What we learn from learning - Understanding capabilities and limitations of machine learning in botnet attacks

David Santana, Shan Suthaharan, and Somya Mohanty

***Abstract*—With a growing increase in botnet attacks, computer networks are constantly under threat from attacks that cripple cyber-infrastructure. Detecting these attacks in real-time proves to be a difficult and resource intensive task. One of the pertinent methods to detect such attacks is signature based detection using machine learning models. This paper explores the efficacy of these models at detecting botnet attacks, using data captured from large-scale network attacks. Our study provides a comprehensive overview of performance characteristics two machine learning models — Random Forest and Multi-Layer Perceptron (Deep Learning) in such attack scenarios. Using Big Data analytics, the study explores the advantages, limitations, model/feature parameters, and overall performance of using machine learning in botnet attacks / communication. With insights gained from the analysis, this work recommends algorithms/models for specific attacks of botnets instead of a generalized model.**

Keywords: Botnet detection, Network Security, Machine Learning, Ensemble Methods, Deep-Learning

I. Introduction

In an increasingly connected world of machines where usability takes presence over security, computers/machines have become more vulnerable to exploits by malicious entities. A large portion of such machines are utilized for their computational and connectivity resources to conduct synchronized attacks across networks. Botnets are a group of exploited machines (with malicious software) that are leveraged for large scale attacks across different cyber-infrastructure components [1].

Attacks from botnets include email/message based spam, deploying malware, executing distributed denial of service (DDoS) attacks, and even spreading the botnet software to more computers [2]. In Q1 2017 [1], cyber-infrastructure resources in almost 72 countries were targeted using botnets. The Federal Bureau of Investigation [3] reports that 18 computers per second are participating in botnets, amounting to 500 million computers worldwide. With growing participation (maliciously exploited) of connected devices in the form of Internet-Of-Things (IoT) in botnets [4], detection of such attacks have been key towards securing infrastructure.

While machine learning [5] and network modelling techniques [6] have been widely used towards detection of botnet attacks, most of the approaches are tailored towards detection of specific types of such attacks. The paper explores the advantages and disadvantages of such models in a real-world scenario. More specifically, utilizing real network data captured across a computer network [7], for seven different types of botnet attacks, the paper examines the utility of different machine learning models in detecting such attacks.

Using information collected from Netflow records [8], the approach examines the models across standard performance/accuracy measures of - Precision, Recall, and F1-Measure for *attack* versus *non-attack* records using supervised models. The analysis removes any device/user identifiable information from the records by using aggregate statistics and does not perform any deep-packet payload inspection. The limitations were purposely put in place for developing efficient models capable of operating in real-time scenario with limited resources (such as routers).

This study also aims to identify the ideal parameters of detection of specific attacks by conducting exhaustive search of parameters and features for the machine learning models. Big Data techniques were utilized to compute the performance characteristics of the models using parallelization techniques. The goal is to improve detection of botnet attacks in real-world real-time systems, where intrusion detection systems (IDS) can be pro-active utilizing the appropriate models for detection of attacks and eventual mitigation.

II. Background

The most common model for operating botnets is the command and control (C&C) architecture [9]. The controller is a master who issues commands for the distributed architecture of bots through legitimate In-

David Santana is with the University of North Carolina at Greensboro, Greensboro, USA (email: d_santan@uncg.edu).
Shan Suthaharan is with the University of North Carolina at Greensboro, Greensboro, USA (email: s_suthah@uncg.edu)
Somya D. Mohanty is with the University of North Carolina at Greensboro, Greensboro, USA (email: sdmohant@uncg.edu)

ternet Relay Chat (IRC) channels. The predominant detection methods for botnets are based on analyzing communication/traffic streams. They include statistical approaches, where anomaly detection [10] is utilized for identifying abnormal activity related to botnets. More recently, machine learning methods [11] have been used to classify network traffic (normal/attack). Livadas *et al.* [12] utilized multi-stage Bayesian network classifiers to detect botnets, where the first stage is utilized to characterize IRC traffic, and the second for botnet communication. Saad *et. al.* [13] compared the performance of five different supervised machine learning algorithms to detect P2P botnet attacks. The BClus methods proposed by Garcia *et. al.* [14], are a unique approach towards utilizing unsupervised clustering algorithms to annotate botnet attacks. Supervised machine-learning approaches such Random Forest [15] and Deep-Learning [16] models have been explored by various studies establishing their effectiveness in analyzing network based communication/traffic. However, a comparative analysis on the advantages and disadvantages of each of the approaches has been lacking.

