Black-box Adversarial Attacks Against Deep Learning Based Malware Binaries Detection with GAN

Junkun Yuan, Zhejiang University

Shaofang Zhou, Zhejiang University

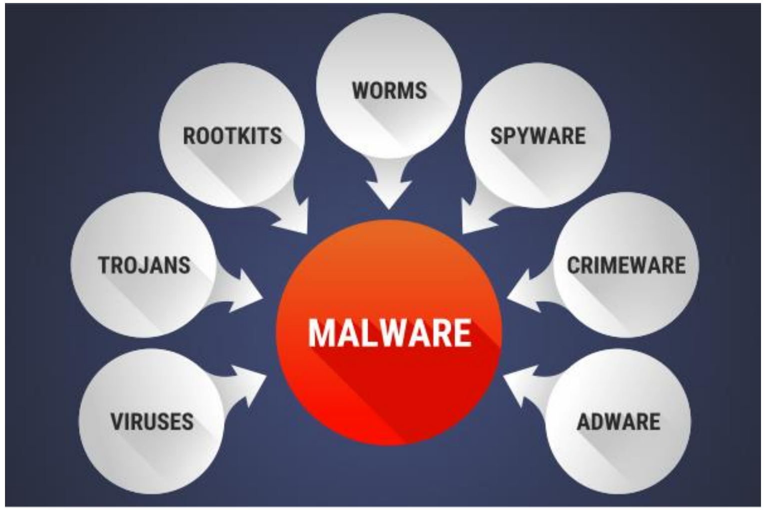
Lanfen Lin*, Zhejiang University

Feng Wang, Zhejiang Police College

Jia Cui, China Information Technology Security Evaluation Center

Background of Malware (malicious software) Detection

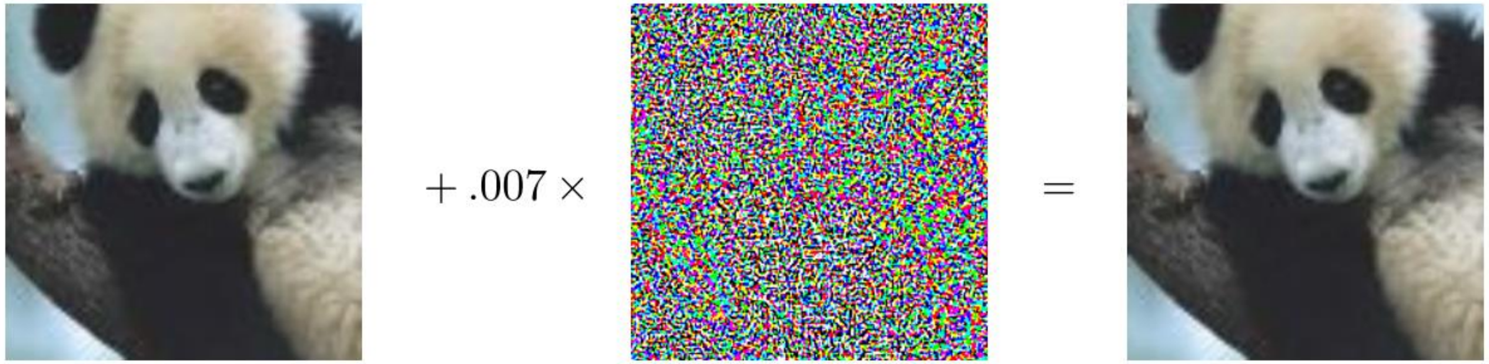
- Malware detection $\begin{cases} \text{manual features (e.g., API calls)} \\ \text{static features (part of the malware file)} \\ \text{both of them or else} \end{cases}$
- Deep neural networks have achieved great success.
- Recent trend: end-to-end detection with deep neural networks based on raw binaries



Background of Adversarial Attacks

- Adversarial samples: add small perturbations to original data that is imperceptible to humans but can mislead the classifiers.

- Adversarial attack studies point out a serious threat to the security of deep learning algorithm and AI applications, and is important for the studies of robust AI.



