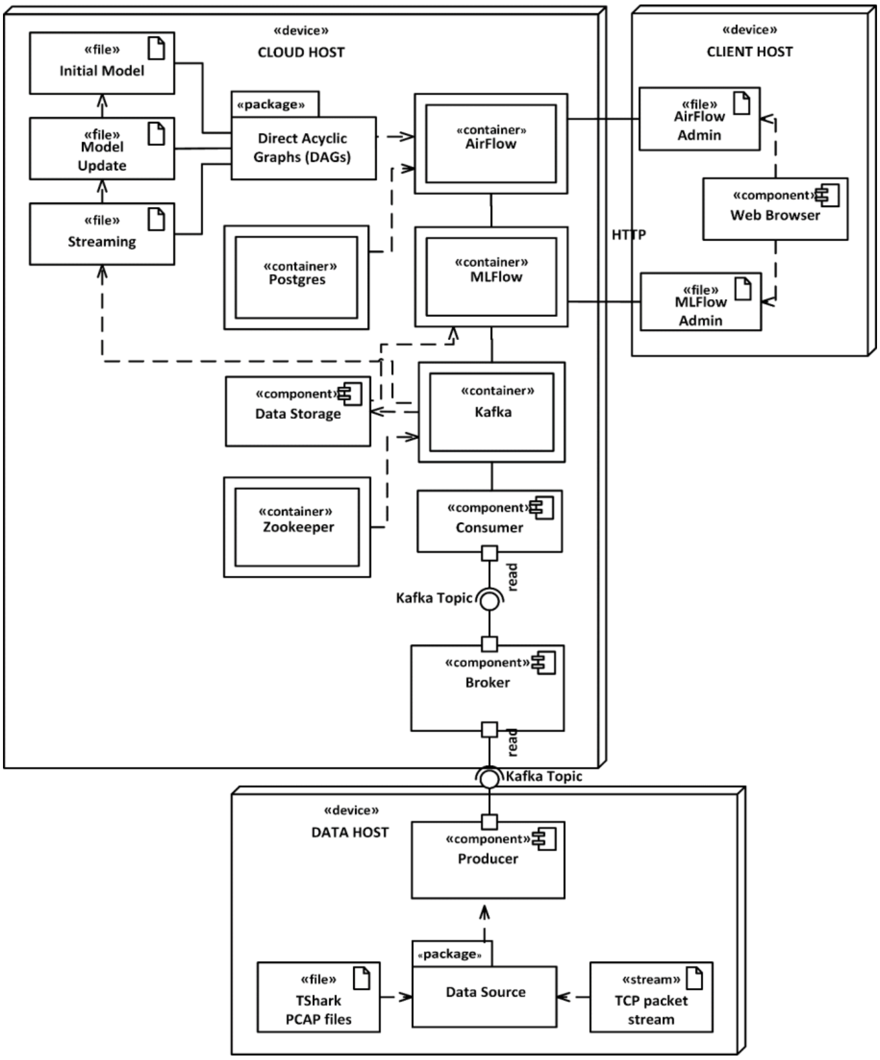


Fig. 4 Containerization of the Analytics Engines

Incremental learning is a methodology for ML which relies on new examples as they appear and model adaptation. It is particularly suitable for real-time industrial applications. In our research we have implemented it by updating and replacing the model automatically, based on evaluation of its performance for analyzing streamlined data. This has been done by a strict discipline of control of the parameters and full containerization of the software. The main advantage is its proficiency in using the memory, CPU, and storage and it serves as a preparation for deploying the analytics to the cloud.

7. Conclusion and Future Work

In this research we are using supervised learning based on classified data since unsupervised learning works only with unstructured data without labelling to ascertain patterns and although it is not constrained by human understanding it usually yields worse results. Although our method has been tested only on packet captures, it has the potential for examining live data traffic. The training and testing sets are currently extracted from the same PCAP files, but thanks to the use of TShark network packet analyser we are also able to capture data from a live network. We experimented with four different methods which have different power for prediction and analysis of new data. Multiple linear regression has numerous explanatory variables (manipulated by the researcher, representing inputs) associated with scalar responses (dependent variables representing the output). Logistic Regression can be employed to structure the probability of suspicious packets in the data stream. SVM algorithm has a capacity to uncover intricate nonlinear relations between dependent and independent variables helping with anomaly detection in data. Incremental learning using NN has a big potential as there was a significant increase in performance after just 20 runs of the model update DAG (up to 88%). Class weighting also contributed to this increase in accuracy.

There are several filters worth investigating: session statistics (amount of downloaded and uploaded data), session duration, HTTP traffic parameters, TCP ports, etc. They could be used to identify unauthorized access, suspicious traffic, brute force, and DOS attacks. We can additionally analyze SSH, SSL, FTP and DNS data to detect suspicious domains and unauthorized servers.

We currently stream the data through Kafka, but we are considering also lighter protocols, such as MQTT and other streaming tools, such as NiFi which might be more suitable for different types of live data. Our plans are finally to deploy the containerized analytics engines to the Kubernetes, so that we can analyze live data stream directly on the private cloud as a security service.

Acknowledgment

This research has been funded by Lloyds Banking Group, but the opinions and conclusions are entirely of the authors and do not reflect the official policy of the bank.