Id	Time (hrs)	#Packets	#Netflows	Size	Bot	#Bots	IRC	SPAM	DDOS
1	6.15	71.9 M	2 M	52GB	Neris	1	✓	✓	
2	4.21	71.8 M	1.8 M	60GB	Neris	1	✓	✓	
3	66.85	167.7 M	4.7 M	121GB	Rbot	1	✓		
4	4.21	62 M	1.1 M	53GB	Rbot	1	✓		✓
5	11.63	4.4 M	1.2 M	37.6GB	Viruit	1		✓	
6	5.18	115.4 M	2.7 M	94GB	Neris	10	✓	✓	
7	4.75	90.3 M	1.3 M	73GB	Rbot	10	✓		✓
8	0.26	6.3 M	107 K	5.2GB	Rbot	3	✓		✓
9	16.36	50.8 M	1.9 M	34GB	Virut	1		✓	

Table I: Botnet Data - Netflow records of communication

A key hurdle in evaluating the capabilities of detection models is the availability of real-world datasets. While the NSL-KDD (KDDCUP-99) [17] has provided researchers with labeled data, including Denial Of Service (DOS), unauthorized access, remote access, and probing attacks, it does not represent the different varieties of attacks current botnets are capable of.

The CTU-13 dataset [14] is a comprehensive dataset representing different scenarios of botnet attacks. The CTU-13 dataset, was developed for a comparative measure of detection performance among various algorithms. The dataset contains real-world network capture of thirteen different types of attacks conducted using different botnets (Neris, Rbot, Virut, etc.). The dataset is in the form of bidirectional netflow records capturing — Start Time, End Time, Duration, Source IP address, Source Port, Direction, Destination IP address, Destination Port, State, SToS, Total Packets and Total Bytes of each connection in the network.

For comparative measure of detection performance in different scenarios of botnet attacks, the records have been labeled with — Normal, Background, C&C, and Botnet flows. The dataset represents a multitude of characteristics of botnet attacks and their communications. Table I, denotes the different types of communication and attack characteristics which covers SPAM and DDoS attacks, and IRC communication. As a result the dataset provides for an excellent opportunity to study and compare the abilities of different machine learning models in detecting the different characteristics of the attacks.

III. Approach

Utilization of machine learning models in detecting botnet operation in a network can be categorized into two generalized approaches — 1) Payload, and 2) Traffic based. As the name suggests, the payload based approach trains models based on features extracted from the payload / data component of the packets transmitted across the network. The drawbacks of such a models are the resource intensive nature (where every packet has to be analyzed for features), issues with privacy, and encrypted data (features cannot be extracted). The traffic based approach aims at alleviating some of the drawbacks of the model by analyzing the packet headers or Netflow records of communication. While privacy is still a concern with such an approach (such as individual IP addresses in features), this can be mitigated by using aggregation of records for time windows.

Our approach utilizes the temporal domain of the Netflow records by developing features for time-windows in a network. We develop aggregated features for time durations/windows which are then utilized to train machine-learning models. For each dataset, the features of the aggregated Netflows are calculated by taking into account all Netflows that were initiated or were continuing (initiated in a earlier window but currently active) in a particular window. The values of each Netflow record which falls into either of the two categories were taken into account for developing features of the window.

The 13 datasets were merged into 4 different categories based on the category of attack/communication — 1) DDoS (4, 7, 8), 2) Spam (1,2,5,6,9), and 3) IRC (1,2,3,4,6,7,8). The target label for each of the records of the dataset were developed in the same way as features for the time-window, considering Botnet and C&C records as *'attack'* labels and only the normal records as *non-attack* label. In other words, if a single Netflow record in the time-window has Botnet and/or C&C tags, the entire window is considered as an attack window. If no records had those tags in the data, the window is considered non-attack. Further more, multiple datasets were created (from the merged) data using different window sizes (0.01, 1.0, 10, 30, 60 seconds) for aggregation and evaluation of the models.