$\boldsymbol{x}$	$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$	$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"panda"	"nematode"	"gibbon"
57.7% confidence	8.2% confidence	99.3 % confidence

Background of Previous Attacks against Malware Detection

- White-box attacks: rely on the complete information (data, gradient, model, et al.) of the detector.
  - Deficiency: not applicable in real-world scenarios
- Manual feature based attacks: speculate and extract the features of malware used for training detection models.
  - Deficiency: need plenty of resources and time, and not useful for raw binaries based detection

Challenges of Byte-level Black-box Attacks against Malware Detection

- Challenge 1: Simple changes lead to functionality damage.
- Challenge 2: binaries data varies widely in size.
- Challenge 3: Subtle perturbations will be ignored when transforming between continuous and discrete space.

?

Introduction of Our Work

- We put forward a novel attack framework GAPGAN which Generates Adversarial Payloads via GANs.

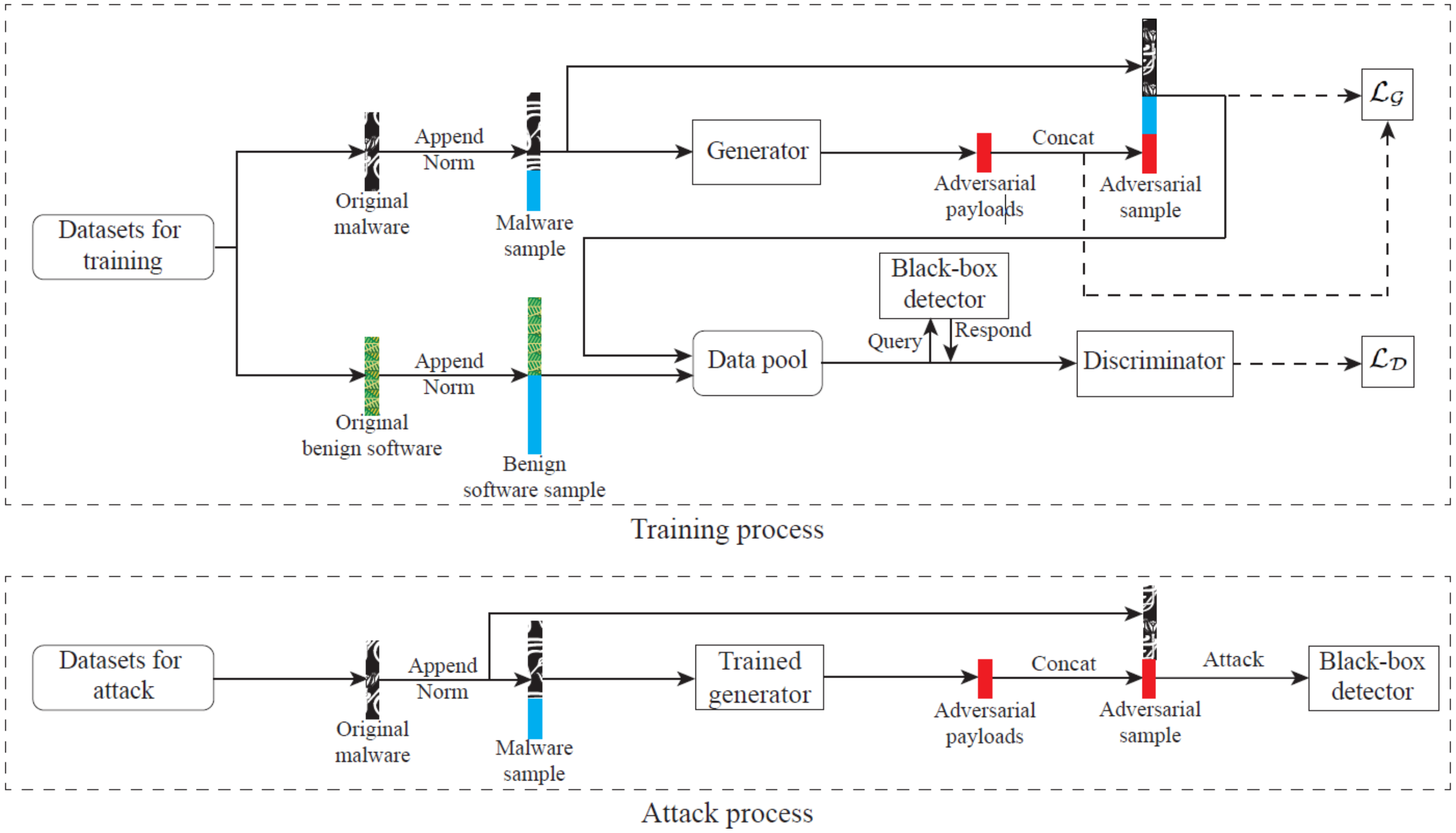
- [x] End-to-end black-box attacks
- [x] Byte-level attacks
- [x] Functionality preservation
- [x] Effective and efficient attacks

