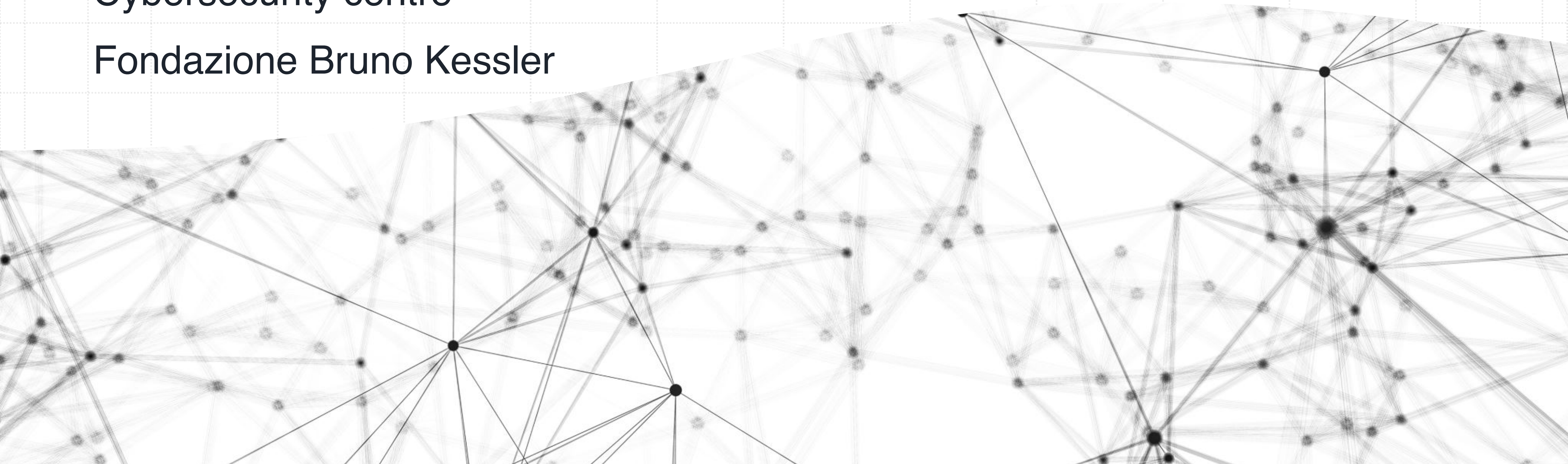
Robustness against Adversarial Machine Learning attacks

Roberto Doriguzzi Corin

Cybersecurity centre

Fondazione Bruno Kessler



Outline

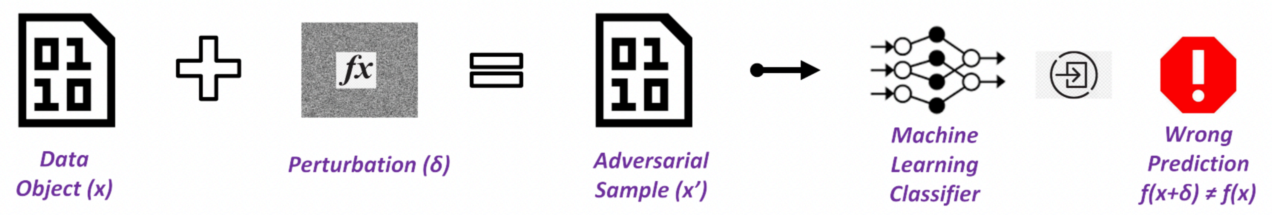
- Adversarial machine learning (AML)
- Generative Adversarial Networks
- Robustness against AML attacks with adversarial training

Adversarial Machine Learning (AML)

AML attacks are attacks against an Intrusion Detection System.