This dataset did not use any records in the original dataset tagged as 'background'. The goal here was to evaluate the efficiency of the developed models on verified records which have been identified by subject matter expert (SME). With the best developed models (using feature and parameter tuning) in such an scenario, the effectiveness of the models can then be tested in an environment where a large portion of the data is unverified. For this propose, a secondary dataset was developed which considered background tag into developing the time-window based features and labeling. In this case, the background tag was assumed to be 'non-attack' traffic considering all of the attack traffic was labeled by the SME in the environment.

A. Feature Extraction

The identified features of the study can broadly be classified into two categories — 1) Extracted , and 2) Analyzed features. The extracted features record simplistic observations of connections within a time-window. Most of the features in this category are general counts ($\mathcal{N}$) or mean ($\bar{\mathcal{W}}$) values are aggregated over connections within a time-window. Table II, describes the various categories of such features, where observations of 1) Connection Information, 2) IP Address, 3) Port, and 4) Protocols used by the Netflow records are used. Extracted features in the connection category record counts of different types of connection and mean duration ($\bar{\mathcal{W}}\_duration$) for each window. In the IP address category, aggregated counts of different classes of source and destination IP address classes are kept. Similarly, port variables record count of utilization of different source and destination ports above and below 1024 (below 1024 are well-known ports). Protocols used by the connects are also kept track of in the variables representing counts of TCP ($\mathcal{N}\_TCP$), UDP ($\mathcal{N}\_UDP$), and ICMP ($\mathcal{N}\_ICMP$) connections.

The analyzed category develops in-depth feature transformations using the raw values recorded from each of the netflow records (within a time-window). The feature calculations in the category are based on either entropy ($\mathcal{S}$) or standard deviation (σ) of the features within the time window.

Similar to the extracted features, the analyzed features were also developed for connection information, IP address, and port data. Connection based entropy of packets transferred ($\mathcal{S}\_packets$), the total number of bytes ($\mathcal{S}\_bytes$), bytes from source ($\mathcal{S}\_srcbytes$), connection states ($\mathcal{S}\_state$), and duration ($\mathcal{S}\_time$) were recorded for each time window. For each of the entropy values, standard deviations were also measured (excluding the state entropy) - $\sigma\_packets$, σ_bytes, $\sigma\_srcbytes$, and σ_time. Entropy in IP address were recorded for change within different classes for both source and destination.

Category	Feature	Description
Extracted Features		
Connection Information	1. $\mathcal{N}\_conn$	Number of connections
	2. $\mathcal{N}\_normal\_flow$	Count of normal connections
	3. $\mathcal{N}\_back\_flow$	Count of background connections
	4. $\bar{\mathcal{W}}\_duration$	Mean of duration of all connection
IP Address	5. $\mathcal{N}\_s\_a\_p\_address$	Count of source IP by class
	6. $\mathcal{N}\_s\_b\_p\_add$	
	7. $\mathcal{N}\_s\_c\_p\_add$	
	8. $\mathcal{N}\_s\_na\_p\_add$	
	9. $\mathcal{N}\_d\_a\_p\_add$	Count of destination IP by class
	10. $\mathcal{N}\_d\_b\_p\_add$	
	11. $\mathcal{N}\_d\_c\_p\_add$	
	12. $\mathcal{N}\_d\_na\_p\_add$	
Port Information	13. $\mathcal{N}\_sports>1024$	Count of source ports over and below 1024
	14. $\mathcal{N}\_sports<1024$	
	15. $\mathcal{N}\_dports>1024$	Count of destination ports over and below 1024
	16. $\mathcal{N}\_dports<1024$	
Protocol Information	17. $\mathcal{N}\_icmp$	Count of protocol used
	18. $\mathcal{N}\_tcp$	
	19. $\mathcal{N}\_udp$	
Analyzed Features		
Connection Information	20. $\mathcal{S}\_packets$	Entropy - packets transferred
	21. $\mathcal{S}\_srcbytes$	Entropy - bytes from source
	22 $\mathcal{S}\_bytes$	Entropy - bytes transferred
	23. $\mathcal{S}\_state$	Entropy - connection states
	24. $\mathcal{S}\_time$	Entropy - connection duration
	25. σ_time	Standard deviation - duration per connection
	26. $\sigma\_packets$	Standard deviation - packets transferred
	27 σ_bytes	Standard deviation - bytes transferred
	28. $\sigma\_srcbytes$	Standard deviation - bytes from source transferred
Entropy IP Address	29. $\mathcal{S}\_srcip$	Entropy - source and destination IP addresses
	30. $\mathcal{S}\_dstip$	
	31. $\mathcal{S}\_src\_a\_ip$	Entropy - source IP by class
	32. $\mathcal{S}\_src\_b\_ip$	
	33. $\mathcal{S}\_src\_c\_ip$	
	34. $\mathcal{S}\_src\_na\_ip$	
	35. $\mathcal{S}\_dst\_a\_ip$	Entropy - destination IP by class
	36. $\mathcal{S}\_dst\_b\_ip$	
	37. $\mathcal{S}\_dst\_c\_ip$	
	38. $\mathcal{S}\_dst\_ns\_ip$	
	39. $\mathcal{S}\_src\_to\_dst$	Entropy of source to destination IP with duration and bytes transferred
Entropy Port	40. $\mathcal{S}\_srcport$	Entropy - source and destination ports
	41. $\mathcal{S}\_dstport$	
	42. $\mathcal{S}\_sports>1024$	Entropy - source ports
	43. $\mathcal{S}\_sports<1024$	
	44. $\mathcal{S}\_dports>1024$	Entropy - destination ports
	45. $\mathcal{S}\_dports<1024$	

