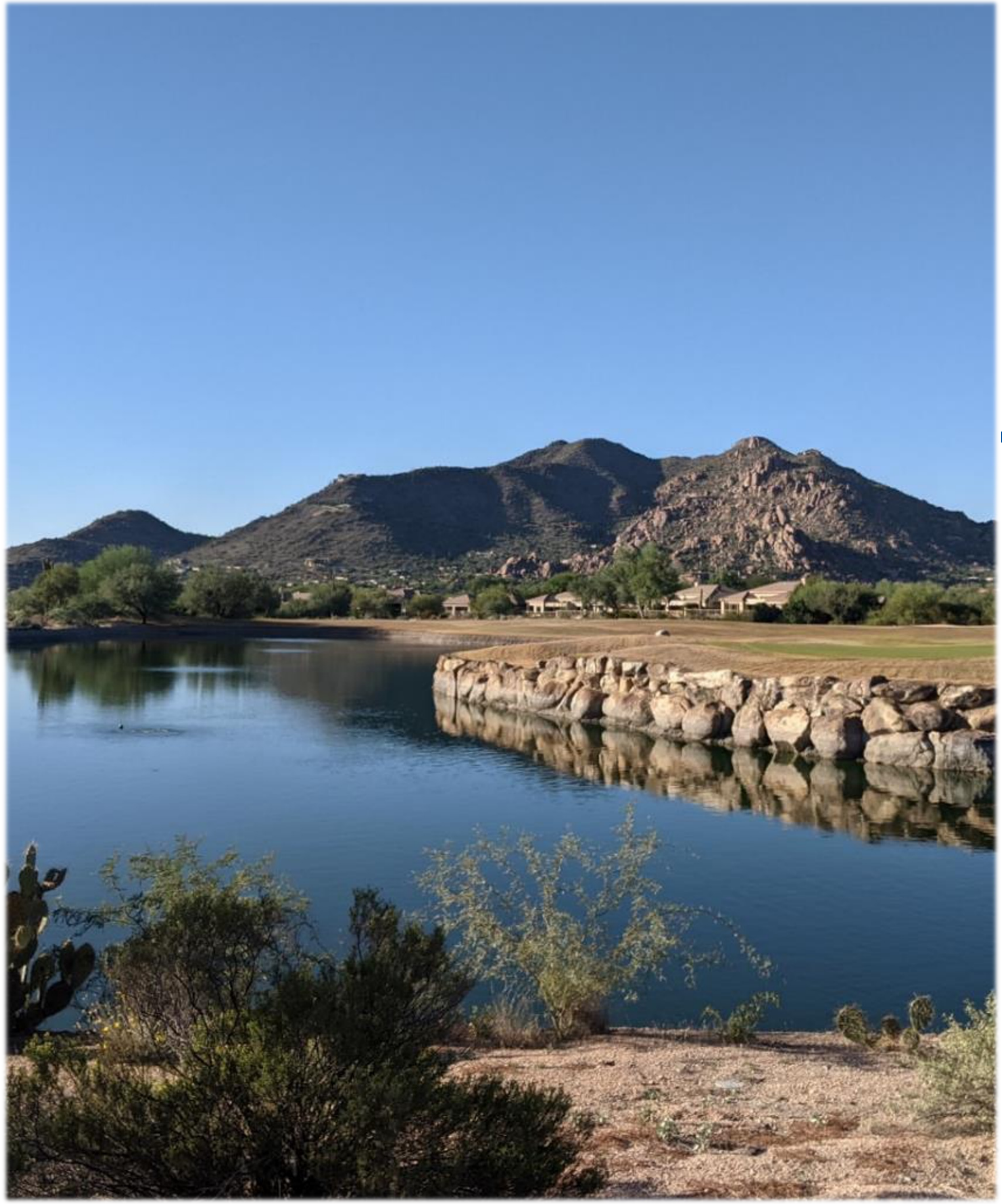


kipple: TOWARDS ACCESSIBLE, ROBUST MALWARE CLASSIFICATION

CAMLIS 2021

Andy Applebaum

@andyplayse4

https://github.com/aapplebaum/kipple

OUTLINE

- **About Me**
  - Professional: researcher at MITRE[1] (ATT&CK®, CALDERA™, security/AI)
  - Personal: chess National Master; 2018 DEF CON chess champion
- **This presentation**
  - Building a robust malware classifier
  - Making robustness more accessible for the community
  - Lessons learned for others trying to break into the field
- **Will be accompanied by:**
  - An open-source release (data, models, code/scripts)
  - A whitepaper
  - Long-form version of these slides
- **Disclaimer: fun side project outside my comfort zone**

1: The author's affiliation with The MITRE Corporation is provided for identification purposes only and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author.

Background: Static Malware Detection

- Look for known **indicators** in a file (md5, strings)
- Super quick, very reliable, low false positive
- **Struggles with new malware; high false negative**

- Find **similarities** between known bad files (ML!)
- **Can detect new malware with high(er) accuracy**
- Requires training data; can be slow; accuracy +/-

Background: Static Malware Detection

EMBER: gradient boosted decision tree, ~2500 features

A ROC curve of the resulting model is shown in Figure 5, and a distribution of scores for malicious and benign samples in the test set is shown in Figure 6. The ROC AUC exceeds 0.99911. A threshold of 0.871 on the model score results in less than 0.1% FP rate at a detection rate exceeding 92.99%. At less than 1% FP rate, the model exceeds 98.2% detection rate.

Anderson, Hyrum S., and Phil Roth. "Ember: an open dataset for training static pe malware machine learning models." arXiv preprint arXiv:1804.04637 (2018).