Problem Definition

- Binary file: $\boldsymbol{\mathcal{X}} = \{0, \dots, 255\}$ ➔ $\boldsymbol{b} = (b_1, \dots, b_n) \in \mathcal{X}^n$
- Benign software and malware: $\boldsymbol{b}_{ben}$, $\boldsymbol{b}_{mal}$
- Label of file b has label $y \in \{-1,1\}$, $y = \begin{cases} 1, & \text{benign software} \\ -1, & \text{malware} \end{cases}$
- The goal of malware detector f: $f(\boldsymbol{b}_{ben}) = 1$, $f(\boldsymbol{b}_{mal}) = -1$
- The goal of adversarial attack model g: $\boldsymbol{b}_{adv} = g(\boldsymbol{b}_{mal})$, $f(\boldsymbol{b}_{adv})$=1, while $\boldsymbol{b}_{adv}$ preserves the original function of $\boldsymbol{b}_{mal}$.

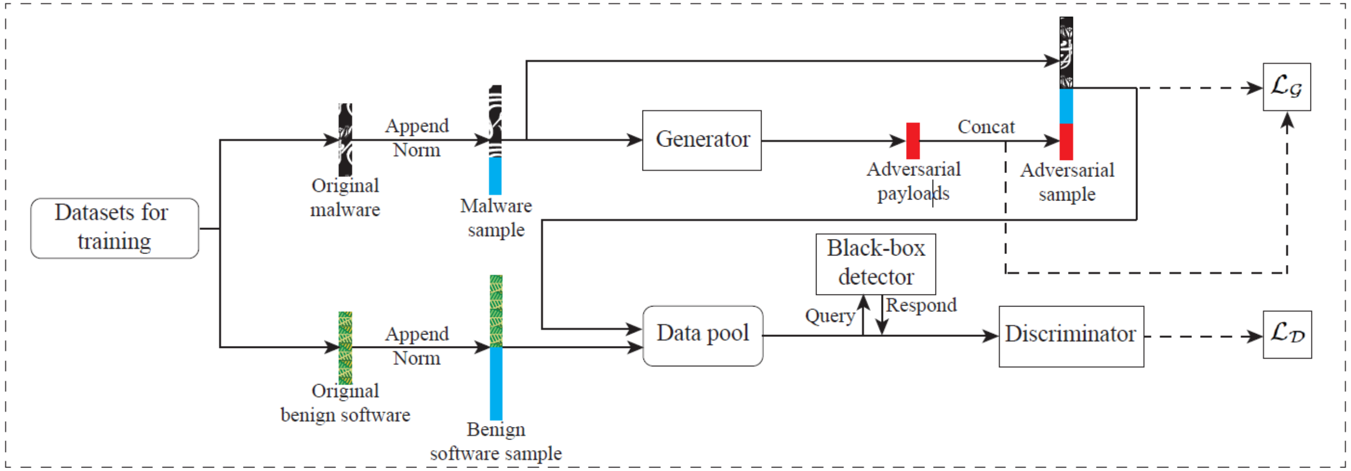
GAPGAN Framework

Training process & Attack process

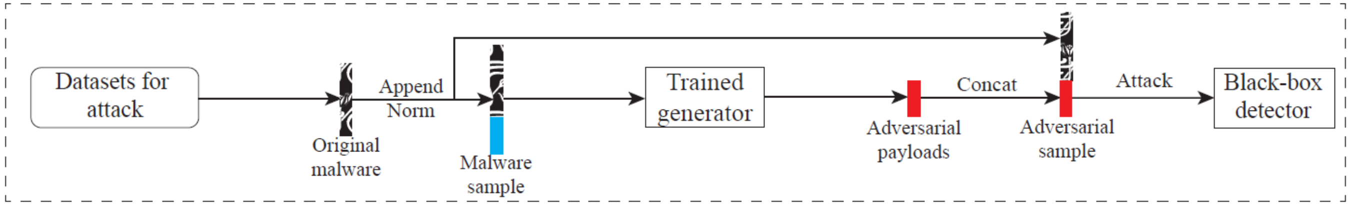


Training Process & Attack Process

- Training process
  - **Generator $\mathcal{G}$**: generate adversarial payloads and concatenate them to craft adversarial samples.
  - **Discriminator $\mathcal{D}$**: distill the target black-box detector f.
  - Train them concurrently.
- Attack process
  - Use the trained generator to attack.