Table II: Extracted Features - Feature variables from the netflow records aggregated by time window

Apart from the standard entropy calculation for the IP addresses, $\mathcal{S}\_src\_to\_dest$ was developed to map single source to destination connections along with their connection duration and amount of data transferred in each connection. Changes in port utilization within a time-window was also captured as entropy variables for source and destination and utilization standard versus unknown ports.

The final output of the feature extraction stage represents the following data:

$$f_v^i = < x_1^i \cdots x_{47}^i > : y^i = l^i \tag{1}$$

Where, f_v^i is the feature vector for the window i, l^i is the label of the window represented by y^i, where

$$l^i = \begin{cases} y^i : 0 \text{ if attack flows in window}(i) = 0 \\ y^i : 1 \text{ if attack flows in window}(i) \geq 1 \end{cases} \quad (2)$$

B. Building Machine Learning Models

The machine learning algorithms explored in here are - 1) Random Forest and 2) Multi-Layer Perceptron (Deep Learning). A Random forest model can be defined as $h = \{h_1(X), \cdots, h_j(X)\}$, where an ensemble of decision tree classifiers ($h_j(X)$, with j trees) are used to make collective decisions. X is the model training data of the features vectors (f_v^i) and their corresponding labels (l^i).

Multi-Layer Perceptrons (MLP) [18], [19] use feed-forward artificial neural networks consisting of an input layer, multiple hidden layers, and an output layer. Each layer is made up of perceptrons which compute a single output (γ) based on a input vector defined by $\gamma = \varphi(\sum_{i=1}^{n} w_i x_i + b)$, where w denotes the vector of weights for the perceptron, x is input vector (either from the actual input vector or previous layer outputs), b being the bias, and φ the activation function. The weights and biases are adjusted in the perceptrons in each layer based on the training input vector (f_v^i) and their corresponding labels (l_i) using back and forward propagation.

For each of the models, the entire dataset was separated into training data (70% of the dataset) for model development, and test data (remaining 30%) for validation. For the Random forest, the number of estimators/decision trees used in the initial model was $j = 10$, maximum of 6 features per estimator, no maximum depth for the trees, and requires two samples to split an internal node. The MLP model, was based on 4-layer approach, with one input, 2 hidden layers each followed by a dropout layer of 50% dropout (to prevent overfitting), and one output layer. Each layer utilizes Rectified Linear Unit (ReLU) as its activation function φ, and binary cross-entropy for the loss function. Performance of each model was recorded for their accuracy (overall classification), precision ($\frac{tp}{tp+fp}$), recall ($\frac{tp}{tp+fn}$), and F-1 score ($\frac{2.precision*recall}{precision+recall}$), where tp is the number of true positive classified of attack labels, fp is the number of false positives classified on attack labels, fn is the number of false negatives classified by the models.