The two most common AML attacks are:

- **Poisoning:** crafted (or mislabelled) samples are inserted into the set of data employed for **training** the detection function of IDSs. The goal is to mislead the learning algorithm and negatively affect IDS performance in the classification phase (e.g., lower detection accuracy).
- **Evasion:** the intrusion traffic is carefully modified so that a trained IDS will not be able to detect it at **runtime** (i.e., no alert will be raised).



Methods to contrast evasion AML attacks

- **Adversarial training:** The first approach is to train the model to identify adversarial examples.
- **Switching models:** The second approach is to use multiple models within your system. The model used to make predictions is changed randomly. This creates a moving target as an attacker would not know which model is currently in use.
- **Generalised models:** A third approach is a generalised model defense which would also use multiple models. Instead of switching models, they are combined to create one generalized model. This means all the individual models would contribute to the final prediction. An adversarial example may be able to trick one model but it would likely not be effective against all of them.

Generative Adversarial Networks

Introduction to GANs

A **generative adversarial network** (**GAN**) is a class of machine learning frameworks designed by Ian Goodfellow et al. in 2014*

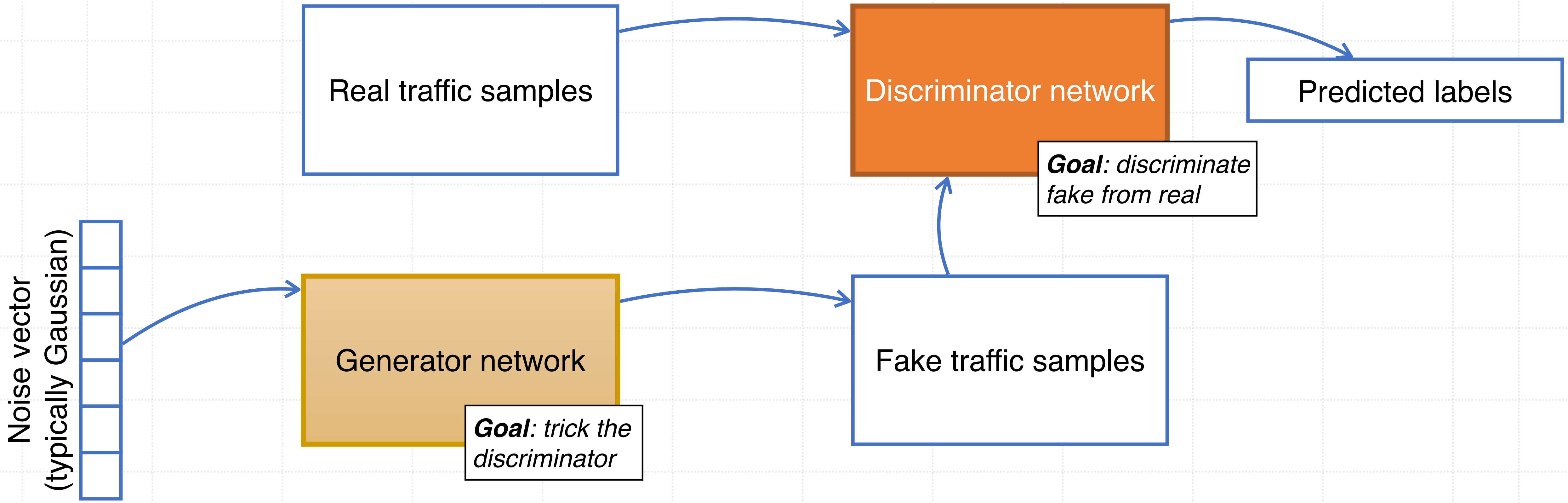
A GAN consists of two neural networks:

- The **Generator** takes random data as input and outputs some data (e.g., a fake traffic sample)
- The **Discriminator** takes a traffic sample as input (either fake or real) and guesses whether the input is fake or real

NOTE*: Generative adversarial networks are based on a* ***game theoretic scenario*** *in which the generator network must compete against an adversary. The generator network directly produces samples. Its adversary, the discriminator network, attempts to distinguish between samples drawn from the training data and samples drawn from the generator.*

*Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). *Generative Adversarial Nets*. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014).

