Improving Robustness of a Network Intrusion Detection System in adversarial environments

Matthew Elliott and Sandra Scott-Hayward
Centre for Secure Information Technologies, Queen's University Belfast, Belfast, BT3 9DT, N. Ireland
Email: melliott21@qub.ac.uk, s.scott-hayward@qub.ac.uk

Abstract — As society and technology develop, more and more of our time is spent online, from shopping to socialising, working to banking. Ensuring our safety from malicious actors trying to capitalise on this digitisation is becoming ever more important. One such system that was developed to defend against attacks is a Network Intrusion Detection System (NIDS), a common tool used to detect intrusion attempts. Early adaptions used pre-configured signature detection to recognise attacks. Those early models evolved to use machine learning based anomaly detection to monitor real-time network activity and autonomously recognise intrusion attempts. Worryingly, the relatively new field of adversarial machine learning has been shown to be extremely effective in creating adversarial attacks that can easily bypass the NIDS. Adversary-aware feature selection, adversarial training and ensemble method were all used to increase the adversarial attack detection rate of the ML classifiers in the NIDS. Adversary-aware feature selection was the most effective, increasing the accuracy of three of the four classifiers from 0 for some adversarial attacks, to 0.98 for all adversarial attacks. In this work we build Hydra2, a tool to let users prototype an attack in a sand-box environment that has the ability of detecting adversarial attacks. The users can then quantify the results by adversarially attacking their prototype NIDS.

Key Words: Network Security, Network Intrusion Detection Systems, Adversarial Training, Adversary-Aware Feature Selection,, Ensemble Method.

I. Introduction

Modern Network Intrusion Detection Systems (NIDS) use machine learning (ML) models to detect anomalies in the traffic that they are analysing. However, these NIDS have shown to be vulnerable to adversarial attacks. These weaknesses in the network security field have been exposed by Hydra, a system developed with the purpose of simulating adversarial attacks against a NIDS [1]. Adversarial attacks can come in many forms with differing methodologies. However, they all have the same goal, to mislead a ML classifier. There are three main categories of adversarial attacks.

Overstimulation is a type of adversarial attack in which traffic that is likely to cause alert to a NIDS is generated and then sent to the target NIDS in such a large quantity as to overwhelm it.

A *poisoning* attack is executed during classifier training by inserting malicious traffic, classified as normal traffic, into the training data, with the aim of reducing the accuracy of target classifier.

Unlike a poisoning attack, an *evasion* attack targets the ML classifiers in a NIDS during their classification phase. Perturbations are made to network traffic in such a way as to mislead classification of the traffic in a ML classifier used in a NIDS.

For all adversarial attacks mentioned above, it is noteworthy that not all features can be altered to avoid detection. Only features that do not alter the network functionality and its ability to execute the intended attack should be considered for perturbation. These features will be referred to as non-compromising.

Aiken et al. [1] created *Neptune,* an anomaly-based NIDS that monitors network activity from flow statistics assembled from live network traffic. In its current form, it is trained to identify SYN flood attacks from benign traffic using either of the following classifiers: Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), and K-Nearest Neighbours (KNN). To assess the impact that adversarial evasion attacks would have on Neptune's ability to recognise a SYN flood attack, Aiken et al. [1] developed *Hydra*. Hydra is designed to perturb SYN flood traffic in such a way as to attempt to evade detection by Neptune. Hydra has no knowledge of Neptune, so the attack is a black-box attack. The perturbations made to the traffic are only achieved through using publicly available tools, hping3 [2] and nping [3], which would be available to a real-world malicious actor.

This paper will build a tool, Hydra2, that is built on top of Hydra, the work done by Aiken el al. [1]. Hydya2 will allow for the prototyping of a NIDS that is designed to detect adversarial evasion attacks. To achieve this, the following techniques will be employed: feature selection, adversarial training and ensemble methods. The following paragraphs give a concise summary of the techniques that are used, with an in-depth analysis provided in the literature review.

Feature selection: A network feature is a measurable and quantifiable attribute of a packet(s) that is read by the NIDS. Network traffic can contain a large number of features. The authors of [4] demonstrated that by reducing the number of features that are being analysed to only the most important should have positive effects for the NIDS attack classification in two ways. Firstly, the lighter workload of analysing fewer features will mean a more economical use of computing power, resulting in faster classification results. Secondly, and arguably more importantly, it also has a positive effect on classification accuracy.

Adversarial training: Adversarial ML techniques can be utilised to increase the accuracy of the ML classifier in an adversarial environment. One such adversarial ML technique developed by the authors of [5], the *min*-max method. Deployed to a NIDS, this method was able to reduce the adversarial evasion rate from 98% to 3%.

Ensemble: The myriad of ML classifiers available to Neptune all have inherently different strengths and weaknesses as can be seen from the range in results per classifier per attack in the results section of this paper. Applied to a NIDS, it is almost impossible to choose a classifier that is well rounded and performs well in every scenario. To address this variation in merits, an ensemble method can be used. An ensemble method is a set of classifiers which individually classify an object and together vote to decide on the final classification.

The remainder of this article will cover the following topics: II. Literature Review on the existing research, III. Introduction to Neptune, IV Introduction to Hydra, V Adversarial Attack Solutions, VI Results and Analysis, VII Recommendations, VIII Conclusions.

II. Literature review

Effectiveness of adversarial attacks against NIDS

When forming adversarial attacks against a NIDS, different papers assumed different threat models. A black-box attack, as defined by Darvish et al. [6], takes place when the adversary has no knowledge of the target's learning algorithm, model topology, model parameters and defense mechanisms. On the contrary, a white-box attack happens when the adversary had full access to that information. Alhajjar et al. [7] proposed the adversary targeting the NIDS in a white-box attack. However, all other papers reviewed treated the NIDS as a black-box. Apruzzese, Giovanni, et a [8] argues that the NIDS should not be treated as a white-box. In the real world, an adversary should not know the inner architecture of a NIDS. Thus in order for an attacker to attack a NIDS as a white-box, they would first likely have to target the NIDS as a black-box in order to gain information on it.

The authors of [9] identify that not all network features can be altered while still maintaining network functionality. A feature that is able to be changed whilst still maintaining the network functionality is known as a *non-compromising* feature. Thus, only the non-compromising features should be considered for perturbation. No literature in this review disagreed with this statement.

Apruzzese, et al. [8] highlights the vulnerability of ML models used in intrusion detection, malware analysis, and spam and phishing detection to adversarial attacks. Evasion and poisoning adversarial attacks were considered. RF, Multi-layer Perception and KNN models were considered for classifying the traffic. The adversarial attacks drastically decreased the recall of all three models.