Different window sizes for the feature vectors (f_v^i) in the data were explored while evaluating the performance of the model. Window sizes ranging from 0.1 all the way with 60 second intervals were evaluated for different types of attacks/communication (DDoS, SPAM, and IRC) with the deep learning and random forest algorithms. For each of these intervals the F1-Score was recorded for cross-evaluation. An exhaustive search was also conducted for evaluation of hyper-parameter values and architecture layouts which yield better performance of detecting attack windows in the learned models. In the random forest models, a grid search method was used, where multiple estimators (10 - 700) were evaluated for their performance. Other parameter values such as min-split of leaves, max depth of the estimators, and number of features per estimators were also evaluated for their F-1 score. In the MLP model, multiple architectures with different number of hidden layers, number of perceptrons per layer, and different activation functions were tested (Tanh, Sigmoid, LeakyReLU) were tested for their accuracy.

Each of the models were also run through a k-fold cross validation ($k = 10$) with random selection of training and test datasets, where the mean scores of the models were taken into account for performance metric comparisons.

IV. Results and Analysis

A. Initial Results

Table III describes the overall performance of the Random Forest and Deep Learning (MLP) models. The reported values shown in the table are for base/initial models with single window size for all attacks (1 sec) and default model parameters as described in Section III-B. In the case of Multi-Layer Perceptron model, where a binary classifier is used to predict windows with attack/non-attack features, the resulting model is able to perform with an overall F1-score (value between 0 - 1) of *.34* in DDoS, *.93* in SPAM, and *.91* in IRC data. The model performs very poorly on the DDoS data indicated by low recall score of *.26*, suggesting its inability to correctly identify attack windows. In case of SPAM and IRC, the model is able to perform well with good accuracy score.

In comparison, Random Forest consistently performs well on each of the data with F1-scores of *.97* in DDoS, *.99* in SPAM, and *.98* in IRC data. While the accuracy of the model is high, we can notice the performance of the model for DDoS attack is lower when compared others. These issues lead to a further investigation of the models on their window sizes for each of the datasets.

B. Model/Feature Tuning

Figure 1 describes the performance (mean F1-score) of the models in different scenarios of attacks (DDoS, SPAM, and IRC) with different window sizes for aggregation of features. In case of DDoS, the both Random Forest and MLP have higher F1-scores with lower window size (in initial model was 1 sec) of 0.01 second. Increasing the time windows to be greater than 0.01 seconds, leads to significant degradation in DDoS detection performance. This can be attributed to lower time duration of individual DDoS connections, where smaller window size

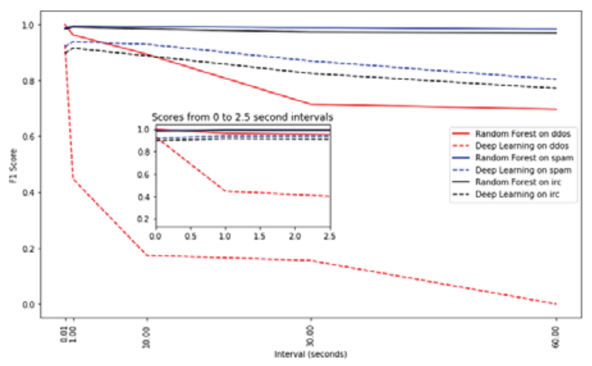

Figure 1: Window Size versus F1-score: Analysis of model performance in different window size aggregation.

will lead to accurate representation of attack features, and in turn lead to better model performance. For the SPAM and IRC datasets, the analysis did not show any benefits of changing the default 1 second interval for window sizes, instead lead to inferior performance (drops in F1-score) at lower (0.01) and higher (10, 30, and 60 second) time windows.

Analysis of the individual model hyper-parameters did not show any significant increase in performance in both the algorithms. In Random Forest increase in the number of estimators showed only 0.01 gain in model accuracy. Similar non-significant increase in model performance was observed in MLP, where larger networks of perceptrons and layers were used. The models were analyzed for the reported prediction probability values instead of just the default to detect all attack windows. The decision threshold of detecting attack window was decreased to 0.3 (based on the Recall scores), where window with predicted probability of attack is $1 \geq 0.3$ is taken to be an attack window, and in turn classifies a window to be normal if predicted probability of $0 \leq 0.7$. While this decreases the F-1 score of the normal windows (0), it aims for increased accuracy in detecting attack windows 1.

