Network Intrusion Detection in an Adversarial Setting

Shreyansh Singh - 16075052

Aim - To fool Machine Learning based classifiers into falsely predicting malicious network traffic as benign

Intrusion Detection

- Dealing with unwanted access to systems and information by any type of user or hardware.
- Intrusion Detection System (IDS) is a device or software that monitors a network or systems for malicious activity or policy violations.
- There are two major categories of IDS -
  - **Network IDS -** These monitor network segments and analyze the network traffic to detect intruders.
  - **Host-based IDS** - These are installed in host machines and analyze processes, logs and other unexpected changes to detect malicious activity.

Challenges to Network Intrusion Detection

Regardless of the specific dataset used for Intrusion detection analysis, the nature of the data associated with this class of problems exhibits certain general characteristics:

- Data is generated constantly and there is a time series nature (continuous or discrete) based on the dataset and the processing approach.
- *Class imbalance*, i.e. Very few positive labels or lack of labels
- Attack types change a lot over time, with attackers developing novel methods all the time.
- Variety in the type of data: packets, flows, numerical, unstructured text (URLs) and so on.

Intrusion Detection Methodologies

The methodologies used in NIDS can be divided into the following major categories -

- **Signature based** - These systems perform simple signature matching using signatures or indicators extracted from previously known attacks.
- **Anomaly based** - These type of systems try to model normal behavior of the network in contrast to what is anomalous and potentially malicious.
- **Hybrid systems** - These types of systems are a combination of the characteristics of the above approaches.

Challenges to Anomaly-based NIDS

Some challenges are faced by Anomaly-based NIDS are -

- The problem of generalization, which makes it difficult to prove whether they can be used widely in practice.
- Training data is very unbalanced, which makes it difficult to apply unsupervised classification techniques.
- High False Positive rate (FPR) result in a large number of alarms to be analyzed that are generated by the NIDS, therefore, time and fatigue can be a problem.
- Lack of high quality representative datasets can lead to problems in the evaluation of different approaches

Evaluation Metrics

In NID studies, four major categories of evaluation metrics are used in the majority of studies -

- Attack detection accuracy with most common metrics like False Positive, False Negative, True Positive and True Negative rates. The False Positive Rate (FPR) and the True Positive Rate (TPR) are used in the construction of Receiver Operating Characteristic (ROC) curves and the calculation of the Area Under the Curve (AUC)
- Performance overhead which the IDS is adding to the overall network environment.
- Attack coverage, which is the detection accuracy of the IDS without benign traffic.
- Workload processing, which is the amount of traffic that can be processed by an IDS vs. the amount of network traffic the IDS discards.

IDS Datasets

One of the most used datasets is the KDD'99 which was derived from the DARPA'98 dataset. The dataset was used in a competition that was held during the **Fifth International Conference on Knowledge Discovery and Data Mining** and the main competition task was to create a predictive model that can be used in network intrusion detection. But this dataset had certain problems which were analysed by researchers. These issues were -

- There is a huge number of redundant records for about 78% and 75% are duplicated in the train and test set, respectively.
- This redundancy makes the machine learning training quite biased.

IDS Datasets (contd.)

Due to the problems with the KDD'99 dataset, a new dataset was created which was called the NSL-KDD dataset. Although it didn't remove all the problems associated with the KDD'99 dataset, it was still an improvement because of the following reasons -

- No redundant records in the train set, so the classifier will not produce any biased results
- No duplicate record in the test set which have better reduction rates.
- The number of selected records from each difficult level group is inversely proportional to the percentage of records in the original KDD data set.

Adversarial Machine Learning

Introduction

Adversarial Machine Learning (AML) is the study of machine learning in the presence of an adversary that works against the ML system in an effort to reduce its effectiveness or extract information from it. AML can be divided into two main types of attacks -

1. **Evasion (Exploratory) Attacks** - These attacks are performed on the **testing** phase. It does not tamper with ML models, but instead cause it to produce adversary selected outputs
2. **Poisoning (Causative) Attacks -** These attacks are performed on the **training** phase. Attackers attempt to learn, influence, or corrupt the ML model itself.

Introduction (contd.)

Evasion attacks are the most common type of attacks. They can be further be classified as follows-

- **White-box Attacks -** Attackers know full knowledge about the ML algorithm, ML model, (i.e., parameters and hyperparameters), architecture, etc.
- **Black-box Attacks -** Attackers almost know nothing about the ML system (perhaps know number of features, ML algorithm).