MalConv: Neural network, raw bytes -> learned features

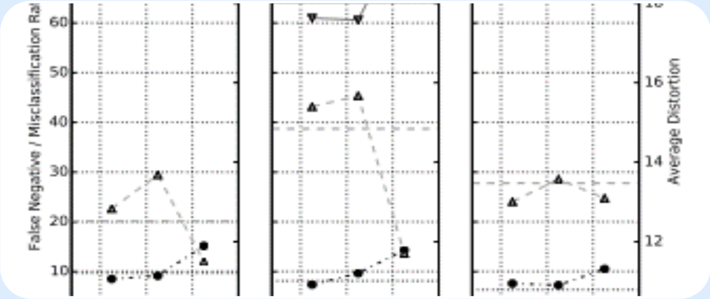
	MalConv		Byte n-grams	
Test Set	Accuracy	AUC	Accuracy	AUC
Group A	**94.0**	**98.1**	82.6	93.4
Group B	90.9	**98.2**	**91.6**	97.0

Raff, Edward, et al. "Malware detection by eating a whole exe." Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

Heuristic-based Detection

- Find **similarities** between known bad files (ML!)
- **Can detect new malware with high(er) accuracy**
- Requires training data; can be slow; accuracy +/-

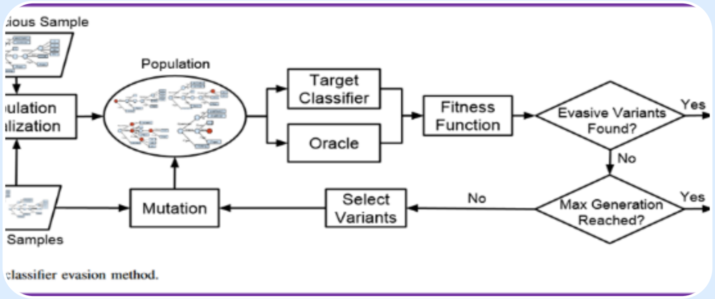
Evading ML-based Malware Detection with Adversarial Examples



Android Malware

- Perturb by adding declared features in Android manifest file

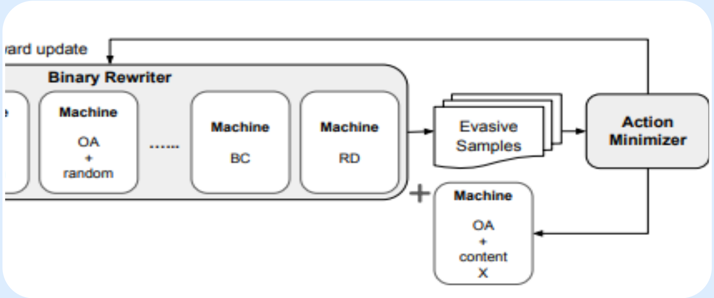
2016: "Adversarial Perturbations Against Deep Neural Networks for Malware Classification," Grosse et al.



PDF Malware

- Perturb by modifying PDF file, adding new features compliant with PDF spec

2016: "Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers," Xu et al.



Windows Executables

- Perturb by modifying PE file, preserving functionality

2020: "MAB-Malware: A Reinforcement Learning Framework for Attacking Static Malware Classifiers." Song, Wei, et al.

KIPPLE

How do we make malware detection more robust?

Motivation: 2021 Machine Learning Security Evasion Competition

- Public competition to build + attack malware classifiers
  - Put on by Microsoft, CUJO AI, NVIDIA, VMRay, and MRG Effitas
  - https://mlsec.io/
- Two tracks: attack and defend
  - Defend: submit a classifier able to detect malware (PE files)
    - **Must satisfy no more than 1% false positive rate**
    - **Must satisfy no more than 10% false negative rate**
  - Attack: make these 50 malware samples evade detection
- My goal: submit **something**
  - Doesn't have to be novel
  - Doesn't have to perform well
  - *Just needs to be in!*

https://mlsec.io/

Approach: Adversarial Retraining + Portfolio of Models

- Obtain a dataset of normal malware
- Using original malware, build a set of adversarial malware
- Train an initial model on only the original malware for baselining
- Train multiple models/portfolios using the original + adversarial malware
- Choose the option with best performance

Hypotheses – what do we hope to see?

Question 1

- Can we build a classifier that's robust to adversarial examples without sacrificing normal accuracy?

Question 2

- Is it better to use a single adversarially-retrained model or a portfolio of models?

Question 3

- When training on adversarial examples, is it better to train on *all* of them or only the evasive ones?

Question 4

- Is it worthwhile to write a classifier that discriminates between normal PE files (malware and benign) versus adversarially-generated ones?

OBTAINING DATA

- Binaries
- Feature vectors

Gathering Malware

- Started with EMBER data (feature vectors)
  - https://github.com/elastic/ember
  - **400K** malware; **200K** unknown; **400K** benign
- Obtained random malware from VirusShare
  - https://virusshare.com/
  - Rate limited so not a lot (**7662**)
- Obtained random malware from 2020 SoReL set
  - https://github.com/sophos-ai/SOREL-20M
  - Rate + hard drive space limited so only **~32K**
- Personal computer for benign binaries
  - **2525** in total, various PE files downloaded over 15yrs
- Obtained MLSEC
  - **150** “normal” malware samples (2019-2021)
  - **544** “adversarial” samples submitted in 2019

Source	Format	Label	Count
EMBER	Feature Vector	Malware	300000
VirusShare	Binary	Malware	7662
SoReL	Binary	Malware	31914
EMBER	Feature Vector	Unknown	200000
EMBER	Feature Vector	Benign	300000
Local	Binary	Benign	2191

Training Data

Source	Format	Label	Count
EMBER	Feature Vector	Malware	100000
MLSEC	Binary	Malware	150
EMBER	Feature Vector	Benign	100000
Local	Binary	Benign	379
MLSEC	Binary	Adversarial	544

Test Data

GENERATING ADVERSARIAL MALWARE

Three main approaches:

- **Functionality-preserving changes**
    - Malware RL (small changes)
    - SecML Malware (big changes)
- **New malware**
    - msfvenom

Training: Total Adversarial Malware Generated

Source	Generation Technique	Total
SoReL	MalwareRL	**37553**
SoReL	GAMMA	**5167**
SoReL	DOS Manipulation	**2590**
SoReL	Small Pad	**225**
SoReL	Large Pad	**277**
VirusShare	MalwareRL	**24581**
VirusShare	GAMMA	**5629**
VirusShare	DOS Manipulation	**2814**
VirusShare	Small Pad	**2347**
VirusShare	Large Pad	**2815**
msfvenom	No Added Code	**5884**
msfvenom	Added SoReL Malware	**33633**
msfvenom	Added VirusShare Malware	**7614**

131,169

45,812

38,186

47,171

Testing/Avoiding Duplication: MLSEC Adversarial Samples

- **MLSEC Included Samples ("*MLSEC 2019 Adversarial*")**
  - Attacker submissions from MLSEC 2019 – **544** in total
- **MLSEC Malware RL ("*MLSEC MRL*")**
  - Ran Malware RL on the 150 normal MLSEC malware samples to generate **1433** new instances
- **MLSEC SecML Malware ("*MLSEC SecML*")**
  - Ran SecML Malware to generate **746** new instances from the 150 MLSEC normal malware samples