The authors of [7] generated adversarial attacks by applying evolutionary computation (using a Particle Swarm Optimization and Genetic Algorithm) as well as deep learning using a Generative Adversarial Network (GAN) using the NSL-KDD [10] and UNSW-NB15 [11] datasets. The results showed SVM and Decision Tree classifiers being the most vulnerable to adversarial attacks with a greater than 90% evasion rate. The authors recommended not using these vulnerable models in a NIDS system.

To find the adversarial training technique most effective in creating traffic that can evade detection, [12] tested three different adversarial training techniques: attack based on training substitute model, Zeroth Order Optimization (ZOO) and on a GAN using the KDD-series dataset. The malicious traffic was used against a black-box NIDS using a Deep Neural Network (DNN) to classify traffic. Attacks based on the substitute model had the least negative impact on the DNN classifier. While ZOO achieved the greatest negative impact on the Deep Neural Network (DNN) model, it is very expensive and required a lot of queries to the NIDS. In the real world, the NIDS may limit the number of queries made, and thus reduce the effectiveness of producing adversarial attacks by not having enough data for ZOO to perform optimally. The attack based on GAN was able to reduce the accuracy of the target DNN from 0.890 to 0.567, however the training of the GAN proved to be unstable, suffering convergence failure.

Increasing robustness of a NIDS against adversarial attacks

Adversarial training of ML models: There are several different techniques for the adversarial training of an ML model suitable for use in a NIDS. The authors of [13] present a min-max method to train a NIDS against adversarial attack examples in the UNSW-NB dataset [14]. Using this min-max method, four different adversarial training methods were tested: 1 Randomize Rounding Approach (rFGSM), Deterministic Approach (dFGSM), Multi-Step Bit Gradient Ascent (BGA) and Bit Coordinate Ascent (BCA). BGA was able to lower the evasion rate the most from 100% to 21.8%.

Feature selection: When investigating the performance of a NIDS in an adversarial environment, the authors of [13] manually selected the 28 most relevant features of the data which were then normalised. They then used Principal Component Analysis (PCA) to select the features that accounted for 95% of the variation, reducing the adversarial evasion rate from 21.8% without PCA to 2.9% with PCA for a BGA attack. The other available attacks, dFGSM, rFGSM, BCA, exhibited a very similar behaviour as BGA.

However, Zhang, Fei, et al [15] highlight when a NIDS is in an adversarial environment, it is possible for feature selection to negatively impact the performance of the classifiers used by the NIDS. Adversary-aware feature selection is important as reducing the feature set may require the adversary to perturb less features in order to reach a comparable probability of evading detection.

Ensemble methods: An ensemble method allows for the protection of an adversarial attack targeting a specific classifier that exhibits a particular weakness against a specific configuration of evasion attack. In fact, it has been proved both theoretically by Biggio et al. [16] and experimentally by Perdisci at al. [17] that ensembles are more robust against adversarial evasion attacks. Dutta, Vibekananda, et al. [18] used a two-layer stacked ensemble method. The first layer used DNN and LSTM models for classification. The second layer was a metaclassifier based on logistic regression which determined the influence of each individual model on the ensemble method. This method outperformed state-of-the-art individual classifiers and meta-classifiers such as RF and SVM. An accuracy of 100% was achieved on the adversarial attacks based on the IoT-23 [19], LITNET-2020 [20], and NetML-2020 [21] datasets. Alternatively, the authors of [22] aimed to increase the robustness of the ensemble method by randomising what classifiers are used, ensuring the attacker cannot fine-tune the attack to the specific specification of the NIDS.

How the literature will be used to aid this work

In this work, we contribute to the domain of network security in adversarial environments by employing some of the individual listed techniques (exact specification in "adversarial attack solutions section) into Hyrda2. Adversarial-aware feature selection, adversarial training and ensemble techniques will able be available to be used individually or as a combination for the end user to prototype.

III. Introduction to Neptune

Neptune was developed as an anomaly-based NIDS for a Software Defined Network (SDN). The SDN is controlled by Faucet [23], which has flow rule implementations that allow for the forwarding of all traffic on all hosts to a specified mirror host. Flow stats are collected via listening to this dedicated mirror host. Argus [24], a layer 2+ network auditing tool is used to collect the flow statistics from the mirror host to then be used for training and classification purposes.

Table 1 lists the flow statistic features Neptune uses in order to classify network traffic.

Table 1 Flow statistics available to Neptune [1]

Feature Category	Feature
Packet header	eth_src,eth dst, ip proto, state flags
Stateful	pkt count, src pkts, dst pkts, bytes, src bytes, dst bytes, pkts per second, bytes per second, bytes per packet, packet pair ratio, pair flow

Neptune is primarily designed to detect SYN Floods. The SYN flood data that the classifiers are trained from comes from a combination of the DARPA SYN flood set [25] in addition to a custom generated SYN flood set with speeds ranging from 10 to 1,000,000 packets per second. The entire training dataset is roughly 60% SYN Flood, 40% benign traffic [1].

IV. Introduction to Hydra

Adversarial attack role

Hydra was developed as a tool to let users expose the vulnerabilities that NIDS classifiers have against adversarial evasion attacks. The goal of the attack produced by Hydra, the adversary, was based on the model proposed by the authors of [26]. The goal of the adversary was to construct a SYN flood attack with resulting flow statistics such that it would be classified as benign traffic by Neptune while still maintaining its network functionality and malicious nature. The adversary only had access to one host in the network. The SYN flood was a blind attack such that no knowledge of the configuration of the NIDS or the ML classifiers used was given [1].

Adversarial attack profile

The adversarial attacks from hydra were constructed in order to perturb three main categories of the flow statistics aggregated and read by Neptune. Payload size, packet rate and pairflow. As seen from the results of Table 6, the individual feature perturbation of only payload size or packet rate or pairflow had little effect on reducing the performance of the classifier. Again on Table 6, combining the individual perturbations proved much more successful in evading detection by the classifiers.

The baseline and perturbed feature values used by Neptune are defined in Table 2. The values of the perturbed features were calculated in such a way to cause the maximum rate of evasion across the ML classifiers that Neptune uses [1].

Table 2, Unaltered original perturbation test settings from [1].