Training process



Attack process

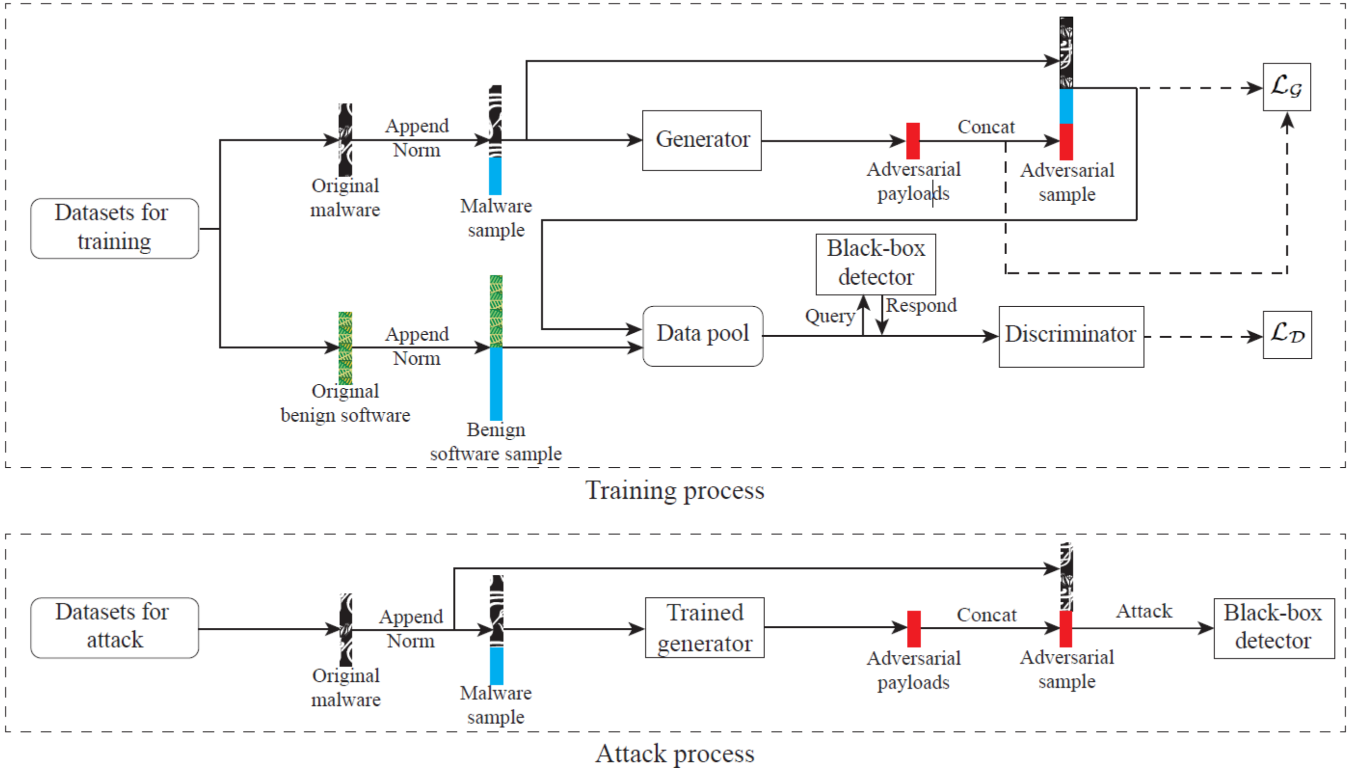
Generating Adversarial Sample & Functionality Preservation