Black-Box Evasion Attacks

- **Adversarial Sample Transferability**
  - **Cross model transferability** - The same adversarial sample is often misclassified by a variety of classifiers with different architectures
  - **Cross training-set transferability** - The same adversarial sample is often misclassified trained on different subsets of the training data.
- Therefore, an attacker can
  - First train his own (white-box) substitute model
  - Then generate adversarial samples
  - Finally, apply the adversarial samples to the target ML model

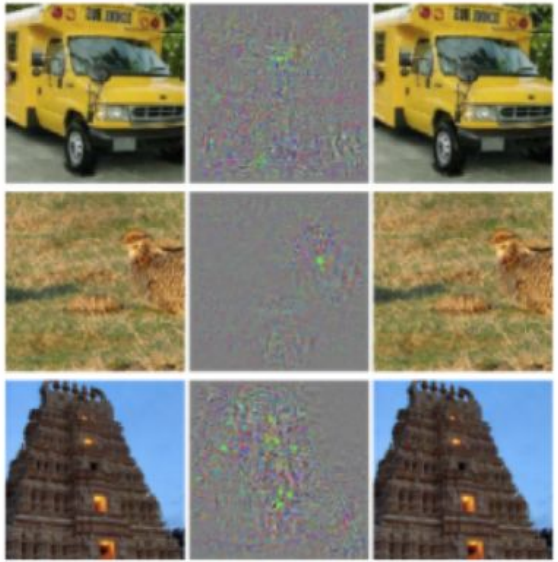
Adversarial Deep Learning

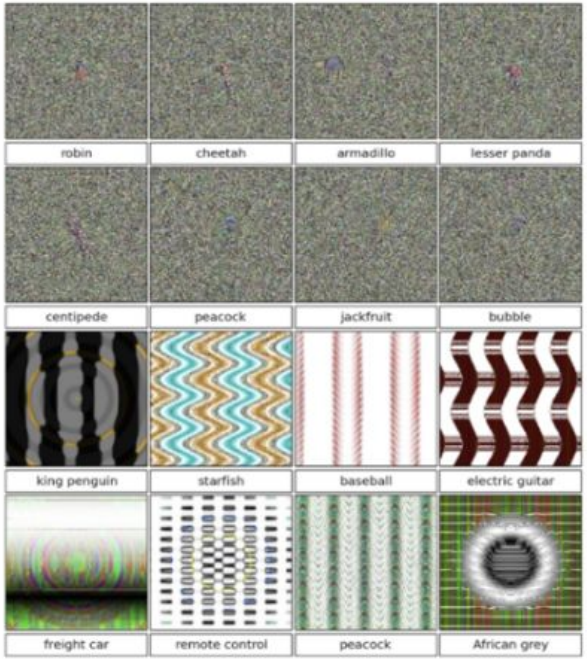
The first breakthrough in Adversarial Deep Learning came in 2013 by Szegedy et. al. where they showed that Deep Learning classifiers can be fooled by introducing small variations in an image. These variations were so small that they were were imperceptible to humans but enough to fool the classifiers.

Another work by Nguyen et. al. showed in 2015 that it was possible to use randomly generated images which had patterns in it, which did not mean anything, to fool the Deep Learning classifiers into predicting them into valid object classes.

Adversarial Deep Learning (contd.)

Adversarial example produced by image perturbation. The neural network believes the images on the right are ostriches. (Szegedy et. al. 2013)





Images generated using evolutionary algorithms. (Nguyen et. al. 2015

Adversarial Deep Learning (contd.)

- Researchers discovered that the images that show adversarial properties for one neural network can transfer these properties to other neural networks trained separately.
- The only models that have shown some resistance to adversarial examples are the Radial Basis Function (RBF) networks but they are not used often as they don’t generalize well.
- Other than those, even shallow linear models are also affected by the same problem.

Adversarial Examples Generation

Research has been done on the methods and algorithms to generate adversarial examples. There are many such methods which have a trade-off on speed of production, performance and complexity. Some of the methods that have been proposed are given below -

- **Evolutionary algorithms** - Proposed by Nguyen et. al. in 2015. But this method is very slow compared to the other two alternatives described below.
- **Fast Gradient Sign Method (FGSM)** - Proposed by Goodfellow et. al. in 2014
- **Jacobian-based Saliency Map Attack (JSMA) -** Proposed by Papernot et al. in 2016. This method is more computationally expensive than FGSM but it has the ability to create adversarial samples with less degree of distortion.