Lessons Learned From Gathering + Generating Data

Have a lot of disk space

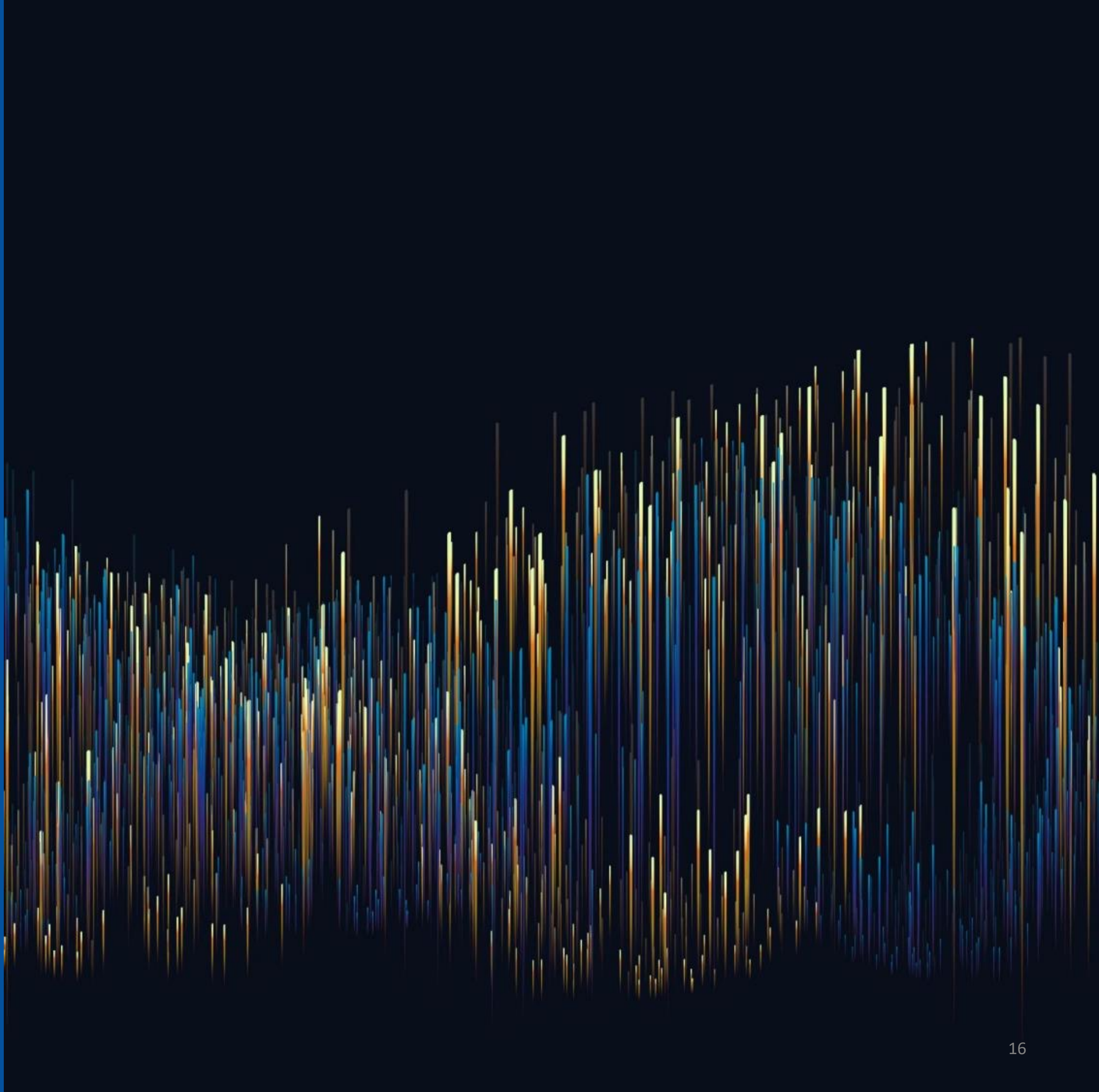
- Kipple was initially built on a small (<30GB) Linux VM on my personal PC
- Space became a deciding factor to download models, features, samples
- Space became a deciding factor when *generating* new samples
- **Eventually resized VM to 300GB – but would be easier to start here!**

Dedicate enough time

- When downloading samples: often those downloads are rate-limited
- When generating new samples: generation can be extremely time consuming
- Processes were run overnight, with multiple instances at a time
- **A cloud deployment would've saved time + helped space issues**

THE INITIAL MODEL

Pretty basic EMBER



Building the Initial Model

- Follow EMBER model training code
- Use gradient boosted decision tree
- Train only on EMBER train data
- Find threshold to set FP rate to 1%

- Performs well on benign (EMBER, local)
- Performs well on EMBER, VirusShare, MLSEC
- Only 90.3% accuracy for SoReL malware

Source	Label	Accuracy
EMBER Test	Benign	99.0%
Local Test	Benign	97.6%
EMBER Test	Malicious	96.5%
VirusShare	Malicious	99.9%
SoReL	Malicious	90.3%
MLSEC	Malicious	99.3%

Evaluating the Initial Model: Adversarial Malware

- Struggles with MLSEC 2019 adversarial data and SoReL MalwareRL + GAMMA samples
- Can detect padding + DOS manipulation
- VirusShare variants looking easier to detect
  - Likely due to data leaks – VirusShare samples pulled from original EMBER training data
- msfvenom lowest accuracy
  - As expected, adding code made it easier to detect
  - VirusShare surprisingly not easier

Source	Generation Technique	Accuracy
MLSEC 2019	-	**53.8%**
SoReL	MalwareRL	**58.9%**
SoReL	GAMMA	**59.6%**
SoReL	DOS Manipulation	89.2%
SoReL	Small Pad	95.1%
SoReL	Large Pad	93.9%
VirusShare	MalwareRL	**83.4%**
VirusShare	GAMMA	**80.8%**
VirusShare	DOS Manipulation	99.6%
VirusShare	Small Pad	99.6%
VirusShare	Large Pad	99.6%
msfvenom	No Added Code	**10.9%**
msfvenom	Added SoReL	**22.7%**
msfvenom	Added VirusShare	**24.3%**