	Payload (B)	Rate (pps)	Pairflow (pps)
Baseline Values	0	650	650
Perturbation Values	90	22	22
Payload+Pairflow	90	650	22
Rate+Pairflow	0	22	22
Payload+Rate+Pairflow	90	22	22

V. Adversarial Attack Solutions

The following section of the paper will give a detailed description as to how the new solutions that Neptune will use to detect adversarial attacks in Hydra2 will fit together. Feature selection, adversarial training and ensemble methods will be employed. Fig 1 shows the how the individual solutions in Neptune can be combined.

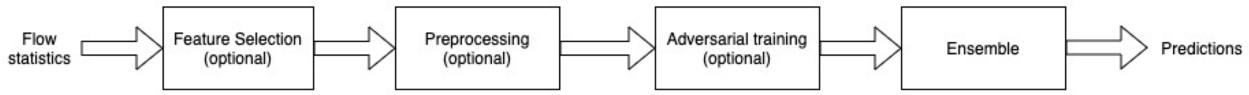

Fig 1, Architecture of applied adversarial attack solutions

Feature selection.

As mentioned in the literature review, Zhang, Fei, et al [15] highlights the importance of feature selection in adversarial environments, and the negative effect on classifier performance if the selection is not adversary-aware.

Table 3 shows which features of the network flow statistics will be perturbed for each attack. The column headers represent each adversarial evasion attack that Hydra, and now Hydra2, will execute against Neptune. From the literature review, we have concluded that it is only relevant to perturb non-compromising features. These non-compromising features represent the stateful features in Table 1. Perturbing any feature from the packet header category in Table 1 may affect the network functionality and thus will not be perturbed.

Of these stateful features available, some perturbations happen as a direct result of a perturbation. For example, increasing the amount of bi-directional traffic (pairflow feature) will result in a direct increase in the pairflow value of that flow statistic. This will be known as a primary perturbation and labelled as “1” in Table 3. Secondary perturbations happen indirectly as a result of primary perturbations. For example, increasing the number of bytes per packet (primary perturbation) will change the bytes per second (secondary feature). Secondary features are labelled as “2” in Table 3.

Table 3, primary perturbed features labelled as “1” and secondary features labelled as “2”

	Rate	Payload	Pairlfow	Rate+Payload	Rate+Pairflow	Payload+Pairflow	Payload+Rate+Pairflow
pkts							
src_pkts							
dst_pkts							
bytes		2		2		2	2
src_bytes		2		2		2	2
dst_bytes							
pkts_per_second	1			1	1		1
bytes_per_second	2	2		1	2	2	2
bytes_per_packet		1		1		1	1
packet_pair_ratio							
pair_flow			1		1	1	1

Hydra2 will have the ability to remove both the primary and secondary perturbed features as a way to assess the impact they have on the ability of Neptune to correctly classify the adversarial evasion attacks.

Adversarial Training

As a form of adversarial training, a proportion of the non-adversarial SYN flood flow statistics will be replaced by adversarial SYN flood samples. Either one of the following adversarial attack methods: Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM), Projected Gradient Decent (PGD) and HopSkipJump (HSJ), will be used to create adversarial samples against a target classifier. The reasoning behind this selection will be addressed below.

Hydra2 is designed to create attacks in such a way that the flow samples produced from that adversarial attack would evade detection by Neptune. In the same way that any ML model is subject to over-fitting, the classifiers in Neptune should not be trained on the features directly taken from each attack. Furthermore, in a real-world scenario, the NIDS should not have any knowledge of the attack before it is executed, so cannot fine-tune its training of its classifiers to detect this specific type and parameters of attack.

To address this problem, the adversarial trainer developed in Hydra2 will have no knowledge of the specific parameters of the attack, as defined in Table 2. This will eliminate the risk of overfitting the classifier(s) to the actual adversarial attacks that will be executed.

The adversarial trainer will be able to generate adversarial samples that target the same classifier weaknesses that the adversarial attacks executed by Hydra2 will attempt to leverage. This is done on a specific attack basis. This is achieved through the adversarial trainer only perturbing the features that the specific attack from Hydra2 will perturb.

Below is the pseudo-code of how this training is achieved.

```
1) Train the target classifier on non-perturbed attack and benign traffic.
2) Using the selected adversarial attack method, generate adversarial samples with the same feature
   perturbations as the attack that is being emulated.
     a. For the evasion attack that is to be emulated, identify all features (primary and secondary)
        defined in Table 3 that should be perturbed.
     b. Create a mask to define which features should be perturbed and which should not from
        the information in the sub-step before.
     c. Pass the mask to the adversarial attack method.
     d. Attack the trained target classifier, by selecting and then perturbing proportion of the
        SYN flood flow samples according to the mask.
     e. This attack will produce adversarial attacks that leverage weaknesses in the classifier that
        Hydra2 may use to evade detection in an adversarial attack.
3) Replace the existing SYN flood samples with their adversarial counterpart.
4) Re-train the ML classifier to recognise these adversarial samples in the data as SYN Flood
   attacks.
```

This retraining technique is similar to the technique used by the authors of [27]. They show that models that are vulnerable to adversarial attack via FGSM can be retrained using the FGSM adversarial examples to improve the model's robustness to these attacks.

Before going in-depth on the adversarial methods used, $J(\theta, x, y)$ is defined as the cost function of the model. x represents the feature values, y represents the labels and θ represents the parameters of the model.

Fast Gradient Sign Method

FGSM proposed by Goodfellow et al. [28] is able to efficiently change the predicted probability of the target classifier by perturbing the sample, x, in a one-step method by adding a small deviation in the direction of the gradient as defined in (1) below:

$$x^{adv} = x + \varepsilon sign(\nabla_x J(\theta, x, y) \tag{1}$$

Where ε is the magnitude of the perturbations.

Sriram et al. [29] showed that adding samples perturbed via FGSM can reduce the accuracy of a NIDS using an SVM classifier from 0.6651 to 0.6325 and LR classifier from 0.6332 to 0.6227. While this is a relatively small impact, the next two training methods build on this. The results of this adversarial attack method can be used as a baseline.

Basic Iterative Method

Kurakin et al. [30] further builds on the FGSM, producing the BIM. The FGSM is applied multiple times with a small step size and clipping the feature values after each iteration to ensure the values are within an ε-neighbourhood of the original data as defined in equation (2) below:

$$x_0^{adv} = x, \qquad x_{n+1}^{adv} = Clipx\{x_n^{adv} + \varepsilon sign \nabla_x J(\theta, x, y))\}$$
(2)

Guo, et al. [31] created a NIDS that individually used an SVM and KNN model to classify network flow statistics of benign and Denial of Service (DoS) traffic using the KDD99 dataset [32], which is very similar to the role of Neptune. The BIM attack samples reduce the recall of the DoS traffic using the SVM classifier from 0.912 to 0.1 and reduces the KNN classifier from 0.99 to 0.292.