Fast Gradient Sign Method (FGSM)

- In FGSM, a perturbation δ is generated by computing the gradient of the cost function J in respect to the input x :

$$\delta = \epsilon sign(\nabla_x J(\theta, x, y))$$

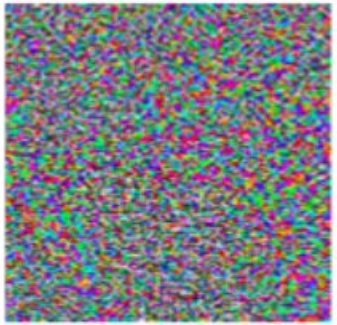
  where θ are model parameters, x is the input to the model, y are the labels associated with x, ϵ is a very small value and $J(\theta, x, y)$ is the cost of the function used when training the neural network.
- This method is very fast because it requires the gradient which can be computed very efficiently using backpropagation.
- The perturbation is then added to the initial sample and the final result produces a misclassification.

FGSM Example

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Generating adversarial samples with FGSM (Goodfellow et. al. 2014)

Jacobian-based Saliency Map Attack (JSMA)

- JSMA, as the name suggests, generates adversarial sample perturbations based on the concept of saliency maps. The direction sensitivity of the sample in regards to the target class is calculated using a saliency map.
- Basically, the algorithm works by trying to determine which input features will be most likely to create a targeted class change.
- Using this sensitivity map, one or more features are chosen as possible perturbations and the model is checked to establish whether or not this change resulted in a misclassification.
- If it does not result in a misclassification, the next most sensitive feature is selected and a new iteration occurs until an adversarial sample that can fool the network is generated.

Since the method usually takes a number of iterations, it is not as fast as FGSM.

JSMA (contd.)

- Papernot et. al. designed an efficient saliency adversarial map under the L_0 distance (i.e. the number of features *i* such that $x'_i \neq x_i$) . The Jacobian matrix computed for a given sample *x* is expressed as -

$$J_f(x) = \frac{\partial f(x)}{\partial x} = [\frac{\partial f_j(x)}{\partial x_i}]_{i \times j}$$

- In this way, the input features of *x* that made most significant changes to the output can be identified.

Adversarial Example Generation using FGSM & JSMA

- Both FGSM and JSMA operate under the threat model of a strong attacker, e.g. an attacker that has knowledge of at least the underlying model.
- However, If the attacker has only access to the model output and has some knowledge of the input to be provided, he can use the output of the model with different inputs to create an approximation of the model.
- Since the adversarial attacks have the transferability property, it is possible for the attacker to craft adversarial samples on the approximated model which can later be used as attack vectors against the original model

Data Collection and Analysis

KDD’99 and NSL-KDD

The attacks that are present in the datasets can be divided into four major categories -

- **Denial of Service (DoS) Attacks** attacks are an interruption in an authorized user’s access to a computer network, in other words, they are attacks against availability. This category contains attacks such as *smurf*, *neptune*, *mailbomb*, *udpstorm*, etc.
- **User to Root (U2R) Attacks** indicate attempts of privilege escalation. Some attacks of this type in the dataset are *buffer overflow, loadmodule, sqlattack* and *rootkit*.
- **Root to Local (R2L) Attacks** attacks aim to gain remote access to a system by exploiting a vulnerability. Some examples of this type of attacks are *multihop, guesspasswd, httptunnel and xsnoop.*
- **Probe** attacks aim to gather information by using enumeration techniques like scanning or probing different parts of the network, for e.g. the ports. Some examples of such types of attacks are *ipsweep, portsweep, nmap and mscan.*

Feature	Type	Feature	Type
duration	cont.	is_guest_login	sym.
protocol_type	sym.	count	cont.
service	sym.	srv_count	cont.
flag	sym.	serror_rate	cont.
src_bytes	cont.	rerror_rate	cont.
dest_bytes	cont.	srv_rerror_rate	cont.
land	sym.	diff_srv_rate	cont.
wrong_fragment	cont.	srv_diff_host_rate	cont.
urgent	cont.	dst_host_count	cont.
hot	cont.	dst_host_srv_count	cont.
num_failed_logins	cont.	dst_host_same_srv_rate	cont.
logged_in	sym.	dst_host_diff_srv_rate	cont.
num_compromised	cont.	dst_host_same_src_port_rate	cont.
root_shell	cont.	dst_host_srv_diff_host_rate	cont.
su_attempted	cont.	dst_host_serror_rate	cont.
num_root	cont.	dst_host_srv_serror_rate	cont.
num_file_creations	cont.	dst_host_rerror_rate	cont.
num_access_files	cont.	dst_host_srv_rerrorv_rate	cont.
num_outbound_cmds	cont.	is_host_login	sym.