- Append zeros (blue part in figure) to the end of input binaries to match the input size t of the network as $\boldsymbol{b}' = (b_1, \ldots, b_n, 0, \ldots, 0) \in \mathcal{X}^t$.
- Normalize to continuous space: $\boldsymbol{x} = (x_1, \ldots, x_t) \in \mathbb{R}^t$.
- Generate adversarial payloads:

$$\boldsymbol{a}_{adv} = \mathcal{G}(\boldsymbol{x}_{mal})$$

- Generate adversarial sample:

$$\boldsymbol{x}_{adv} = [\boldsymbol{x}_{mal}, \boldsymbol{a}_{adv}]$$



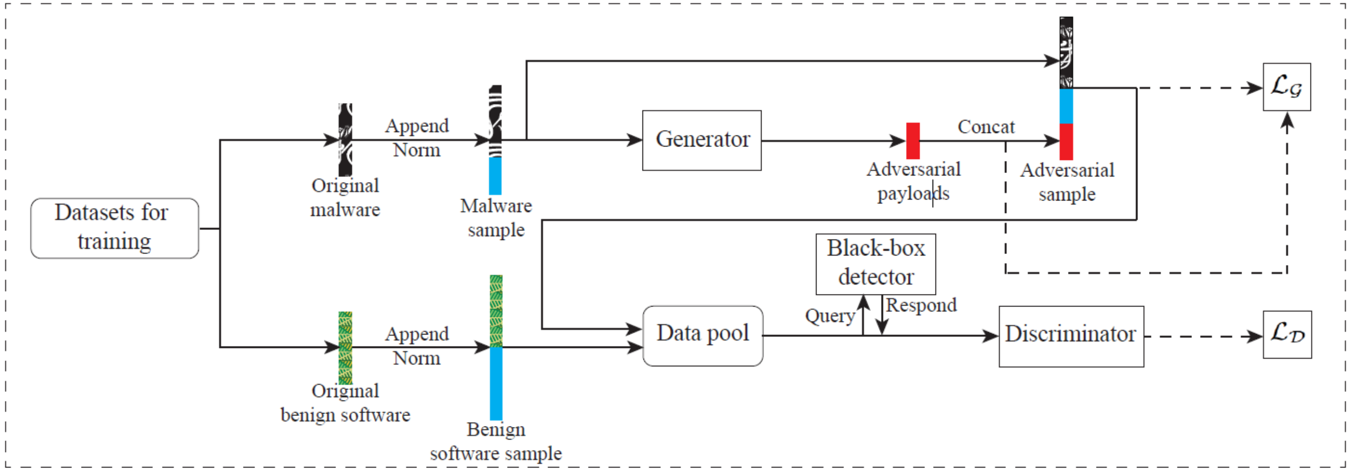
Generator $\mathcal{G}$

- Goal: learn characteristics of $\boldsymbol{x}_{mal}$ and generate corresponding effective sample $\boldsymbol{x}_{adv}$ that can mislead discriminator $\mathcal{D}$.
- Adversarial loss function:

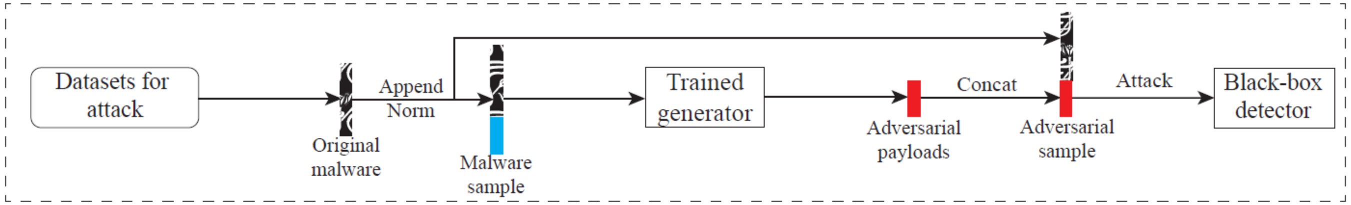
$$\mathcal{L}_{\mathcal{G}} = -(1-\beta)\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}_{adv}}}[\mathcal{D}(\boldsymbol{x})] - \beta\mathbb{E}_{\boldsymbol{a}\sim p_{\boldsymbol{a}_{adv}}}[\mathcal{D}(\boldsymbol{a})]$$

- Consider both the global and the local (i.e., $\boldsymbol{x}_{adv}$ and $\boldsymbol{a}_{adv}$) effectiveness with β:

$$\beta = \frac{\exp(\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}_{adv}}}[\mathcal{D}(\boldsymbol{x})])}{\exp\left(\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}_{adv}}}[\mathcal{D}(\boldsymbol{x})]\right)+\exp(\mathbb{E}_{\boldsymbol{a}\sim p_{\boldsymbol{a}_{adv}}}[\mathcal{D}(\boldsymbol{a})])}$$