C. Final metrics

Table III outlines the performance metrics of tuned models. With the window size tuning we see improvements in performance across all models (as compared to initial values), especially in the DDoS data. The MLP models gained a significant boost in their performance in detecting DDoS attack from an F1-score of *0.34* in initial models to *0.95* in 10 millisecond window size aggregation. As a result, the combined score of the MLP model also improves to 0.93.

In the case of Random Forest, the models perform on nearly perfect accuracy across all types of attacks. In the case of DDoS we observe an increase in accuracy by 2% where F1-Score increase from *0.97* to *0.99*. With the probability tuning, Figure 2 shows the accuracy of Random forest in detecting attack windows with only 8 windows mis-classified as normal while getting 9,671 windows accurately detected (overall 17 windows incorrectly classified out of14,927) in training data. Similar results were also observed in detecting SPAM and IRC windows.

Attack Type	**Random Forest**	**Deep learning**
	Accuracy - Precision - Recall - F1	*Accuracy - Precision - Recall - F1*
	Initial Untuned Model	
DDOS	0.9764 - 0.9793 - 0.9329 - 0.9708	0.7835 - 0.9837 - 0.2652 - 0.3431
SPAM	0.9905 - 0.9913 - 0.9927 - 0.9942	0.9298 - 0.9219 - 0.9586 - 0.9399
IRC	0.9862 - 0.9864 - 0.9856 - 0.9912	0.9169 - 0.9320 - 0.9027 - 0.9171
	Tuned Model	
DDOS	0.999 - 0.999 - 0.999 - 0.999	0.942 - 0.946 - 0.965 - 0.956
SPAM	0.993 - 0.993 - 0.996 - 0.994	0.907 - 0.962 - 0.879 - 0.918
IRC	0.992 - 0.989 - 0.995 - 0.992	0.895 - 0.925 - 0.857 - 0.889

Table III: Performance Metrics of Models: Accuracy, Precision, Recall, and F1-score of initial/untuned versus tuned models

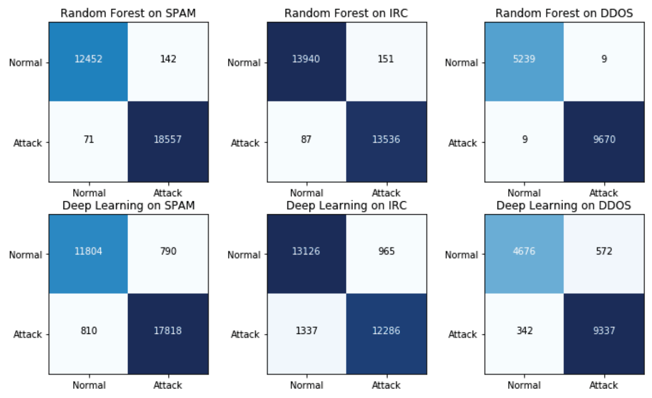

Figure 2: Confusion Matrices Of Each Model On Each Attack: Prediction matrix of Deep Learning and Random Forest with tuned parameters.

V. Discussion

A. Feature Tuning

Analysis of performance metrics observed for feature (window) and model tuning suggests large gains in model detection accuracy can be observed by using different window sizes for feature aggregation. In comparison, model hyper-parameter tuning provide negligible performance benefits, and can in some cases lead to increase in complexity of the underlying model. Our analysis suggests using 0.01 sec window size for DDoS, and 1 sec windows for SPAM and IRC, in order to get the best performance out of the trained models. In a real-world scenario, this would involve training multiple models with different aggregated window size based features. A general model which uses a single aggregation size will not be suitable for detection of every type of attack, as shown by our initial results.