KDD'99 and NSL-KDD features

Features in the Dataset

The features in the dataset can be divided into three main categories -

- **Basic** features are the ones related to connection information such as hosts, ports, protocols and services used.
- **Traffic** features are calculated during a window interval as an aggregate. A further subdivision is “aggregates based on the same host” and “aggregates over the same service”. In the NSL-KDD dataset, the time window (in KDD’99) was substituted with a connection window of the last 100 connections.
- **Content** features are extracted from the payload or packet data and they are related to the content of specific applications or protocol used.

Data Preprocessing

The following steps were followed for data preprocessing -

- One-Hot encoding was used to convert the categorical features to numerical features.
- All the features (now all numerical) were normalized using Min-Max Scaler as very large values can dominate the dataset and affect the performance of certain classifiers like SVM and the MLP.
- The dataset had labels consisting of 39 distinct attack categories. These attacks were grouped into four major families - “DoS”, “U2R”, “R2L” and “Probe”. Hence the problem was transformed into a five-class classification problem (including the “normal” class).

After preprocessing, the final number of features are **122**. The number of data points in the training set are **1,25,973** and in the test set **22,544**.

Results

Baseline Models

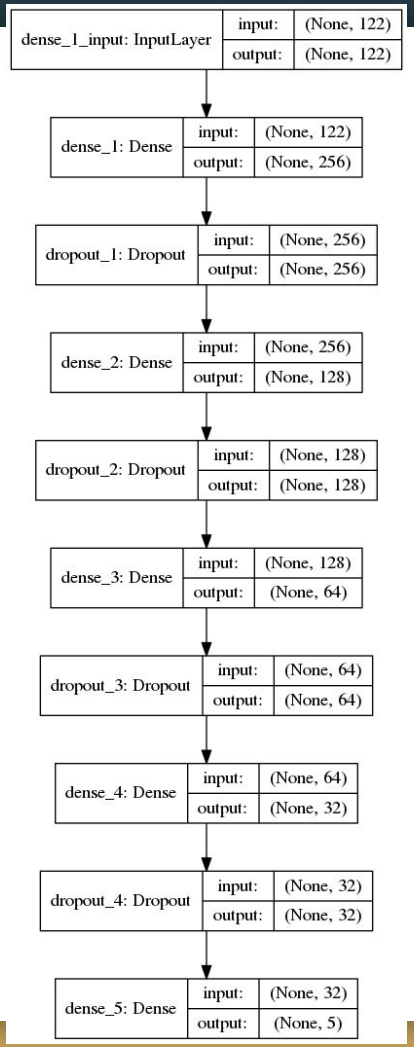
A number of different were trained and tested on the NSL-KDD dataset to establish a baseline. The results are shown below -

Method	Accuracy	F1-Score	AUC (normal)
Decision Tree	0.989	0.992	0.992
Random Forest	0.993	0.994	0.994
Linear SVM	0.945	0.958	0.954
Voting Ensemble	0.993	0.993	0.746
MLP	0.985	-	-

Adversarial Test Set Generation

- Used both FGSM and JSMA
- A pre-trained MLP was used as the underlying model.
- Following table shows the difference between the two methods in terms of changed features on average as well as the unique features changed for all data points in the test set.