Training process



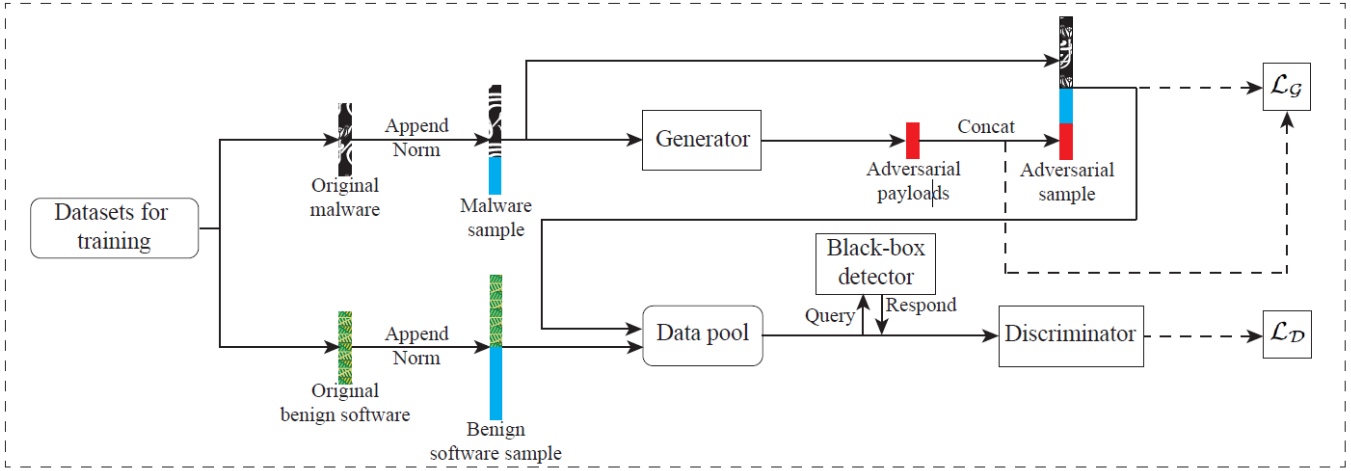
Attack process

Discriminator $\mathcal{D}$

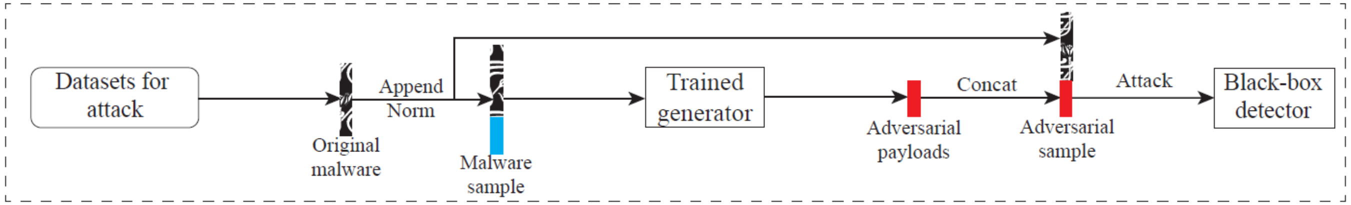
- Goal: Dynamically distill the target black-box model f.
- Distillation function:

$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{x}_{adv}} \mathcal{H}\big(\mathcal{D}(\boldsymbol{x}), f(\boldsymbol{x})\big) + \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{x}_{ben}} \mathcal{H}\big(\mathcal{D}(\boldsymbol{x}), f(\boldsymbol{x})\big)$$

- Sample a batch of mixed data and get labels by querying f, use them for fitting $\mathcal{D}$ with $\mathcal{H}$.
- $\mathcal{D}$ tries to learn the decision strategies of f on x_{ben} and x_{adv}.