Projected Gradient Descent

As with BIM, the PGD adversarial training method proposed by Madry et al. [33] iteratively takes advantage of FGSM in the projection area, *S*, a fixed set of possible perturbations, to generate the adversarial examples as defined in equation (3):

$$x^{adv} = \prod_{x+S} (x + (\varepsilon sign(\nabla_x J(\theta, x, y)))$$
(3)

Peng et al [34] set up a NIDS used to classify Normal, DoS, Probe, R2L and U2R traffic using individual classifiers including RF, SVM and LR, similar to Neptune. Adversarial samples of this network traffic were generated using the PGD technique. These adversarial samples reduced the classification accuracy of the RF classifier from 0.7688 to 0.3287, SVM from 0.7505 to 0.3278 and LR from 0.7508 to 0.4062. PGD was able to expose and exploit weaknesses all three of these classifiers.

HopSkipJump

The HSJ attack proposed by Chen et al. [35] is based on estimating the gradient direction using binary information at decision boundaries. Using this information, it generates adversarial samples against a trained model. Adversarial samples are produced exclusively by observing the information from the output labels generated by the target model.

Unlike the previous three adversarial methods, to our knowledge, HSJ has never been used in a NIDS environment. However, there are strong reasons to consider this is an appropriate adversarial training technique. Firstly, as specified in the adversarial training pseudo code above, the HSJ method allows for a mask to be applied to the flow samples that are being perturbed, which is essential in allowing the adversarial flow sample to maintain its theoretical attack functionality. Secondly, Chen et al. [35] compare the performance of HSJ to two other adversarial training techniques that are already being used in our research as specified above; FGSM and BIM. It has been shown that the HSJ technique requires significantly fewer queries than the previously mentioned techniques while remaining competitive in its abilities to mislead classifiers. Finally, all three previous techniques require the classifier to have a classifier loss gradient. Neptune has four available classifiers, of which only two have a classifier loss gradient: SVM and LR. The HSJ technique has no such requirement, which provides a platform for the adversarial training of all Neptune's classifiers as described in Table 4 below.

Table 4, classifier compatibility with adversarial training technique

Adversarial Training Technique	**Compatible Classifiers**
FGSM	SVM, LR
BIM	SVM, LR
PGD	SVM, LR
HSJ	RF, KNN, SVM, LR

Pre-processing

Neptune does not pre-processed its flow statistics before passing it to its classifiers. However, for the adversarial training to make logically sized perturbations across multiple features, the flow statistics will have to be pre-processed. The pre-processing techniques that are available to the end user of this system are: normalisation [36], standardisation [37], and robust scaling [38].

Multiple pre-processing techniques were made available to the user as each Neptune classifier preformed differently in terms of adversarial attack detection across the three pre-processing types available.

Ensemble

Multiple configurations of the ensemble have been made available to the end user as follows:

Hard voting

In the hard voting ensemble, each classifier has a single equally weighted vote and the majority decides the outcome.

Soft voting

In a more sophisticated version of hard voting, the soft voting ensemble is offered where each classifier determines the probability of each label. This is particularly advantageous when the classifiers do not have a high prediction probability for any single label as it avoids the classifiers rounding-up when not certain on a classification.The outcome will be the label with the largest cumulative probability.

Weighted voting

In addition to using either hard or soft voting, the importance of a classifier can be increased or decreased according to its configured weight. A classifier with a weight of 2 will have twice the influence over the final result as a classifier with a weight of 1.

VI. Results and Analysis

The following sections present the findings of the implemented adversarial training techniques proposed in the previous section. Looking at Fig 1 of Hydra2, it is possible to combine these adversarial training techniques together. However, in the interests of keeping this section concise, only the individual results of each technique will be explored. These results should act as a baseline to any user that wants to further combine the techniques when using the Hydra2 tool.

Repeating previous results from Neptune.

In order to build upon the existing work of Hydra by Aiken el al [1], the results were recreated and verified. Table 5 quantifies the performance of the classifiers on classifying adversarial traffic. The RF, KNN and LR classifiers largely agree with the findings of Aiken el al [1]. However, the SVM classifier has a much lower F1 score, which is due to the low TP rate which decreased from 0.971 as found by Aiken el al [1] to 0.8. This suggests than in Hydra2, the SVM classifier struggles to detect a SYN flood attack more than the original SVM model in the original Hydra.

The results for Tables 5, 9 and 10 were calculated using the following equations:

$$Acc = \frac{TP+TN}{TP+TN+FP+FN},\ \Pr = \frac{TP}{TP+FP},\ Re = \frac{TP}{TP+FN},\ F1 = 2 \cdot \frac{Pr \cdot Re}{\Pr + Re}$$

Where TP=True Positives, FP=False Positives, FN=False Negatives, Acc=Accuracy, Pr=Precision, Re=Recall and F1=F1 Score

However, the SVM classifier has a much lower F1 score, which is due to the low TP rate which decreased from 0.971 as found by Aiken el al [1] to 0.8. This suggests than in Hydra2, the SVM classifier struggles to detect a SYN flood attack more than the original SVM model in the original Hydra model.

Table 5, 5-fold cross-validated Neptune Classifier performance

Classifier	Accuracy	TP	TN	F1
RF	0.999	0.989	1.000	0.994
KNN	0.991	0.948	0.995	0.971
SVM	0.983	0.800	1.000	0.888
LR	0.999	0.994	1.000	0.997

In order to establish a baseline classifier performance, all classifiers were passed all adversarial evasion attacks. Each classifier did not use any technique described in the adversarial attack solutions section above. The results were recorded in Table 6. All but three of the values agree with the findings in the previous work. KNN accuracy reduces from 1 in the previous work to 0.61. Also, the SVM accuracy for Rate and Pairflow both reduced from 1 in the previous work to 0. It is Multiple iterations of this test produced the same results that differ from the original work. Therefore, these results in Table 6 will act as the baseline against which to measure the success of the techniques introduced in Hydra2.

Table 6: Baseline adversarial evasion attacks from Hydra2

	RF	KNN	SVM	LR
Rate	1.000	0.610	0.000	1.000
Payload	0.980	0.980	0.980	0.980
Pairflow	1.000	1.000	0.000	1.000
Rate+Payload	1.000	1.000	0.000	1.000
Payload+Pairflow	0.000	1.000	0.000	0.000
Rate+Pairflow	1.000	0.450	0.000	1.000
Payload+Rate+Pairflow	0.000	1.000	0.000	0.000

Feature selection

A SYN flood attack with no perturbations, *Standard*, was added to the total attacks set of attacks executed by Hydra2. This Standard attack will act as a non-adversarial baseline to ensure any technique deployed does not negatively affect the classifiers' performance in a non-adversarial environment.