GAN workflow


Real traffic samples
Discriminator network
Predicted labels
Goal: *discriminate fake from real*
Noise vector (typically Gaussian)
Generator network
Goal: *trick the discriminator*
Fake traffic samples

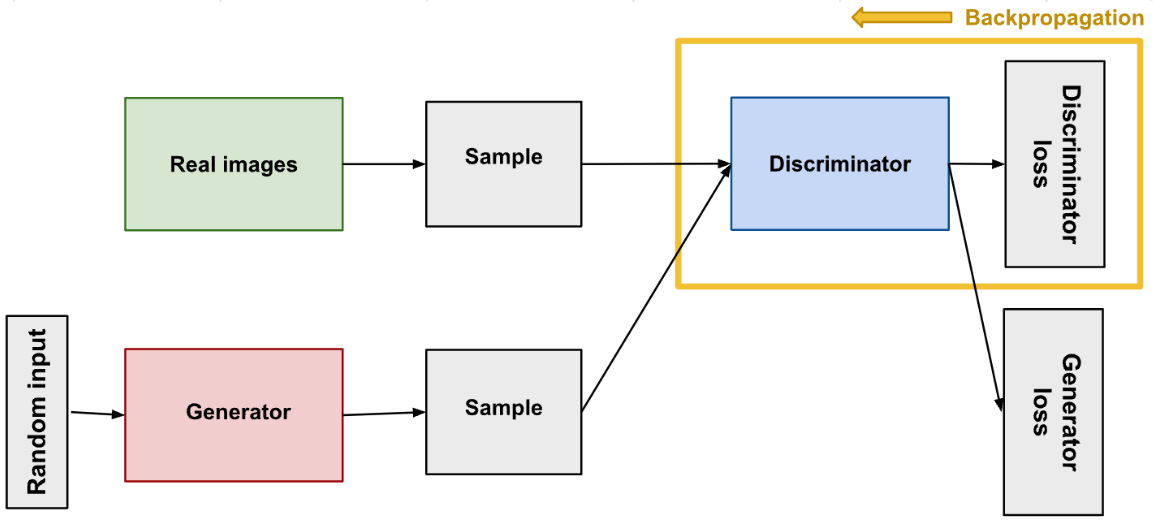
Training the discriminator

First phase: we train the discriminator using real and fake traffic samples. The latter are produced by the Generator using random numbers.

- The labels are 0 for fake samples and 1 for real samples
- Trained for one step using, e.g., binary cross-entropy

$$Loss = -\frac{1}{s}\sum_{j=1}^{s}(y^{[j]}\log\hat{y}^{[j]} + (1-y^{[j]})\log(1-\hat{y}^{[j]}))$$