Training process

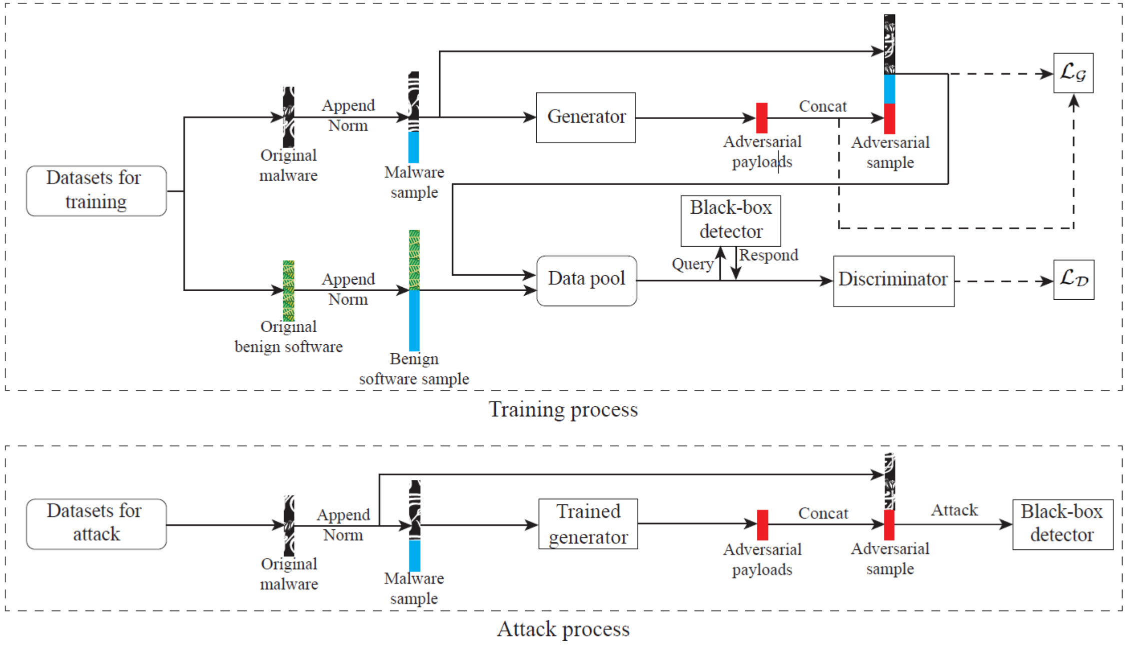


Attack process

Dynamic Threshold Strategy

- Challenge: subtle perturbations will be ignored when transforming between continuous and discrete space.
- Dynamic threshold strategy: directly set the bytes as zeros that below the dynamic threshold.
- $e = \begin{cases} e, if\ |e| > \epsilon * \dfrac{i}{T_{max}} \\ 0, \qquad\qquad\quad else \end{cases}$

e: byte in payloads
i: current training iteration time
T_{max}: maximum training iteration time
ϵ: maximum threshold value



Datasets

Datasets	Class	Number	Max	Mean	Source
1	Malware	3,436	93,986	51,715	VirusTotal
	Benign	3,436	98,304	41,651	Chocolatey
2	Malware	5,000	195,584	80,707	VirusTotal
	Benign	5,000	196,608	98,072	Chocolatey
3	Malware	10,000	394,128	126,276	VirusTotal
	Benign	10,000	393,640	128,808	Chocolatey
4	Malware	3,000	196,189	117,812	Kaggle 2015
	Benign	3,000	195,320	92,526	Chocolatey

- Malware: from VirusTotal and Microsoft Malware Classification Challenge (Kaggle 2015)
- Benign software: from Chocolatey Software
- 70% for training the black-box model, 30% for adversarial attacks.

Target Black-box Models

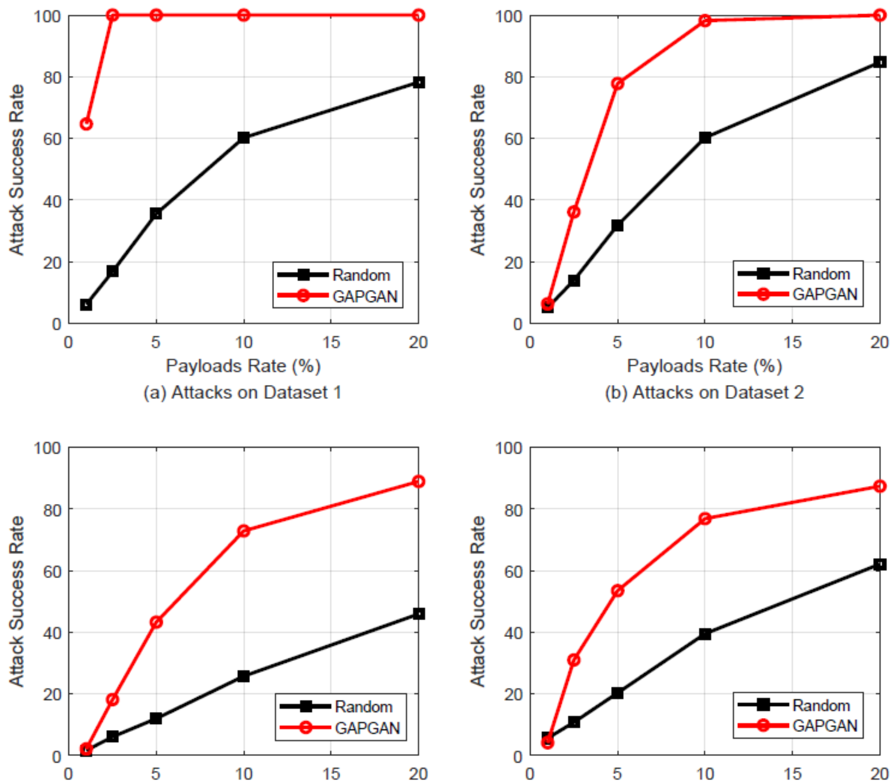
Datasets	MalConv	A	B	C	D
1	96.40%	-	-	-	-
2	96.42%	94.94%	95.99%	95.30%	94.70%
3	97.22%	-	-	-	-
4	95.55%	95.02%	95.27%	95.24%	95.30%