For each adversarial attack, the features that would have got perturbed by the adversarial attack, as defined in Table 3, have been removed. The results of this feature selection against the corresponding adversarial attack are recorded in Table 7.

From Table 7, both the RF and LR classifies exhibit almost perfect accuracy on the adversarial samples. SVM was able to see an increase in its attack accuracy, improving the *Pairflow* and *Payload+Pairflow* from 0 to 0.945 and 0.99 respectively. However, KNN saw only a minor improvement to its *Rate+Pairflow* accuracy. The reason for not seeing any considerable increase in the performance of the KNN classifier is likely due to the fact that its performance was already very high across the majority of attacks.

Table 7: Accuracy of classifiers with adversarial features removed

	RF	**KNN**	**SVM**	**LR**
Standard	1.000	1.000	1.000	1.000
Rate	1.000	0.620	0.000	1.000
Payload	0.980	0.980	0.980	0.980
Pairflow	1.000	1.000	0.945	1.000
Rate+Payload	1.000	1.000	0.000	1.000
Payload+Pairflow	1.000	1.000	0.990	1.000
Rate+Pairflow	1.000	0.525	0.000	1.000
Payload+Rate+Pairflow	1.000	1.000	0.000	1.000

As seen in Table 7, feature removal was able to increase the attack accuracy for all classifiers to a minimum of 0.98 against the Payload+Pairflow attack.

While it is theoretically possible to improve the accuracy of a classifier by having knowledge of the adversary, as was required in Table 7, it can be argued that in the real-world example, it is impossible to predict the type of adversarial evasion attack you will be targeted by before the attack is executed.

To discover the consequences of preforming the feature removal on an attack different the current one being executed, a *Rate+Pairflow* attack was executed and the feature selection of a *Rate+Payload* attack was implemented. As can be seen from Fig 2, all classifier accuracies were reduced to less than 0.1. All classifiers preformed worse than their counterpart which did not implement any feature removal.

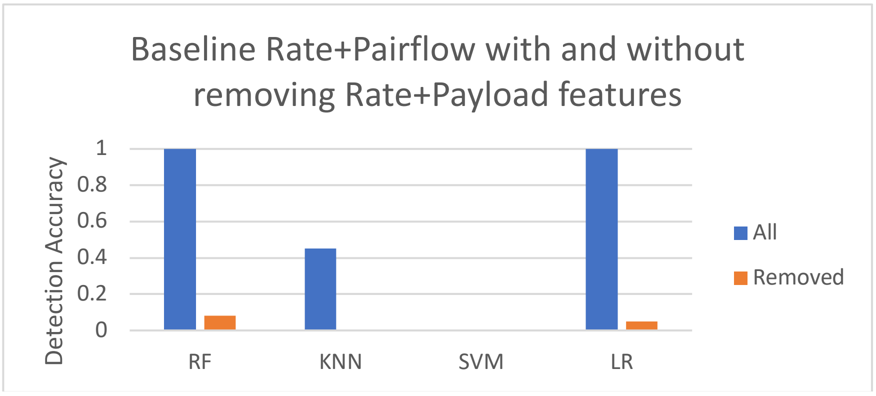

Fig 2, impact of removing the wrong adversarial features

Based on the results of Fig 2, it was concluded that for the accuracy of the classifier in adversarial settings, it is more important to remove the adversarial features than keep the non-adversarial features. A blanket feature selection that removed all possible features that could be perturbed by any attack was implemented. This would make it impossible for any of the perturbed features of any of the available adversarial attacks to have influence over the classifiers.

Table 8 records the results of this test. As can be seen, and backing up the previous conclusion, it is more important for the removal of adversarial features than it is for the inclusion of non-adversarial features.

Table 8, Attack detection accuracy with all possible adversarial features removed

	RF	KNN	SVM	LR
Standard	1.000	1.000	1.000	1.000
Rate	1.000	1.000	0.000	1.000
Payload	0.980	0.980	0.980	0.980
Pairflow	1.000	1.000	0.985	1.000
Rate+Payload	1.000	1.000	0.000	1.000
Payload+Pairflow	1.000	1.000	0.990	1.000
Rate+Pairflow	1.000	1.000	0.000	1.000
Payload+Rate+Pairflow	1.000	1.000	0.000	1.000

It was unclear as to the extent removing all possible adversarial features would have on the classifiers performance to classify non-adversarial traffic. As with Table 5, the 5-fold cross validated results were recorded for the classifiers but now from a reduced feature set. When comparing the original Table 5 results to Table 9, it can be seen that there is almost no effect on the performance of the classifiers under non-adversarial conditions when all features that could be possibly perturbed in an adversarial evasion attack are removed.

Table 9, Performance of classifiers assessed on 5-fold cv training data

	Accuracy	TP	TN	F1
RF	0.997	0.984	0.999	0.991
KNN	0.994	0.964	0.997	0.980
SVM	0.980	0.764	1.000	0.866
LR	0.997	0.984	0.998	0.991

Pre-processing

In order for the adversarial training to produce logical perturbed values, the data had to be pre-processed. There were three different types of pre-processing techniques that were employed (normalisation [36], standardisation [37], and robust scaling [38]), all of which produced varying results for each classifier.

When looking at the appendix, Table 14 recorded the baseline accuracy of all the classifiers under normalised data. Under normalised conditions, the SVM classifier is almost able to detect every intrusion on all attacks while maintaining an F1 score of 0.971, suggesting that is simply not classifying all traffic as an attack. While both RF and KNN reduced their detection accuracy of a subset of attacks, the LR classifier ends up classifying all traffic as background traffic.

Looking at the appendix, Table 15 records the results of the classifiers ability to detect adversarial evasion attacks under standardised conditions. RF experienced no change under standardised conditions.

KNN dropped slightly over a small number of attacks. As the same as normalisation, SVM increased its attack detection accuracy while LR had an accuracy of 0 across all adversarial attacks.

Table 10 records the ability of the classifiers ability to recognise adversarial attacks under robust conditions. Unlike the previous two pre-processing techniques, LR under normalised conditions does not give an adversarial evasion detection rate of 0. Furthermore, the results of the classifiers under robust conditions are closest to the baseline results of the classifiers under conditions with no pre-processing in Table 6.