Lessons Learned From Generating an Initial Model

Keep good records

- Embarrassingly, we lost the model parameters used for the initial model!
- Likely followed EMBER source, but remained an issue throughout development

Separate training and testing data

- VirusShare variants proved to be derived from our training data
- **Make sure you track *where* your data is coming from**
- **Make sure to generate test data from a different source as your train data**

ADVANCED MODELS

- Retraining

Building and Testing a Retrained Model

- Retrain model with new adversarial samples
  - Score original EMBER benignware as benign
  - Score original EMBER malware as malware
  - Score new adversarial variants as malware
  - Discard EMBER unclassified instances
- Select a threshold that ensures 1% FP rate
- Does pretty well on all categories
  - Not perfect on everything: but an improvement

Source	Label	Accuracy
Local Test	Benign	78.0%
EMBER Test	Malicious	94.4%
MLSEC	Malicious	96.7%
MLSEC 2019	Adversarial	76.7%
MLSEC MRL	Adversarial	84.0%
MLSEC SecML	Adversarial	86.6%

Initial Model vs. Retrained Model

Source	Label	Accuracy
Local Test	Benign	97.6%
EMBER Test	Malicious	96.5%
MLSEC	Malicious	99.3%
MLSEC 2019	Adversarial	53.9%
MLSEC MRL	Adversarial	56.6%
MLSEC SecML	Adversarial	76.4%

Initial Model

Source	Label	Accuracy
Local Test	Benign	78.0%
EMBER Test	Malicious	94.4%
MLSEC	Malicious	96.7%
MLSEC 2019	Adversarial	76.7%
MLSEC MRL	Adversarial	84.0%
MLSEC SecML	Adversarial	86.6%

Retrained Model

ADVANCED MODELS

- Building a portfolio

Portfolio Options

- Idea: combine multiple models each focused on classifying the *adversarial* malware
- Two primary paradigms, both treating *only* the adversarial samples as malware
  - **All**. Here, all EMBER data (malware **and** unknowns) is treated as benign (i.e.: *normal PE* vs. *adversarial*)
  - **Benign**. Here, only benign EMBER data is considered as benign; malware and unknown discarded
- Four model variations for which adversarial samples to include:
  - **Adversarial**. Includes all adversarial malware instances
  - **Variants**. Includes only MalwareRL and SecML Malware instances
  - **msf**. Includes only msfvenom instances
  - **Undetected**. Includes only msfvenom instances not detected by the initial model
- To build a *portfolio*, select a set of models to include and find cutoffs matching 1% FP
  - Use success on MLSEC Adversarial + EMBER Malware to break ties

Individual Model Results – 1% False Positive Rate

Individual Model	Local Benign	EMBER Malware	MLSEC Malware	MLSEC ‘19 Adversarial	MLSEC Malware RL	MLSEC SecML
Adversarial (All)	41.7%	4.4%	16.0%	52.6%	60.3%	47.9%
Adversarial (Benign)	40.1%	**53.8%**	**77.3%**	**86.6%**	84.3%	88.6%
Variants (All)	**95.3%**	9.3%	43.3%	78.3%	**89.9%**	**71.6%**
Variants (Benign)	**95.0%**	**60.6%**	**87.3%**	**88.2%**	**91.0%**	**94.4%**
Msf (All)	29.3%	0.4%	4.0%	8.1%	4.9%	15.8%
Msf (Benign)	24.5%	6.7%	50.7%	20.4%	35.5%	59.4%
Undetected (All)	21.6%	0.4%	46.7%	15.6%	39.4%	55.0%
Undetected (Benign)	72.3%	0.6%	4.0%	0.6%	2.6%	8.7%

Individual Model Results – 1% False Positive Rate

Individual Model	Local Benign	EMBER Malware	MLSEC Malware	MLSEC '19 Adversarial	MLSEC Malware RL	MLSEC SecML
Adversarial (All)	41.7%	4.4%	16.0%	52.6%	60.3%	47.9%
Adversarial (Benign)	40.1%	**53.8%**	**77.3%**	**86.6%**	84.3%	88.6%
Variants (All)	**95.3%**	9.3%	43.3%	78.3%	**89.9%**	**71.6%**
Variants (Benign)	**95.0%**	**60.6%**	**87.3%**	**88.2%**	**91.0%**	**94.4%**
Msf (All)	29.3%	0.4%	4.0%	8.1%	4.9%	15.8%
Msf (Benign)	24.5%	6.7%	50.7%	20.4%	35.5%	59.4%
Undetected (All)	21.6%	0.4%	46.7%	15.6%	39.4%	55.0%
Undetected (Benign)	72.3%	0.6%	4.0%	0.6%	2.6%	8.7%

- Variants performs best
- msf/undetected struggle to be useful
- **Benign** usually outperforms **All**…

Individual Model Results – 1% False Positive Rate

Individual Model	Local Benign	EMBER Malware	MLSEC Malware	MLSEC '19 Adversarial	MLSEC Malware RL	MLSEC SecML
Adversarial (All)	41.7%	4.4%	16.0%	52.6%	60.3%	47.9%
Adversarial (Benign)	40.1%	**53.8%**	**77.3%**	**86.6%**	84.3%	88.6%
Variants (All)	**95.3%**	9.3%	43.3%	78.3%	**89.9%**	**71.6%**
Variants (Benign)	**95.0%**	**60.6%**	**87.3%**	**88.2%**	**91.0%**	**94.4%**
Msf (All)	29.3%	0.4%	4.0%	8.1%	4.9%	15.8%
Msf (Benign)	24.5%	6.7%	50.7%	20.4%	35.5%	59.4%
Undetected (All)	21.6%	0.4%	46.7%	15.6%	39.4%	55.0%
Undetected (Benign)	72.3%	0.6%	4.0%	0.6%	2.6%	8.7%

- Variants performs best
- msf/undetected struggle to be useful
- **Benign** usually outperforms **All**…
- Undetected **All** does better than **Benign**