References

[1] V. Vassilev, V. Sowinski-Mydlarz, P. Gasiorowski et al. (2020), Intelligence Graphs for Threat Intelligence and Security Policy Validation of Cyber Systems, In: P. Bansal, M. Tushir, V. Balas and R. Srivastava (eds.), Advances in Intelligent Systems and Computing, Vol. 1164, Springer, pp. 125-140; ISSN 9789811549915

[2] Boutaba, R., Salahuddin, M.A., Limam, N. et al. (2018). "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities." J. of Internet Services and Applications, Springer. [online at: https://jisajournal.springeropen.com/articles/10.1186/s13174-018-0087-2 [Accessed 07 Jun. 2020].

[3] Ray, Sunil (2015). "7 Regression Techniques you should know!". Analytics Vidhya. [online at: https://www.analyticsvidhya.com/; [Accessed 07 Jun. 2020].

[4] Seif, George (2020). "Selecting the best Machine Learning algorithm for your regression problem". [online at: https://towardsdatascience. com/selecting-the-best-machine-learning-algorithm-for-your-regression-problem-20c330bad4ef; Accessed 07 Jun. 2020].

[5] Asiri, Sidath (2018) "Machine Learning Classifiers". Towards Data Science [online] Available at: https://towardsdatascience.com/ [Accessed 08 Jun. 2020].

[6] Mukkamala, S., Sung, A. H. (2002) "Intrusion Detection: Support Vector Machines and Neural Networks", IEEE Xplore. DOI: 10.1109/IJCNN.2002.1007774[online at: https://www.researchgate.net/profile/Andrew_Sung/publication/3950039_Intrusion_detection_using_neural_networks_and_support_vector_machines.pdf; Accessed 16 Jun. 2020].

[7] Zander, S., Nguyen, T., Armitage, G. (2005) "Automated Traffic Classification and Application Identification using Machine Learning", Proc. IEEE Conf. on Local Computer Networks (LCN'05) [online at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.474.6875&rep=rep1&type=pdf; Accessed 16 Jun. 2020].

[8] Moore, A. W., Zuev, D. (2005) "Internet traffic classification using bayesian analysis techniques". Proc, ACM SIGMETRICS, Vol. 33, pp, 50–60 [online] Available at: https://www.cl.cam.ac.uk/~awm22 /publications/ moore2005internet.pdf [Accessed 16 Jun. 2020].

[9] Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A. (2006) "Traffic classification on the fly". ACM SIGCOMM, Vol. 36(2), pp. 23-26 [online at: https://www.researchgate.net/publication/220195236_ Traffic_classification_on_the_fly; Accessed 16 Jun. 2020].

[10] Shon, T., Moon, J. (2007) "A hybrid machine learning approach to network anomaly detection", Information Sciences, Vol, 177, Issue 18, pp. 3799-3821. [online at: https://www.sciencedirect. com/science/article/pii/S002002550700164; Accessed 09 Jun. 2020].

[11] Tuor, A., Kaplan, S., Hutchinson, B. et al. (2017) "Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams". Western Washington Univ. Pacific Northwest National Laboratory. [online] Available at: https://www.aaai.org/AAAIW17/paper/download/15126/14668 [Accessed 09 Jun. 2020].

[12] Liang, E., Zhu H., Jin, X. et al. (2019). "Neural Packet Classification". UC Berkeley, Johns Hopkins University [online] Available at: https://arxiv.org/pdf/1902.10319.pdf [Accessed 08 Jun. 2020].

[13] Lee, J., Kim J., Kim, I. et al. (2019) "Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles". IEEE Access, vol. 7, pp. 165607-165626 [online at: https://ieeexplore.ieee.org/abstract/ document/8896978 [Accessed 09 Jun. 2020].

[14] Soysal, M., Schmidt, E. G. (2010) "Machine learning algorithms for accurate flow-based network traffic classification". Performance Evaluation, Vol. 67, Issue 6, pp. ages 451-467 [online at: https://www.sciencedirect.com/science/article/pii/; Accessed 14 Jun. 2020].

[15] https://www.netresec.com/?page=PcapFiles

[16] Liu, Y. (2012) "A Survey of Machine Learning Based Packet Classification", The Inst. Comp., Information and Cognitive Systems Univ. of British Columbia [online at:

http://blogs.ubc.ca/ computersecurity/files/2012/04/A-Survey-of-Machine-Learning-Based-Packet-Classification-Yu-Liu.pdf; Accessed 14 Jun. 2020].

[17] Varghese, S. P., Joseph, J. (2014) “Automated Packet Classification and Layer Identification of Network Packets a Review”, IOSR J. of Comp. Eng. [online at: https://pdfs.semanticscholar. org/3d17/; Accessed 14 Jun. 2020].

[18] TensorFlow Core v2.2.0 (2020). [online at: https://www.tensorflow.org /api_docs /python/tf/keras/layers/Dense; Accessed 09 Jun. 2020]

[19] Brownlee, J. (2017) “How to Use the Keras Functional API for Deep Learning”, Machine Learning Mastery [online at: https:// machinelearningmastery.com/keras-functional-api-deep-learning/. Accessed 09 Jun. 2020].

[20] Fruhwirt, P., Schrittwieser, S., Weippl, E. R. (2012). “Using machine learning techniques for traffic classification and preliminary surveying of an attacker’s profile”, SBA Research, St. Polten University, Viena. [online at: https://publications.sba-research.org/publications/using %20machine%20learning_paper.pdf; Accessed 09 Jun. 2020].

[21] Kraus, M. (2019) “Keeping your ML model in shape with Kafka, Airflow and MLFlow”. Medium [online at: https://medium.com/vantageai/keeping-your-ml-model-in-shape-with-kafka-airflow-and-mlflow-143d20024ba6; Accessed 17 Dec. 2020]