- The weights of the Generator are frozen



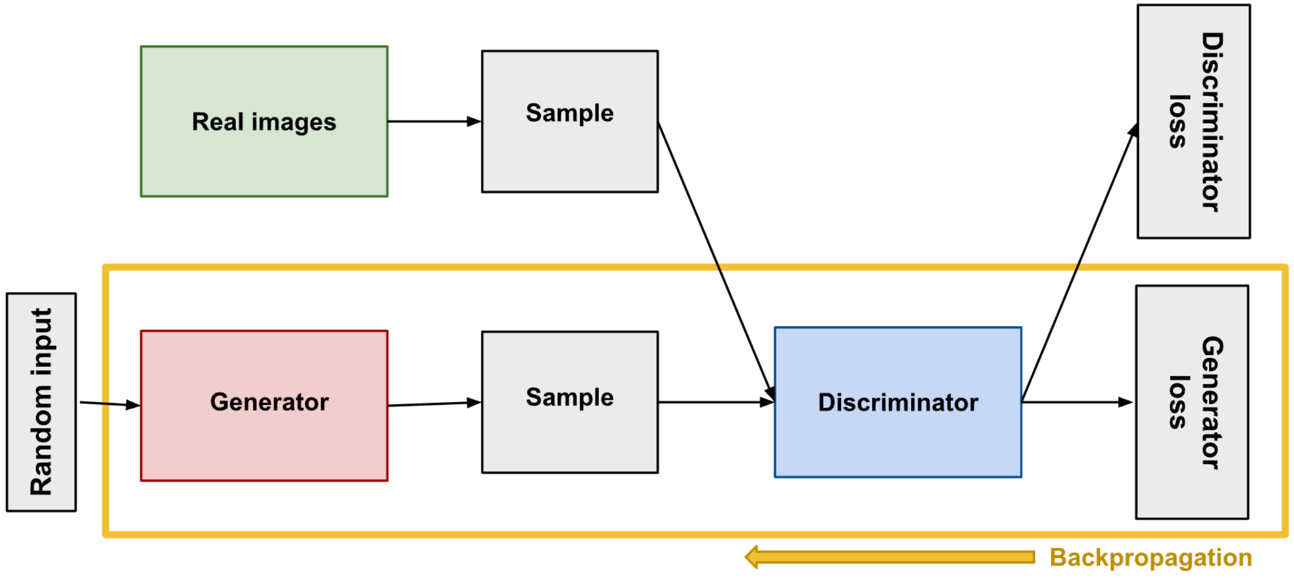
Backpropagation in discriminator training (source: https://developers.google.com/)

NOTE: The generator never sees real samples. However, it gradually learns to produce convincing fake samples exploiting the gradients flowing back through the discriminator.

Training the generator

Second phase: we train the generator only using fake traffic samples.

- Since we want the Generator to produce “good” fake samples, we label them as real (label 1) or, alternatively, we can use the negative of the discriminator’s loss function
- Backpropagate through both the discriminator and generator to obtain gradients.
- The weights of the Discriminator are frozen now, so the back-propagation will affect only the Generator
- The generator loss penalises the generator for failing to fool the discriminator



Backpropagation in generator training (source: https://developers.google.com/)

Training the GAN as a whole

GAN training is executed by alternating discriminator and generator training:

1. The discriminator is trained from one or more epochs
2. Then, the generator is trained for one or more epochs
3. Repeat steps 1 and 2 until the generator generates perfect fake samples and the discriminator's accuracy goes down to 50% (i.e., the discriminator is forced to guess)

Note 1: nothing guarantees that an equilibrium will be reached. For instance, if the GAN training continues after the discriminator starts providing random feedback, the generator may begin learning from flawed feedback and deteriorate in quality

Note 2: GANs are very sensitive to hyper-parameters and they often require a considerable effort in fine-tuning them

Why GANs in cybersecurity?

- One of the many major advancements in the use of deep learning methods in many domains is a technique called **data augmentation**.
- Data augmentation results in **better-performing models**, both increasing model skill and providing a regularizing effect, **reducing generalization error**. It works by creating new, artificial but plausible examples from the input problem domain on which the model is trained.