Portfolio Results

Model 1	Model 2	Model 3	Local Benign	EMBER Malware	MLSEC Malware	MLSEC '19 Adversarial	MLSEC Malware RL	MLSEC SecML
Initial	-	-	**97.6%**	**96.5%**	99.3%	53.9%	56.6%	76.4%
Retrained	-	-	**78.0%**	94.4%	96.7%	76.7%	84.0%	86.6%
Initial	Adversarial (All)	-	41.7%	**96.0%**	**100.0%**	83.1%	66.4%	94.6%
Initial	Adversarial (Benign)	-	41.4%	95.7%	**100.0%**	86.4%	70.6%	96.2%
Initial	Variants (All)	Msf (All)	37.5%	92.5%	92.0%	89.3%	84.9%	95.0%
Initial	Variants (All)	Msf (Benign)	28.5%	93.8%	98.0%	89.9%	84.2%	95.7%
Initial	Variants (All)	Undetected (All)	28.8%	92.5%	93.3%	**91.7%**	**89.0%**	95.9%
Initial	Variants (All)	Undetected (Benign)	70.5%	92.7%	92.0%	89.3%	**85.2%**	95.0%
Initial	Variants (Benign)	Msf (All)	37.5%	95.6%	**100%**	88.6%	78.3%	95.0%
Initial	Variants (Benign)	Msf (Benign)	60.7%	93.5%	95.3%	87.9%	81.1%	95.2%
Initial	Variants (Benign)	Undetected (All)	28.8%	95.6%	**100%**	**91.0%**	84.5%	**99.3%**
Initial	Variants (Benign)	Undetected (Benign)	70.5%	95.7%	**100%**	88.6%	78.8%	**97.2%**

Portfolio Results

Model 1	Model 2	Model 3	Local Benign	EMBER Malware	MLSEC Malware	MLSEC '19 Adversarial	MLSEC Malware RL	MLSEC SecML
Initial	-	-	**97.6%**	**96.5%**	99.3%	53.9%	56.6%	76.4%
Retrained	-	-	**78.0%**	94.4%	96.7%	76.7%	84.0%	86.6%
Initial	Adversarial (All)	-	41.7%	**96.0%**	**100.0%**	83.1%	66.4%	94.6%
Initial	Adversarial (Benign)	-	41.4%	95.7%	**100.0%**	86.4%	70.6%	96.2%
Initial	Variants (All)	Msf (All)	37.5%	92.5%	92.0%	89.3%	84.9%	95.0%
Initial	Variants (All)	Msf (Benign)	28.5%	93.8%	98.0%	89.9%	84.2%	95.7%
Initial	Variants (All)	Undetected (All)	28.8%	92.5%	93.3%	**91.7%**	**89.0%**	95.9%
Initial	Variants (All)	Undetected (Benign)	70.5%	92.7%	92.0%	89.3%	**85.2%**	95.0%
Initial	Variants (Benign)	Msf (All)	37.5%	95.6%	**100%**	88.6%	78.3%	95.0%
Initial	Variants (Benign)	Msf (Benign)	60.7%	93.5%	95.3%	87.9%	81.1%	95.2%
Initial	Variants (Benign)	Undetected (All)	28.8%	95.6%	**100%**	**91.0%**	84.5%	**99.3%**
Initial	Variants (Benign)	Undetected (Benign)	70.5%	95.7%	**100%**	88.6%	78.8%	**97.2%**

Portfolio Results

Model 1	Model 2	Model 3	Local Benign	EMBER Malware	MLSEC Malware	MLSEC ‘19 Adversarial	MLSEC Malware RL	MLSEC SecML
Initial	-	-	**97.6%**	**96.5%**	99.3%	53.9%	56.6%	76.4%
Retrained	-	-	**78.0%**	94.4%	96.7%	76.7%	84.0%	86.6%
Initial	Adversarial (All)	-	41.7%	**96.0%**	**100.0%**	83.1%	66.4%	94.6%
Initial	Adversarial (Benign)	-	41.4%	95.7%	**100.0%**	86.4%	70.6%	96.2%
Initial	Variants (All)	Msf (All)	37.5%	92.5%	92.0%	89.3%	84.9%	95.0%
Initial	Variants (All)	Msf (Benign)	28.5%	93.8%	98.0%	89.9%	84.2%	95.7%
Initial	Variants (All)	Undetected (All)	28.8%	92.5%	93.3%	**91.7%**	**89.0%**	95.9%
Initial	Variants (All)	Undetected (Benign)	70.5%	92.7%	92.0%	89.3%	**85.2%**	95.0%
Initial	Variants (Benign)	Msf (All)	37.5%	95.6%	**100%**	88.6%	78.3%	95.0%
Initial	Variants (Benign)	Msf (Benign)	60.7%	93.5%	95.3%	87.9%	81.1%	95.2%
Initial	Variants (Benign)	Undetected (All)	28.8%	95.6%	**100%**	**91.0%**	84.5%	**99.3%**
Initial	Variants (Benign)	Undetected (Benign)	70.5%	95.7%	**100%**	88.6%	78.8%	**97.2%**

Portfolio Results

Model 1	Model 2	Model 3	Local Benign	EMBER Malware	MLSEC Malware	MLSEC '19 Adversarial	MLSEC Malware RL	MLSEC SecML
Initial	-	-	**97.6%**	**96.5%**	99.3%	53.9%	56.6%	76.4%
Retrained	-	-	**78.0%**	94.4%	96.7%	76.7%	84.0%	86.6%
Initial	Adversarial (All)	-	41.7%	**96.0%**	**100.0%**	83.1%	66.4%	94.6%
Initial	Adversarial (Benign)	-	41.4%	95.7%	**100.0%**	86.4%	70.6%	96.2%
Initial	Variants (All)	Msf (All)	37.5%	92.5%	92.0%	89.3%	84.9%	95.0%
Initial	Variants (All)	Msf (Benign)	28.5%	93.8%	98.0%	89.9%	84.2%	95.7%
Initial	Variants (All)	Undetected (All)	28.8%	92.5%	93.3%	**91.7%**	**89.0%**	95.9%
Initial	Variants (All)	Undetected (Benign)	70.5%	92.7%	92.0%	89.3%	**85.2%**	95.0%
Initial	Variants (Benign)	Msf (All)	37.5%	95.6%	**100%**	88.6%	78.3%	95.0%
Initial	Variants (Benign)	Msf (Benign)	60.7%	93.5%	95.3%	87.9%	81.1%	95.2%
Initial	Variants (Benign)	Undetected (All)	28.8%	95.6%	**100%**	**91.0%**	84.5%	**99.3%**
Initial	Variants (Benign)	Undetected (Benign)	70.5%	95.7%	**100%**	88.6%	78.8%	**97.2%**