Method	Num. of unique altered features	Avg. features changed per data point	Percentage of altered features
FGSM	122	76.47	62.68
JSMA	89	11.58	9.49

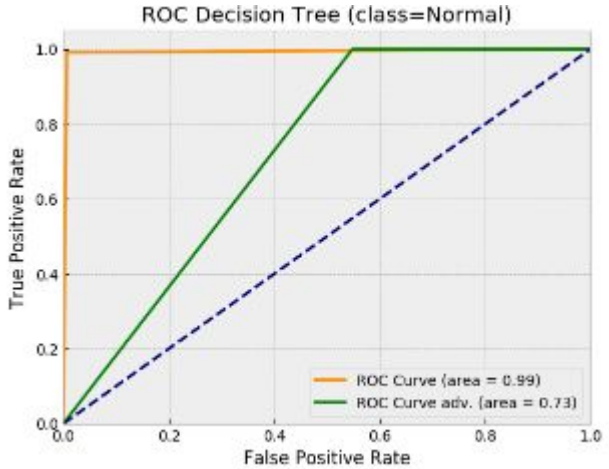


Architecture of the underlying MLP model

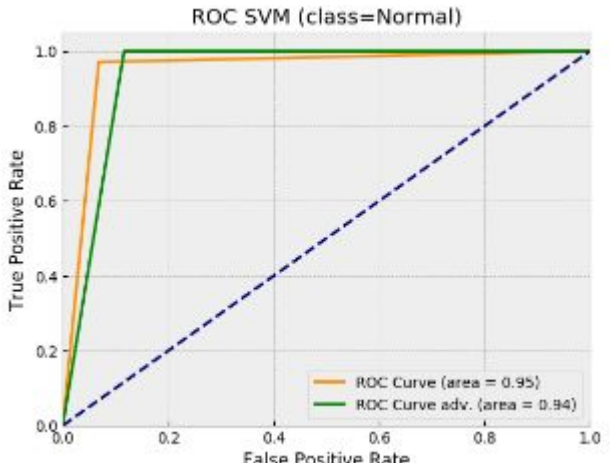
Model Evaluation on Adversarial Data

- This section presents the results of the baseline models on the adversarial test set generated by the JSMA method in terms of Accuracy, F1-score and AUC for the ROC curve.
- One thing to note here is that, both the AUC results as well as the ROC curves in the figures below, are only presented for the the “normal” class, while the F1-score is an average score over all classes.

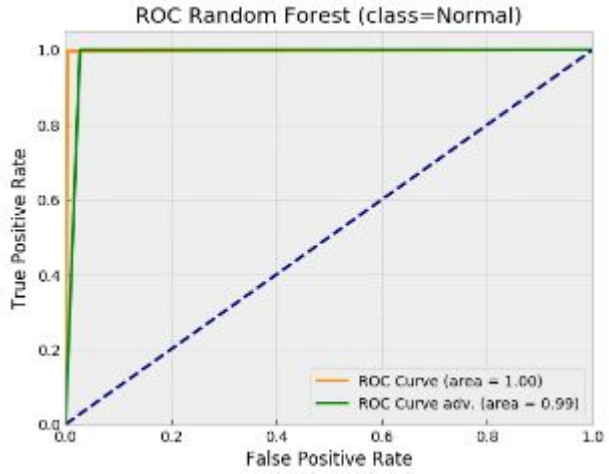
Method	Accuracy	F1-Score	AUC (normal)
Decision Tree	0.660	0.802	0.744
Random Forest	0.968	0.977	0.986
Linear SVM	0.810	0.846	0.949
Voting Ensemble	0.914	0.914	0.723
MLP	0.670	-	-