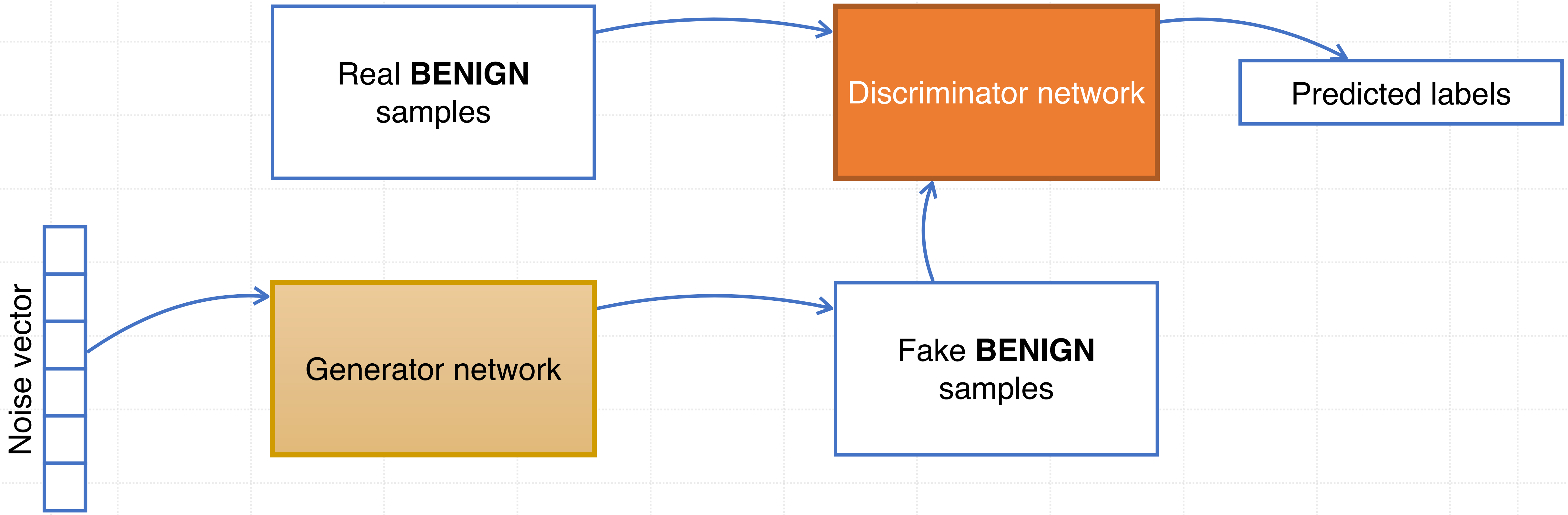
GADoT: GAN-based Adversarial Training for Robust DDoS Attack Detection

Abdelaty, M., Scott-Hayward, S., Doriguzzi-Corin, R., & Siracusa, D. (2021, October). Gadot: Gan-based adversarial training for robust ddos attack detection. In 2021 IEEE Conference on Communications and Network Security (CNS) (pp. 119-127). IEEE.

Approach

- **The goal** is to increase the robustness against the perturbed DDoS samples, in anticipation of an attacker who can craft packets and perturb **malicious flows to mimic those characteristics of benign flows**.
- **Process**: we use a GAN to generate fake benign samples in order **to perturb the features of the DDoS samples** with values seen in benign traffic
- **Data augmentation**: the ML model at the core of an IDS is trained with benign samples, DDoS samples, and adversarial DDoS samples.