Portfolio Results

Model 1	Model 2	Model 3	Local Benign	EMBER Malware	MLSEC Malware	MLSEC '19 Adversarial	MLSEC Malware RL	MLSEC SecML
Initial	-	-	**97.6%**	**96.5%**	99.3%	53.9%	56.6%	76.4%
Retrained	-	-	**78.0%**	94.4%	96.7%	76.7%	84.0%	86.6%
Initial	Adversarial (All)	-	41.7%	**96.0%**	**100.0%**	83.1%	66.4%	94.6%
Initial	Adversarial (Benign)	-	41.4%	95.7%	**100.0%**	86.4%	70.6%	96.2%
Initial	Variants (All)	Msf (All)	37.5%	92.5%	92.0%	89.3%	84.9%	95.0%
Initial	Variants (All)	Msf (Benign)	28.5%	93.8%	98.0%	89.9%	84.2%	95.7%
Initial	Variants (All)	Undetected (All)	28.8%	92.5%	93.3%	**91.7%**	**89.0%**	95.9%
Initial	Variants (All)	Undetected (Benign)	70.5%	92.7%	92.0%	89.3%	**85.2%**	95.0%
Initial	Variants (Benign)	Msf (All)	37.5%	95.6%	**100%**	88.6%	78.3%	95.0%
Initial	Variants (Benign)	Msf (Benign)	60.7%	93.5%	95.3%	87.9%	81.1%	95.2%
Initial	Variants (Benign)	Undetected (All)	28.8%	95.6%	**100%**	**91.0%**	84.5%	**99.3%**
Initial	Variants (Benign)	Undetected (Benign)	70.5%	95.7%	**100%**	88.6%	78.8%	**97.2%**

Portfolio Results – what we used for kipple

Model 1	Model 2	Model 3	Local Benign	EMBER Malware	MLSEC Malware	MLSEC '19 Adversarial	MLSEC Malware RL	MLSEC SecML
Initial	-	-	**97.6%**	**96.5%**	99.3%	53.9%	56.6%	76.4%
Retrained	-	-	**78.0%**	94.4%	96.7%	76.7%	84.0%	86.6%
Initial	Adversarial (All)	-	41.7%	**96.0%**	**100.0%**	83.1%	66.4%	94.6%
Initial	Adversarial (Benign)	-	41.4%	95.7%	**100.0%**	86.4%	70.6%	96.2%
Initial	Variants (All)	Msf (All)	37.5%	92.5%	92.0%	89.3%	84.9%	95.0%
Initial	Variants (All)	Msf (Benign)	28.5%	93.8%	98.0%	89.9%	84.2%	95.7%
Initial	Variants (All)	Undetected (All)	28.8%	92.5%	93.3%	**91.7%**	**89.0%**	95.9%
Initial	Variants (All)	Undetected (Benign)	70.5%	92.7%	92.0%	89.3%	**85.2%**	95.0%
Initial	Variants (Benign)	Msf (All)	37.5%	95.6%	**100%**	88.6%	78.3%	95.0%
Initial	Variants (Benign)	Msf (Benign)	60.7%	93.5%	95.3%	87.9%	81.1%	95.2%
Initial	Variants (Benign)	Undetected (All)	28.8%	95.6%	**100%**	**91.0%**	84.5%	**99.3%**
Initial	Variants (Benign)	Undetected (Benign)	70.5%	95.7%	**100%**	88.6%	78.8%	**97.2%**

RESULTS

How did kipple do?

MLSEC Home Phishing Malware Defender User Management Rules README Logout

Defender scoreboard. Lists the total number of times an ML model was bypassed. The smaller the number, the better the result.

Please note, only submissions involving ZIP uploads are counted here, fast API ML checks are not.

List

	ML "secret" fmbuylrn bypassed	ML "submission 3" qhdyuvnv bypassed	ML "scanner_only_v1" tlgwdpam bypassed	ML "model2_thresh90" vftuemab bypassed	ML "A1" amsqr bypassed	ML "kipple" rwchsfde bypassed
	162	1840	714	734	193	231

KIPPLE: 3RD PLACE FINISHER IN MLSEC 2021

Was in first place up until 48 hours before!

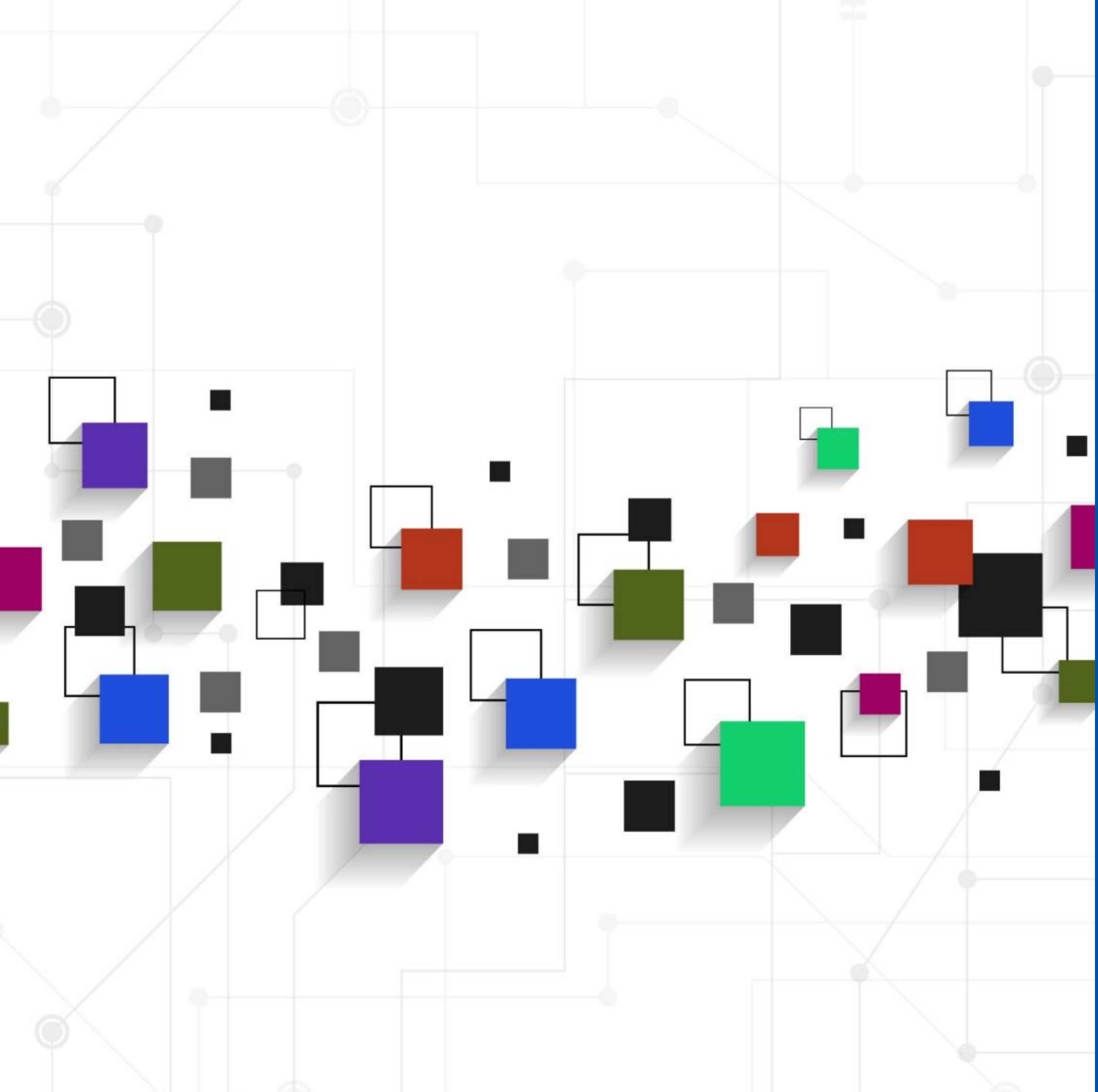
(final submission included stateful correlation, higher thresholds, and built-in MD5 signaturing for benignware)