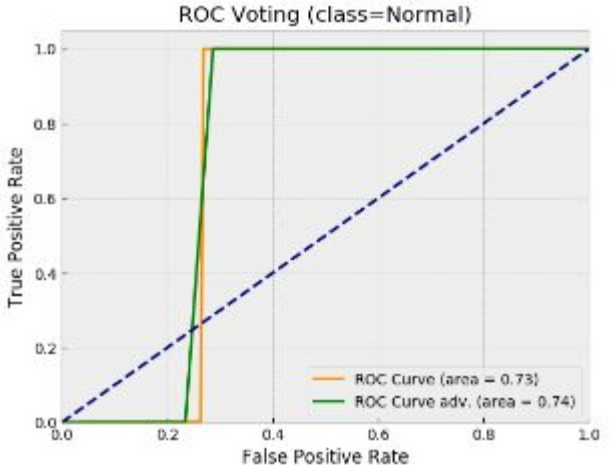
Decision Tree ROC curves



SVM ROC curves



Random Forest ROC curves



Voting Ensemble ROC curves

Feature Evaluation

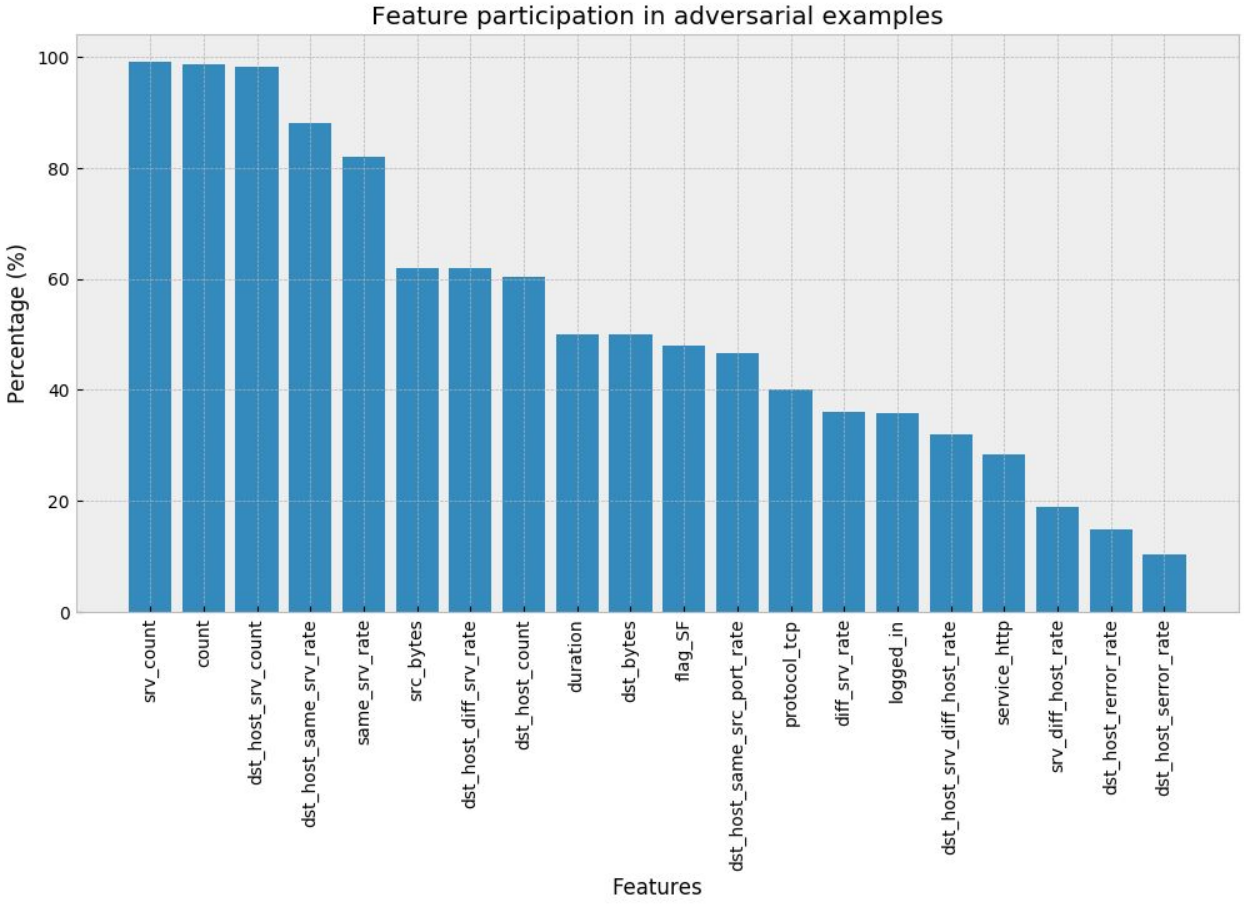
- After generating the adversarial test set using JSMA, a ranking of the features in terms of frequency with which they appear in the adversarial test set as changed was created. This was calculated by subtracting the original test set from the adversarial test set.

$$\delta = X^* - X_{\text{test}}$$

where X^* is the adversarial test set and X_{test} is the original test set. In order to find which features were altered for each data point $\delta^{(i)}$ we find the feature indexes j where feature $\delta_j^{(i)} = 0$

Feature	Description
srv_count	number of connections to the same service as the current connection in the past 100 connections
count	number of connections to the same host as the current connection in the past 100 connections
dst_host_srv_count	number of connections to the same service and destination host as the current connection in the past 100 connections
dst_host_same_srv_rate	% of connections to the same service and destination host
src_bytes	number of data bytes from source to destination
same_srv_rate	% of connections to the same service
dst_bytes	number of data bytes from destination to source
dst_host_diff_srv_rate	% of different services on the current host
dst_host_count	number of connections to the same destination host as the current connection in the past 100 connections
duration	duration of the connection