Adversarial training with GaDoT



GAN-based adversarial training

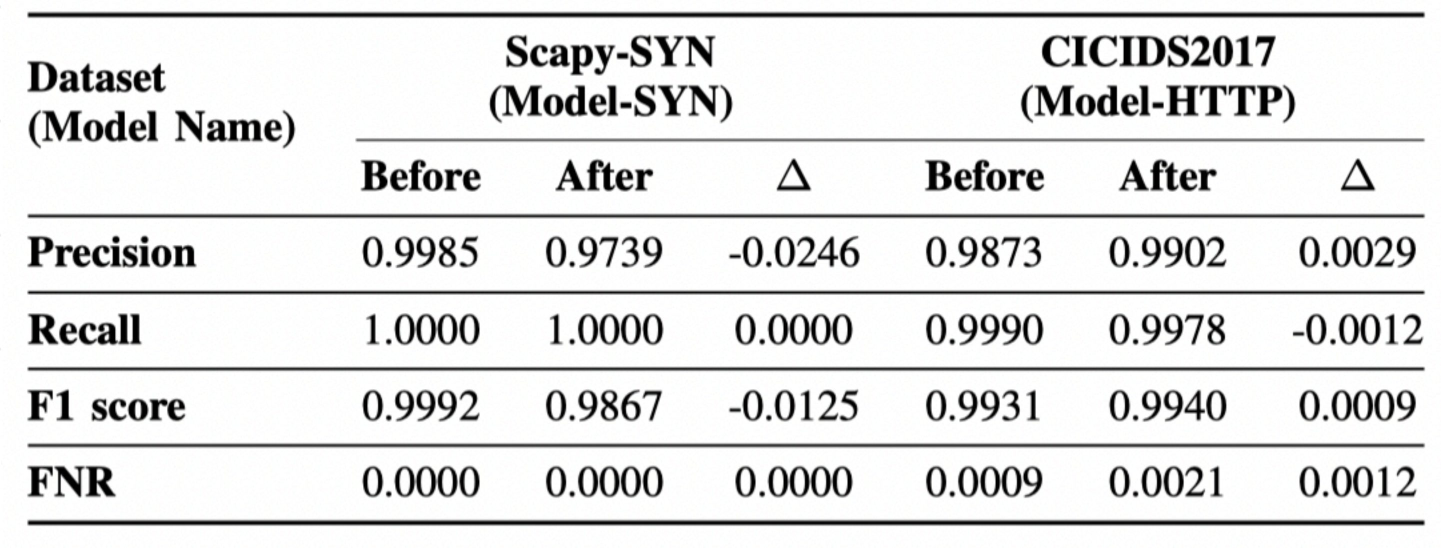
Idea: use a GAN to generate adversarial examples that can be used to train the IDS

- Once trained, the Generator is used to **produce fake-benign samples**
- Using these fake-benign samples, **we perturb the DDoS samples** by replacing their features with values from those fake samples
- The perturbed samples are added to the training dataset, which already contains benign samples and unperturbed DDoS samples, creating an **augmented dataset** for adversarial training

Evaluation on unperturbed data

Measure any negative effect of adversarial training with GADoT on LUCID when tested on **unperturbed data**.

Thus, we want to ensure that **adversarial training does not cause any degradation** to the performance of LUCID.



Dataset (Model Name)	**Scapy-SYN (Model-SYN)**			**CICIDS2017 (Model-HTTP)**		
	Before	**After**	**Δ**	**Before**	**After**	**Δ**
Precision	0.9985	0.9739	-0.0246	0.9873	0.9902	0.0029
Recall	1.0000	1.0000	0.0000	0.9990	0.9978	-0.0012
F1 score	0.9992	0.9867	-0.0125	0.9931	0.9940	0.0009
FNR	0.0000	0.0000	0.0000	0.0009	0.0021	0.0012

Perturbation of a SYN flood attack

Perturbations:

- **IP flags**: bit do not fragment randomly set to 0 or 1.
- **TCP length**: SYN packets with random payload content.
- **Padding replacement**: the SYN packet is followed by a random number of dummy packets with a random delay.
- **SYN Packet Replication**: the SYN packet is repeated a random number of times with a variable delay.
- **IP flags & TCP length & Padding replacement**: combination of these three perturbations.
- **IP flags & TCP length & SYN Packet Replication**: combination of these three perturbations.

Padding replacement and SYN Packet Replication, these are **perturbations to the Flow length feature** in which the attacker exploits the zero-padded rows in DDoS samples to fool the DDoS detection model.
The attacker's aim is to **force the feature extractor to fill them with a random number of additional packets**, making the DDoS samples less dissimilar to the benign samples (in terms of number of packets per flow).
This can easily be achieved by injecting dummy packets into the network with the same IP address and TCP port of the SYN packet, or by replicating the same SYN packet multiple times within a time window.