Table 10, classifier performance under robust scaled conditions

	RF	**KNN**	**SVM**	**LR**
Standard	1.000	1.000	1.000	1.000
Rate	1.000	1.000	1.000	1.000
Payload	0.980	0.980	0.980	0.980
Pairflow	1.000	0.125	0.040	0.735
Rate+Payload	1.000	1.000	1.000	1.000
Payload+Pairflow	0.145	0.170	0.110	0.195
Rate+Pairflow	1.000	0.005	0.000	0.000
Payload+Rate+Pairflow	0.000	0.000	0.000	0.000
TP	0.989	0.964	0.846	0.969
TN	1.000	1.000	1.000	1.000
F1	0.994	0.981	0.916	0.984

Adversarial Training

As stared previously, robust scaled results produce the most appropriate baseline results for analysis of the adversarial training results. Only robust pre-processed adversarial trained results will be discussed will be compared to the baseline robust pre-processing results in Table 10. However, the results of the adversarial training for all other forms of pre-processing will be included in the appendix (Table 16 - 19) for the benefit of the end user of Hydra2 who may want to use different pre-processing techniques.

For all four available adversarial attack techniques used in the adversarial trainer, proportions of the total SYN Flood attack training data ranging from 0.2 to 1 of the total available were replaced with the generated adversarial samples. There appeared to be no correlation between the accuracy of the classifier in detecting the adversarial attacks and the proportion of SYN Flood samples in the training data that were adversarial perturbed.

It was noticed that some results obtained via adversarial training were unreliable. These will be highlighted in yellow in the results tables.

The adversarial trainer used FGSM to adversarially train the SVM and LR classifiers, of which the results are published in Table 11. Both SVM and LR managed to increase their accuracy on the *Payload+Pairflow* attack. SVM all other results for SVM had no significant change. LR saw a larger improvement than SVM, additionally improving on *Rate+Pairflow* and *Payload+Rate+Pairflow* while maintaining its other results.

BIM is able to slightly improve on FGSM, unreliably improving on its detection of *Rate+Pairflow*. On the contrary, BIM preforms worse when used in the adversarial trainer compared to FGSM. Arguably, LR while adversarially trained under BIM still observes a marginal benefit over no adversarial training. It is able to improve the accuracy of the *Rate+Payload, Payload+Pairflow, Rate+Pairflow and Payload+Rate+Pairflow* as the expense of *Rate* and *Payload*. In this case, it is clear that the iterative nature of BIM which is built on FGSM (as referenced before) has a marginal positive influence on the performance of the SVM classifier when used in the adversarial trainer. However, it has a negative effect on the LR classifier.

PGD and FGSM gave identical results for SVM. For LR, the only difference between the results of PGD and FGSM come from the *Rate* and *Rate+Payload* attacks. However, both results can be unreliable.

BIM did not improve on FGSM while PGD was only able to match the performance of FGSM. Due to the iterative nature of BIM and PGD, it could be argued that both of these adversarial methods are inferior to FGMG due to the increased computational workloads required and not improve on the accuracy of the classifiers.

Table 11, classifier performance under FGSM, BIM and PGD adversarial training with robust pre-processing, unreliable results highlighted in yellow

	FGSM		BIM		PGD	
Classifier	**SVM**	**LR**	**SVM**	**LR**	**SVM**	**LR**
Rate	1.000	1.000	1.000	0.000	1.000	0.890
Payload	0.980	0.980	0.980	0.000	0.980	0.980
Pairflow	0.040	1.000	0.040	1.000	0.040	1.000
Rate+Payload	0.000	0.890	0.000	1.000	0.000	1.000
Payload+Pairflow	0.955	1.000	0.995	1.000	0.995	1.000
Rate+Pairflow	0.000	1.000	0.580	1.000	0.000	1.000
Payload+Rate+Pairflow	0.000	1.000	0.000	1.000	0.000	1.000

The effectiveness of using the HSJ adversarial attack in the adversarial trainer was recorded in Table 12. The accuracy of both SVM and LR was reduced to 0 for the majority of adversarial attacks, far worse than even the robustly scaled baseline defined in Table 10. RF performed well under the HSJ training conditions, improving the two results that were below 0.98: *Payload+Pairflow* and *Payload+Rate+Pairflow*. KNN experienced an improvement similar to RF. apart from the *Payload+Rate+*Pairflow attack, it improved on every attack accuracy that was less than 0.98 (*Pairflow, Payload+Pairflow* and *Rate+Pairflow*).

Table 12, classifier performance under HSJ adversarial training with robust pre-processing, unreliable results highlighted in yellow

	RF	**KNN**	**SVM**	**LR**
Rate	1.000	1.000	1.000	0.000
Payload	0.980	0.980	0.980	0.980
Pairflow	1.000	1.000	0.000	0.000
Rate+Payload	1.000	1.000	0.000	0.000
Payload+Pairflow	1.000	0.755	0.000	0.700
Rate+Pairflow	1.000	1.000	0.000	0.000
Payload+Rate+Pairflow	0.560	0.000	0.000	0.000

Ensemble

Table 13, Ensemble results. Labelled as classifier(weight). No explicit weight assumes a weight of 1.

Ensemble configuration	RF, KNN, SVM, LR	RF, KNN, LR	RF, KNN(2), LR	RF, KNN(2), LR
Voting Type	hard	hard	hard	soft
Standard	1.000	1.000	1.000	1.000
Rate	0.970	1.000	1.000	1.000
Payload	0.980	0.980	0.980	0.980
Pairflow	1.000	1.000	1.000	1.000
Rate+Payload	1.000	1.000	1.000	1.000
Payload+Pairflow	0.000	0.000	0.620	0.980
Rate+Pairflow	0.980	1.000	0.970	0.850
Payload+Rate+Pairflow	0.000	0.000	0.940	1.000

Henceforth, the ensemble created in column 2 of Table 13 will be referred to as E1, column 3 as E2, column 4 as E3 and column 5 as E4.E1 simply combines all the classifiers into an evenly weighted hard voting classifier. Compared with the baseline results for KNN in Table 6, it can be seen that this particular ensemble performed worse than the individual classifier.

E2 improved on E1 by removing a poorly preforming classifier, SVM. While this is an improvement, looking at Fig 1, we can see that the ensemble now is just emulating the performance of RF and LR which have the same individual performance on each attack type.

E3 addressed the issue of the KNN classifier being ignored by giving it a weighting equal to the sum of the RF and LR classifiers. This ensemble now outperforms any individual classifier when considering the robustness to all possible attacks. The RF and LR are apple to compensate for the weaknesses in KNN and vice versa.