Top 10 adversarial features using JSMA



Most used features in adversarial sample generation

Results Analysis / Discussion

Data Modelling

- All the models have an overall accuracy and F1-score around 99%.
- The AUC scores however show a major difference where we observe that the Decision Tree and the Random Forest classifier outperform the SVM and the Voting ensemble.
- This implies that the first two methods are performing slightly better in classifying the “normal” test samples exhibiting a lower False Positive Rate (FPR).
- This can also be observed in the ROC-AUC curves shown earlier.

Adversarial Test Set Generation

- The FGSM method changes each feature very slightly while JSMA searches through the features of each data point and changes one at each iteration in order to produce an adversarial sample
- This means that FGSM is not suitable for tasks like NIDS since the features are generated from network traffic and controlling them in a fine grained manner would not be possible for an adversary.
- On the contrary, JSMA changes only a few features at a time and although it takes more time to generate adversarial examples as it is iterative, it can form the basis for a practical attack due to the lower number of features that have to be changed.

Model Evaluation on Adversarial Data

- In terms of overall classification accuracy all classifiers were affected.
- The most severely affected is the Decision Tree with a drop of 32.9% and the Linear SVM whose accuracy dropped by 13.5%. The Random Forest classifier showed some robustness with an accuracy drop of 2.5%.
- When it comes to F1-score, the Decision Tree was affected the most and its score was reduced by 19%. Linear SVM saw an F1-score reduction of 11.2%. The other two classifiers did not suffer as much and again the Random Forest showed the highest robustness by dropping only 1.7%.

Model Evaluation on Adversarial Data (contd.)

- The AUC over the normal class is an indicator of how robust were the classifiers against targeted misclassification towards the normal class, in other words, how many attacks were misclassified as normal traffic.
- The best performing classifier was the the Linear SVM, which only dropped 0.5%, followed by the Random Forest which dropped 0.8%. The Decision Tree classifier was severely affected, losing 24.8% percentage points.

Based on the results, it seems that the only method that was **robust across all metrics was the Random Forest**. The **Decision Tree was the worst** performing classifier, which also corresponds with the result of Papernot et. al. (although in the image classification domain), in which also, Decision Tree was one of the worst performing methods.

Feature Evaluation

The table of the top-10 features which contribute most during the generation of adversarial samples was shown earlier.

- Among those features, the top two are about the number of connections to the same service/host as the current connection in the past 100 connections.
- The next two features are about the rate and the count of the connections to the same host and port.
- This tells us that one way an attacker could get around the detection would be to lessen the number of requests they generate. Exploiting this can be very helpful for running bots as one can generate connections to external command and control servers and can hide their traffic under normal traffic that a user creates.

Feature Evaluation (contd.)

- A similar type of reasoning can also be applied to other count and rate types of features.
- This type of discussion is also relevant to the Denial of Service (DoS) type of attacks and while this dataset is quite old, historically, there have been attacks that followed the “low and slow” approach in order to appear as close to legitimate traffic as possible.
- When it comes to features related to service types, using common protocols like HTTP or HTTPS can be be a good strategy in order to hide into other normal traffic, instead of using protocols that might be easier to get discovered.

Limitations

Limitations

- In our study, we had a pre-processed dataset and not raw data which made it easier to attack.
- A physical attack would require some idea on how the raw network data are processed and the types of features that are generated.
- Unlike images, not all traffic related characteristics can be changed, even when an adversary has the ability to craft specific network packets and payloads.
- To protect against adversaries, NIDS classifiers will have to use features that can not be easily manipulated by an attacker.

Future Work

Future Work

- Although defenses have been proposed against these attack methods in several studies, these defenses do not generalize very well.
- A future study would be to examine some of these defenses and establish whether they improve the situation or not especially in the NIDS domain.
- In our study a neural net was used as the source model for preparing the adversarial examples. An extension of this study could be to use other models as the source as well.
- Further study is also required to understand the effect of the adversarial methods in different attack classes which would potentially yield a better overview of which features are more important for each attack type when it comes to adversarial sample generation.
- This can eventually be used by adversaries to select strategies that would allow them to hide their malicious traffic depending on the chosen attack.

THANK YOU!