- A: CNN-based model
- B: CNN-LSTM-based model
- C: CNN-GRU-based model
- D: Parallel-CNN-based model

Attack Success Rate (ASR) of GAPGAN against MalConv

Payloads Rate	Dataset 1	Dataset 2	Dataset 3	Dataset 4
1%	64.66%	6.28%	2.15%	4.13%
2.5%	**100.00%**	36.10%	18.14%	30.99%
5%	**100.00%**	77.78%	43.27%	53.49%
10%	**100.00%**	**98.21%**	72.89%	76.88%
20%	**100.00%**	**100.00%**	**88.95%**	**87.41%**

- Payloads rate: the rate of the length of payloads to that of binaries for detection.
- ASR of adversarial samples can reach 100% with only 2.5% of the total length of the data for detection.



(a) Attacks on Dataset 1
(b) Attacks on Dataset 2
(c) Attacks on Dataset 3
(d) Attacks on Dataset 4

Attack Success Rate of GAPGAN and Others

	Adversarial attack methods			
	Opt. [14]	AdvSeq [23]	MalGAN [11]	GAPGAN
Black-box		✓	✓	✓
Run time	>2h	-	**0.02s**	**0.02s**
Attack level	**Bytes**	API calls	API calls	**Bytes**

Detector	Dataset 2			Dataset 4		
	Random	Opt.	GAPGAN	Random	Opt.	GAPGAN
MalConv	60.21%	**99.87%**	98.21%	57.52%	68.34%	**76.88%**
A	57.84%	**90.41%**	76.04%	17.10%	**85.09%**	51.31%
B	44.04%	93.32%	**99.35%**	46.50%	**77.24%**	68.67%
C	64.25%	**92.74%**	84.40%	55.72%	**78.17%**	64.96%
D	70.47%	97.23%	**99.93%**	9.03%	74.49%	**87.80%**

- Opt.: byte-level optimization based white-box attack method
- AdvSeq: API calls sequences based attack method
- MalGAN: API calls based attack method

Attacks under Defenses

Defense	Detector	Dataset 2			Datast 4		
		Random	Opt.	GAPGAN	Random	Opt.	GAPGAN
RND	Malconv	24.64%	51.23%	**63.69%**	49.59%	41.25%	**75.73%**
	A	20.67%	**57.84%**	45.00%	0.76%	**37.14%**	23.64%
	B	0.00%	62.29%	**87.47%**	5.79%	37.82%	**41.07%**
	C	7.65%	**39.47%**	34.57%	22.91%	29.74%	**39.22%**
	D	9.52%	43.58%	**92.35%**	3.06%	54.41%	**71.09%**
Adv.	Malconv	23.87%	29.78%	**57.04%**	13.10%	22.17%	**30.46 %**
	A	0.00%	15.14%	**23.72%**	0.00%	7.72%	**9.49%**
	B	0.00%	27.17%	**39.17%**	0.00%	9.38%	**15.82%**
	C	1.04%	19.77%	**24.18%**	4.99%	13.47%	**18.13%**
	D	0.00%	31.65%	**41.73%**	0.00%	17.97%	**27.60%**

- RND: random nullification data defense method
- Adv: adversarial training defense method

Thanks!