Perturbation of a SYN flood attack (results)

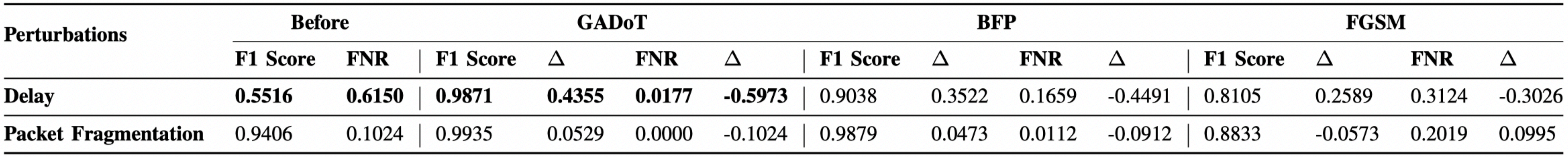
FNR measures the ability of the attacker to evade the IDS

Perturbations	**Before**		**GADoT**				**BFP**				**FGSM**			
	F1 Score	**FNR**	**F1 Score**	**Δ**	**FNR**	**Δ**	**F1 Score**	**Δ**	**FNR**	**Δ**	**F1 Score**	**Δ**	**FNR**	**Δ**
IP Flags	**0.6596**	**0.5071**	**0.9867**	**0.3271**	**0.0003**	**-0.5068**	0.6558	-0.0038	0.5071	0.0000	0.6541	-0.0055	0.5071	0.0000
TCP Len	0.9992	0.0000	0.9867	-0.0125	0.0000	0.0000	0.9953	-0.0039	0.0000	0.0000	0.9933	-0.0059	0.0000	0.0000
SYN Packet Replication	0.9547	0.0851	0.9867	0.0320	0.0000	-0.0851	0.9951	0.0404	0.0000	-0.0851	0.9933	0.0386	0.0000	-0.0851
Padding Replacement	**0.7875**	**0.3494**	**0.9867**	**0.1992**	**0.0000**	**-0.3494**	**0.9757**	**0.1882**	**0.0382**	**0.3112**	0.9127	0.1252	0.1491	-0.2003
IP Flags; TCP Len; SYN Packet Replication	**0.6322**	**0.5368**	**0.9846**	**0.3524**	**0.0041**	**-0.5327**	**0.9888**	**0.3566**	**0.0130**	**-0.5238**	0.9066	0.2744	0.1596	-0.3772
IP Flags; TCP Len; Padding Replacement	**0.5862**	**0.5843**	**0.9867**	**0.4005**	**0.0000**	**-0.5843**	**0.9731**	**0.3869**	**0.0431**	**-0.5412**	0.5999	0.0137	0.5657	-0.0186

Perturbation of a HTTP-based attack

Perturbations:

- **Packet fragmentation** perturbs the TCP Len feature of each packet and increases the number of packets in each flow as well
- **Random delay** between the DDoS packets and the fragmentation of those packets. The delay between packets is reflected in the packet arrival time, which is one of the features used by LUCID



Perturbations	Before		GADoT				BFP				FGSM			
	F1 Score	FNR	F1 Score	Δ	FNR	Δ	F1 Score	Δ	FNR	Δ	F1 Score	Δ	FNR	Δ
Delay	**0.5516**	**0.6150**	**0.9871**	**0.4355**	**0.0177**	**-0.5973**	0.9038	0.3522	0.1659	-0.4491	0.8105	0.2589	0.3124	-0.3026
Packet Fragmentation	0.9406	0.1024	0.9935	0.0529	0.0000	-0.1024	0.9879	0.0473	0.0112	-0.0912	0.8833	-0.0573	0.2019	0.0995

Conclusions

- The experimental results show that ML-based **IDSs are vulnerable** to AML attacks
- Potential adversaries can exploit their domain expertise to craft perturbed flood attacks **without requiring knowledge of the underlying detection model**
- Adversarial training is an effective solution to **increase the robustness of the IDSs** to AML attacks
- **GADoT achieve high performance** with F1 scores above 98% and FNR below 1.8% on perturbed DDoS network traces.
- The models trained using GADoT have detected the SYN and HTTP GET flood attacks, **whether they are perturbed or not**, allowing for the mitigation of such attacks