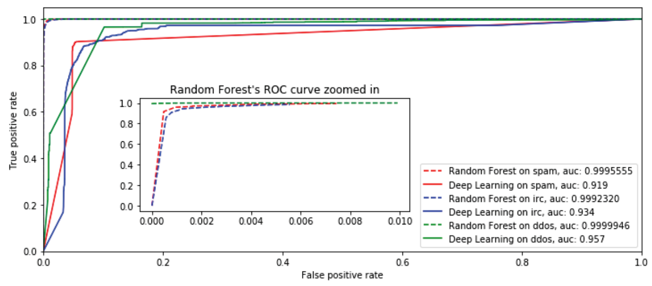

Figure 3: Multi-Layer Perceptron & Random Forest ROC curve: ROC curve for each attack type (tuned model).

B. Model Performance

Based on the performance metrics of the tuned models, it can clearly be observed that Random Forest based models outperform MLP models in detecting attack windows in every category. This is especially highlighted in the model Receiver Operating Characteristics (ROC) curves outlined in Figure 3. While, the Area Under the Curve (AUC) values for MLP are above *0.90* for all types of attacks, the model performs the worst on detection of SPAM attacks. MLP models work well in detecting DDoS attack windows with AUC score of *0.97*, closely followed by botnet communication detection of IRC at *0.95*. In comparison, the Random Forest models perform at near perfect accuracy with AUC scores of DDoS - *1.0*, IRC - *0.99*, and SPAM - *1.0*.

The relative poor performance of MLP models can partially be attributed to the number of available training samples. MLP models (and more generally Deep Learning) greatly benefit from large amounts of training data and can outperform traditional models in such a case. This was also observed in detecting DDoS attacks, where 10 millisecond window size aggregation lead to increased model performance. While dataset used in our experiments were limited, in large organizations this would note be the case, and MLP based methods can be used for detection of attacks. In scenarios, where amount of labeled data is limited ensemble based methods such a Random Forests will outperform Deep-Learning based approaches as shown by our results.

C. Feature Importance

Diving deeper into the characteristics of the Random Forest model, Figure 4 shows the importance of top ten features in each attack category. In the case of SPAM, the mean duration of the connections ($\bar{\mathcal{W}}$_duration) play an important role in detecting attacks, followed by standard deviation of connection duration (σ_time), entropy of destination IP address ($\mathcal{S}$_time) suggesting

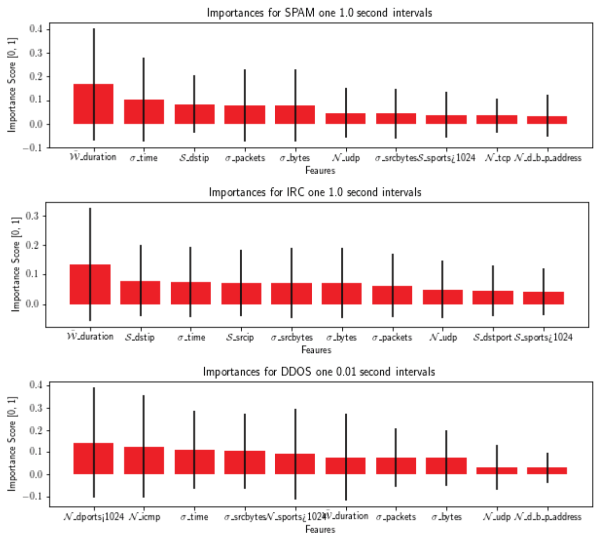

Figure 4: Feature Importance Random Forest - Importance of features in different types of attack with normalized scores (0 - 1)

SPAM messages sent out to large number of destinations, standard deviation of packets (σ_packets) - difference in the number of packets in attack versus normal windows, and bytes (σ_bytes) transferred within a time window where SPAM communication will see a larger amount transferred. While other features (feature number - 19, 21, 42, 18, 10) play a role in detection of the SPAM based attacks they are of less significance from the top five features.

Similarly, in IRC detection, the mean duration of the connections ($\bar{\mathcal{W}}$_duration) again is of significant importance followed by entropy of destination IP address ($\mathcal{S}$_destip) suggesting a controller communicating with a large number of bots, standard deviation of connection duration (σ_time) where communication windows have smaller connection durations, entropy of source IP address ($\mathcal{S}$_srcip) - a few/single controller communicating to bots, standard deviation of bytes (σ_bytes), and packets (σ_packets) transferred in communication during attacks. $\mathcal{N}$_udp also plays a role as UDP is one of the preferred modes of communication in IRC.