CRITICAL ANALYSIS: AREAS OF IMPROVEMENT

- *Kipple* has a low false positive rate
- *Kipple* still misses Malware RL-style attacks
  - Given enough time, can evade with random decisions
  - Frameworks like MAB-malware proved (+/-) successful
    - https://github.com/weisong-ucr/MAB-malware
- More importantly: *kipple* lacked knowledge of *traditional, non-ML* evasion techniques
  - Crypters, packers, etc.
  - Multiple off-the-shelf tools were able to bypass *kipple*'s detection

Individual Model Results – 0.01% False Positive Rate

Individual Model	EMBER Malware	MLSEC 2019 – All	MLSEC Malware RL
Initial	0.0%	0.0%	0.0%
Retrained	**71.2%**	36.0%	72.2%
Adversarial (All)	4.0%	**40.7%**	**91.2%**
Adversarial (Benign)	10.8%	**40.1%**	**91.5%**
Variants (All)	3.8%	**40.4%**	**91.4%**
Variants (Benign)	11.1%	**42.9%**	**91.4%**
Msf (All)	0.0%	0.0%	0.0%
Msf (Benign)	0.3%	0.2%	1.2%
Undetected (All)	0.0%	4.2%	5.7%
Undetected (Benign)	0.2%	0.7%	0.6%



CLOSING THOUGHTS AND DISCUSSION

Kipple might not be solving the "robustness" problem – but we think this research still helps

Major Conclusions

- Can we build a classifier that's robust to adversarial examples without sacrificing normal accuracy?

Yes!

- Is it better to use adversarial retraining or a portfolio of models?

Portfolios look better

- When training on adversarial examples, is it better to train on only those that bypassed classification?

From the msf/undetected case – bypass is better!

- Is it worthwhile to write a classifier that discriminates between normal PE files (malware and benign) versus adversarially-generated ones?

Surprisingly – it's not entirely clear!

LESSONS LEARNED

- Make sure you have space
- Make sure you dedicate enough time
- Keep good records
- Ensure training and testing data are separate