Finally, E4 uses a soft voting technique to enhance the voting ability of the ensemble, forming an almost perfect ensemble, apart from the Rate+Pairflow attack which has an accuracy of 0.85. Furthermore, this has been achieved without any computationally expensive pre-processing or adversarial training techniques.

This is not where the research into the ensemble should end. Referring to Fig 1, the construction of the ensemble is the final step in prototyping a defence against adversarial attacks using the Hydra2 tool. Any of the feature selection, pre-processing* and adversarial training techniques can be applied to each model before entering it into an ensemble.

VII. Recommendations

As presented in Table 6 of the results section, it is clear that adversarial evasion attacks are effective at misleading classifiers in the network security domain. This paper describes how it is possible to improve the detection of a ML based NIDS in an adversarial environment.

While it is shown in Table 8 that adversary-aware feature selection can improve a classifiers performance in detecting adversarial evasion attacks, it is unclear as to the extent of the effect this feature selection will have on: a) the ability of the NIDS to identify a more detailed set of categories of network traffic, not just SYN-flood and benign, and b) the effectiveness of this type of feature selection has on the ability of the NIDS to classify other attacks, both adversarial and non-adversarial. Other attacks, as well as more detailed network categories should be added to Hydra2 to explore this topic.

Lastly, the method of adversarial training used in this paper proved to be the least successful in improving the classifiers' ability to detect adversarial evasion attacks. One possible reason for this could be due to the adversarial trainer perturbing flow statistic features without context. For example, looking at the perturbed features of the *Rate+Payload* attack, the adversarial trainer will not recognise that the *pkts_per_second, bytes_per_second* and *bytes_per_packet* should all be perturbed in a proportional manor. The trainer will perturb the features without any consideration, likely leading to adversarial flow samples with illogical values that could never be achieved in an actual adversarial evasion attack.

VIII. Conclusions

Hydra2 was created as an educational tool, enabling the end-user to easily prototype a NIDS that should be able to detect adversarial evasion attacks. The user can then attack the NIDS with a range of adversarial evasion SYN Flood attacks and quantify the detection ability of the target NIDS that they just built.

Hydra2 has demonstrated that through adversarial-aware feature selection, it is possible to improve the accuracy of the NIDS. The feature results from Table 8 show that by removing all possible features that could be perturbed through an adversarial attack, the RF, KNN and LR classifiers individually can improve their ability to detect adversarial evasion attacks from 0 for some attacks, to at least 0.98 for all attacks. Furthermore, from Table 9, this feature selection has relatively little impact on the performance of the original classifiers' ability to classify between SYN Flood and benign network flow samples.

It has been demonstrated that it is possible to produce our own adversarial samples and to train on them to improve the ability of the classifier to detect previously unseen adversarial examples. FGSM proved to be the most effective and computationally in-expensive technique in improving the performance of the SVM and LR classifiers. HSJ, the only technique compatible with RF and KNN, had an overall net positive influence on RF but had no net improvement on the already accurate KNN classifier.

Of all the techniques employed, the adversarial trainer proved to be the least successful. It was by far the most computationally expensive technique employed but offered the overall least improvement to the classifiers. Often, trained against certain attacks, the accuracy of the classifier would decrease when compared to its non-adversarially trained counterpart.

Finally, It has been demonstrated that it is possible to configure an ensemble of classifiers that can outperform any single classifiers' ability to detect adversarial attacks in that ensemble. Thus, other classifiers in the ensemble are able to compensate for a lack of ability of any single classifier to a particular adversarial attack.

It is proposed that all industry exposed to technology should fully explore the techniques outlined in the paper, both in terms of absolute effectiveness as well as effectiveness in relation to computational power required to detect adversarial evasion attacks.

References

[1] J. a. S. S.-H. Aiken, "Investigating adversarial attacks against network intrusion detection systems in sdns.," 2019.

[2] S. Sanfilippo, "hping3(8) - Linux man page," [Online]. Available: https://linux.die.net/man/8/hping3. [Accessed May 2021].

[3] G. Lyon, "nping," [Online]. Available: https://nmap.org/nping/. [Accessed April 2021].

[4] M. A. T. a. A. B. E.-S. M. M. Sakr, "Filter versus wrapper feature selection for network intrusion detection system," 2019.

[5] M. O. S. a. A. M. R. Abou Khamis, "Investigating resistance of deep learning-based ids against adversaries using min-max optimization," IEEE, 2020.

[6] B. D. e. a. Rouani, "Safe machine learning and defeating adversarial attacks," IEEE, 2019.

[7] E. P. M. a. N. D. B. Alhajjar, "Adversarial Machine Learning in Network Intrusion Detection Systems," arXiv, 2020.

[8] M. C. L. F. a. M. M. G. Apruzzese, "Addressing adversarial attacks against security systems based on machine learning," IEEE, 2019.

[9] G. G. a. F. R. I. Corona, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," 2013.

[10] Canadian Institute for Cybersecurity, "NSL-KDD dataset," [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html. [Accessed April 2020].

[11] NourMoustafa, "UNSW_NB15 DATASET," IEEE, October 2019. [Online]. Available: https://ieee-dataport.org/documents/unswnb15-dataset. [Accessed April 2021].

[12] J. L. C. Z. a. Y. F. K. Yang, "Adversarial examples against the deep learning based network intrusion detection systems," IEEE, 2018.

[13] M. O. S. a. A. M. R. Abou Khamis, "Investigating resistance of deep learning-based ids against adversaries using min-max optimization," IEEE, 2020.

[14] N. a. J. S. Moustafa, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).," IEEE, 2015.

[15] F. e. a. Zhang, "Adversarial feature selection against evasion attacks," IEEE, 2015.

[16] B. G. F. a. F. R. Biggio, Multiple classifier systems for adversarial classification tasks, Springer, Berlin, Heidelberg, 2009.

[17] R. G. G. a. W. L. Perdisci, Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems, IEEE, 2006.

[18] M. C. M. P. a. R. K. V. Dutta, "A deep learning ensemble for network anomaly and cyber-attack detection," IEEE, 2020.

[19] Stratosphere Lab, "Aposemat IoT-23," [Online]. Available: https://www.stratosphereips.org/datasets-iot23. [Accessed May 2020].

[20] K. U. o. Technology, "LITNET-2020: an annotated real-world network flows dataset for network intrusion detection.," 2020. [Online]. Available: https://dataset.litnet.lt/. [Accessed May 2021].

[21] Advanced Computing & Networking Systems, " NetML-Competition2020," 2020. [Online]. Available: https://github.com/ACANETS/NetML-Competition2020. [Accessed May 2021].