In DDoS attack scenario, the models rely on identifying the windows where communication exists destination ports $\geq$ 1024 ($\mathcal{N}$_dports>1024). Also, in DDoS based attacks the number of ICMP based flows increases as identified by the $\mathcal{N}$_icmp feature importance. Standard deviation of flow duration (σ_time) - short flows for DDoS, amount of source bytes transferred ($\mathcal{N}$_srcbytes), number of source ports greater than 1024 ($\mathcal{N}$_sports>1024), mean duration of flows ($\bar{\mathcal{W}}$_duration) , and standard deviation of packets (σ_packets) - bytes (σ_bytes) play key roles in identifying attack windows.

The analysis suggests the models can perform accurate detection of the attack/communication windows with limited number of features (in comparison to the previously identified 45 features), which be used to store lower number of variables thereby considerably decreasing the size of the feature set, and in turn reducing the complexity of the machine learned models.

VI. Conclusion

We analyzed the capabilities and limitations of two key machine learning models in their ability to detect attacks in performed over network communication. Using Big Data approaches to develop features (from large datasets) and train - test machine learning models, we were able to identify operating characteristics of the learned models in real-world scenario. Our analysis of the models found feature engineering to be a key contributor to improving machine learning models, where the biggest increase in model accuracy resulted from changes in window sizes. Exploration of hyper-parameters related to the model did not show any significant increase both Random Forest and MLP models. Comparing the models itself, we observe Random Forest is capable of operating at near perfect accuracy in detecting different types of botnet attacks/communication when compared to MLP models. The limitation of annotated data plays a key role in the lower performance of MLP models, so in scenarios where there is availability of large annotated data (using honey-pots, subject matter expert, detection over time) MLP based models could potentially be effective. However, in scenarios where there is limited availability of such annotated data, Random Forest models will outperform their counterparts.

The study also observed multiple aggregation windows sizes perform the best for different types of attacks. While this is negates the utilization of a generalized model, in case of resource constraints (space and performance issues related to multiple datasets and models), a generalized dataset using 0.01 second aggregation can be used with best performance in DDoS attacks, and negligible degradation in accuracy for SPAM and IRC. In order to further solve the dataset / model complexity, lower number of features can also be used for model training as identified by out feature importance analysis.

Future efforts include analyzing the performance of models when we have unlabeled background traffic and/or highly imbalanced data. We also plan to evaluate of other types of attacks/communication such as peer-to-peer, click-fraud, and other botnet/malware related network data, and use the metrics for comparison to standard detection tools (such as Bot-Sniffer [10]).

VII. Acknowledgement

This research was supported by University of North Carolina - Greensboro new faculty start-up funds.

References

[1] O. K. Alexander Khalimonenko, "Ddos attacks in q1 2017," May 2017. [Online]. Available: https://securelist.com/ddos-attacks-in-q1-2017/78285/

[2] P. Wang, B. Aslam, and C. C. Zou, "Peer-to-peer botnets: The next generation of botnet attacks," *Electrical Engineering*, pp. 1–25, 2010.

[3] J. Demarest, "Taking down botnets: Public and private efforts to disrupt and dismantle cybercriminal networks," accessed: 2018-01-08. [Online]. Available: https://goo.gl/3fw38R

[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[5] V. Jyothsna, V. R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26–35, 2011.

[6] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1, pp. 18–28, 2009.

[7] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, sep 2014. [Online]. Available: https://doi.org/10.1016%2Fj.cose.2014.05.011

[8] Cisco, "Cisco netflow," accessed: 2018-01-08. [Online]. Available: http://www.cisco.com/warp/public/732/Tech/netflow

[9] H. R. Zeidanloo and A. A. Manaf, "Botnet command and control mechanisms," in *Second International Conference on Computer and Electrical Engineering*, vol. 1. IEEE, 2009, pp. 564–568.

[10] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic." in *NDSS*, vol. 8, 2008, pp. 1–18.

[11] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 2014, pp. 797–801.

[12] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. 967–974.

[13] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting p2p botnets through network behavior analysis and machine learning," in *2011 Ninth Annual International Conference on Privacy, Security and Trust*, July 2011, pp. 174–180.

[14] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[15] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[16] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST, 2016, pp. 21–26.

[17] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 2009, pp. 1–6.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[19] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.