[22] G. F. a. F. R. B. Biggio, "Adversarial pattern classification using multiple classifiers and randomisation," Springer, 2008, pp. 500-509.

[23] Faucet, "Faucet Open source SDN Controller for production networks," 2021. [Online]. Available: https://faucet.nz/.

[24] Argus, "Welcome to Argus," 2021. [Online]. Available: https://openargus.org/.

[25] C. S. U. M. Gharaibeh, "DARPA 2009 Intrusion Detection Dataset," 2009. [Online]. Available: http://www.darpa2009.netsec.colostate.edu/.

[26] B. e. a. Biggio, "Evasion attacks against machine learning at test time," Joint European conference on machine learning and knowledge discovery in databases. Springer, Berlin, Heidelberg, 2013.

[27] J. a. D. S. Kos, "Delving into adversarial attacks on deep policies," arXiv, 2017.

[28] I. J. a. S. J. a. S. C. Goodfellow, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.

[29] S. e. a. Sriram, "Towards evaluating the robustness of deep intrusion detection models in adversarial environment," Singapore, International Symposium on Security in Computing and Communication, 2019, pp. 127-132.

[30] A. I. G. a. S. B. Kurakin, "Adversarial examples in the physical world," arXiv, 2015.

[31] S. e. a. Guo, "A Black-Box Attack Method against Machine-Learning-Based Anomaly Network Flow Detection Models.," Security and Communication Networks 2021, 2021.

[32] Information and Computer Science University of California, October 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Accessed May 2021].

[33] A. e. a. Madry, "owards deep learning models resistant to adversarial attacks," arXiv preprint, 2017.

[34] Y. e. a. Peng, "Evaluating deep learning based network intrusion detection system in adversarial environment.," IEEE, 2019.

[35] J. M. I. J. a. M. J. W. Chen, "Hopskipjumpattack: A query-efficient decision-based attack," IEEE, 2020.

[36] scikit-learn, "sklearn.preprocessing.Normalizer," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html#sklearn.preprocessing.Normalizer. [Accessed 2020].

[37] scikit-learn, "sklearn.preprocessing.StandardScaler," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler. [Accessed 2020].

[38] scikit-learn, "sklearn.preprocessing.RobustScaler," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler. [Accessed 2020].

[39] F. e. a. Zhang, "Adversarial feature selection against evasion attacks," IEEE, 2015.

[40] scikit-learn, "sklearn.preprocessing.MinMaxScaler," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler. [Accessed 2020].

[41] I. e. a. Ahmad, "Security in software defined networks: A survey," IEEE, 2015.

Appendix

Table 14, classifier performance under normalised conditions

	RF	KNN	SVM	LR
Standard	1	1	1	0
Rate	1	1	1	0
Payload	0	0	0.98	0
Pairflow	1	1	1	0
Rate+Payload	0	0	1	0
Payload+Pairflow	0	0	1	0
Rate+Pairflow	1	1	1	0
Payload+Rate+Pairflow	0	0	1	0
TP	0.9948718	0.98974359	1	0
TN	0.99908257	1	0.94541284	1
F1	0.99697274	0.99484536	0.97194058	0

Table 15, classifier performance under standardised conditions

	RF	KNN	SVM	LR
Standard	1	1	1	1
Rate	1	1	1	0
Payload	0.98	0.98	0.98	0
Pairflow	1	1	0.86	0
Rate+Payload	1	1	0.02	0
Payload+Pairflow	0.145	0.905	0.92	0
Rate+Pairflow	1	1	0	0
Payload+Rate+Pairflow	0	0	0	0
TP	0.98974359	1	0.85128205	0.63076923
TN	1	1	0.99908257	0.99954128
F1	0.99484536	1	0.91921206	0.77336737

Table 16, classifier performance under FGSM, BIM and PGD adversarial training with normalisation pre-processing, unreliable results highlighted in yellow

	FGSM		BIM		PGD	
Classifier	**SVM**	**LR**	**SVM**	**LR**	**SVM**	**LR**
Rate	1.000	0.000	1.000	0.000	1.000	0.000
Payload	0.980	0.000	0.980	0.000	0.980	0.000
Pairflow	1.000	0.000	1.000	0.000	1.000	0.000
Rate+Payload	1.000	0.000	1.000	0.000	1.000	0.000
Payload+Pairflow	0.000	0.000	0.000	0.000	0.000	0.000
Rate+Pairflow	1.000	0.000	0.000	0.000	0.000	0.000
Payload+Rate+Pairflow	0.000	0.000	0.000	0.000	0.000	0.000

Table 17, classifier performance under HSJ adversarial training with normalisation pre-processing, unreliable results highlighted in yellow

	RF	**KNN**	**SVM**	**LR**
Rate	1.000	1.000	1.000	0.000
Payload	0.000	0.000	0.980	0.000
Pairflow	1.000	1.000	1.000	0.000
Rate+Payload	0.000	0.030	1.000	0.000
Payload+Pairflow	0.000	0.000	1.000	0.000
Rate+Pairflow	1.000	0.005	1.000	0.000
Payload+Rate+Pairflow	0.000	0.000	0.000	0.000

Table 18, classifier performance under FGSM, BIM and PGD adversarial training with standardisation pre-processing, unreliable results highlighted in yellow

	FGSM		BIM		PGD	
Classifier	**SVM**	**LR**	**SVM**	**LR**	**SVM**	**LR**
Rate	1.000	0.000	1.000	0.000	1.000	0.000
Payload	0.980	0.000	0.980	0.000	0.980	0.000
Pairflow	0.885	0.000	0.835	0.000	0.865	0.000
Rate+Payload	0.000	0.000	0.000	0.000	0.000	0.000
Payload+Pairflow	0.025	0.000	0.050	0.000	0.085	0.000
Rate+Pairflow	0.000	0.000	0.000	0.000	0.000	0.000
Payload+Rate+Pairflow	0.000	0.000	0.000	0.000	0.000	0.000

Table 19, classifier performance under HSJ adversarial training with standardisation pre-processing, unreliable results highlighted in yellow

	RF	KNN	SVM	LR
Rate	1.000	1	1.000	0.000
Payload	0.980	0.98	0.980	0.000
Pairflow	1.000	1	0.980	0.000
Rate+Payload	1.000	1	1.000	0.000
Payload+Pairflow	1.000	1	1.000	0.000
Rate+Pairflow	0.985	0.985	1.000	0.000
Payload+Rate+Pairflow	1.000	1	1.000	0.000