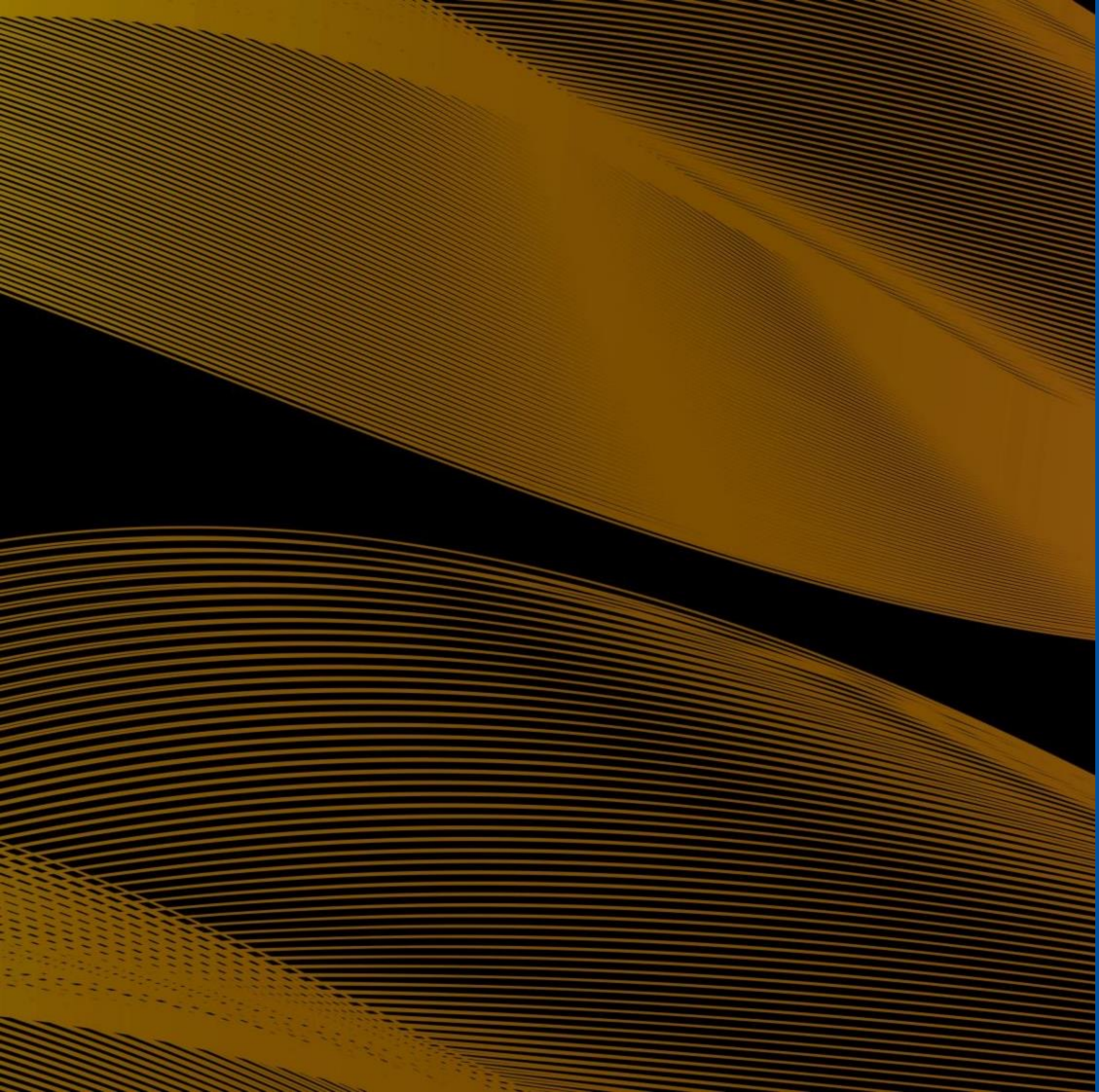
DISCUSSION QUESTIONS

- Does a bigger ensemble targeting traditional obfuscation perform better?
- Can we generate *more* adversarial malware in a way that's time-efficient?
- Would our models perform better trained on *only* evasive samples?
- How can we tweak + optimize the existing adversarial malware frameworks?

THANK YOU

@andyplayse4

https://github.com/aapplebaum/kipple



APPENDIX

Attacking the Competition

- Attempted to attack the two frontrunners (secret and amsqr) to “defend” kipple
  - If we can score against these two, we’ll make kipple seem relatively better
- Tried for model stealing attack
  - Threw benign + adversarial samples at each model
    - 20K in total! ~2500 benign, ~14500 Malware RL, ~4000 GAMMA
  - Trained a GBDT matching the results
  - Evaded our trained model
- Didn’t really work – subject of future talk…
- But did profile the two other models reasonably well
  - Admittedly hard to compare results due to stateful detection

Accuracy	secret	amsqr
Benign	38.3%	89.3%
Malware RL	95.3%	96.5%
GAMMA	97.9%	90.6%

Malware RL

- Open source: https://github.com/bfilar/malware_rl
- OpenAI *gym* extension to train reinforcement learning agents to create evasive malware
  - Builds on older gym-malware work
  - Large action space – each functionality-preserving
  - Idea to train agent to know which sequence of actions to apply to be evasive
- Comes with:
  - Random agent
  - Pre-trained MalConv and EMBER models
- Our usage:
  - Use local benign “train” samples as labeled benign
  - Use random agent to generate MalConv-evading samples

Action Space

```
ACTION_TABLE = {
    'modify_machine_type': 'modify_machine_type',
    'pad_overlay': 'pad_overlay',
    'append_benign_data_overlay': 'append_benign_data_overlay',
    'append_benign_binary_overlay': 'append_benign_binary_overlay',
    'add_bytes_to_section_cave': 'add_bytes_to_section_cave',
    'add_section_strings': 'add_section_strings',
    'add_section_benign_data': 'add_section_benign_data',
    'add_strings_to_overlay': 'add_strings_to_overlay',
    'add_imports': 'add_imports',
    'rename_section': 'rename_section',
    'remove_debug': 'remove_debug',
    'modify_optional_header': 'modify_optional_header',
    'modify_timestamp': 'modify_timestamp',
    'break_optional_header_checksum': 'break_optional_header_checksum',
    'upx_unpack': 'upx_unpack',
    'upx_pack': 'upx_pack'
}
```

Table 1: *Evasion Rate against Ember Holdout Dataset**

gym	agent	evasion_rate	avg_ep_len
ember	RandomAgent	89.2%	8.2
malconv	RandomAgent	88.5%	16.33

SecML Malware

- Extension of *SecML*; library for executing a variety of white-box and black-box attacks against ML classifiers
- Includes multiple built-in attack types, as well as a pre-trained MalConv instance
- Open source: https://github.com/pralab/secml_malware
- Our usage:
  - Leverage local “benign” train samples as input to attacks
  - Run several attack types to generate (not necessarily evasive) samples and save them

MalConv original DR: 100%

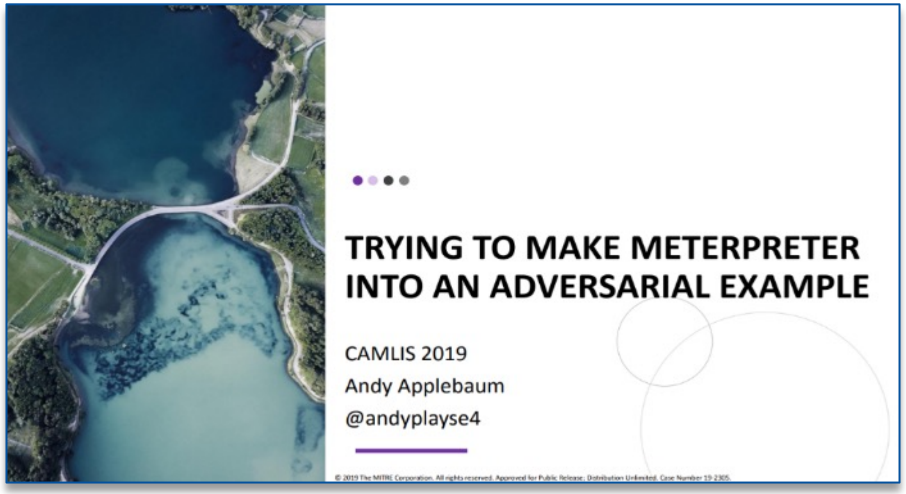
	White-box attacks					Black-box attacks				
	Partial DOS	Extend	Shift	Padding		Partial DOS	Extend	Shift	Padding	GAMMA-padding
1 iter.	60%	5%	87.5%	85%	10 queries	69%	34%	80%	100%	14%
25 iter.	28%	5%	80%	45%	250 queries	56%	25%	79%	100%	13%
50 iter.	28%	5%	80%	45%	500 queries	42%	10%	65%	100%	12%

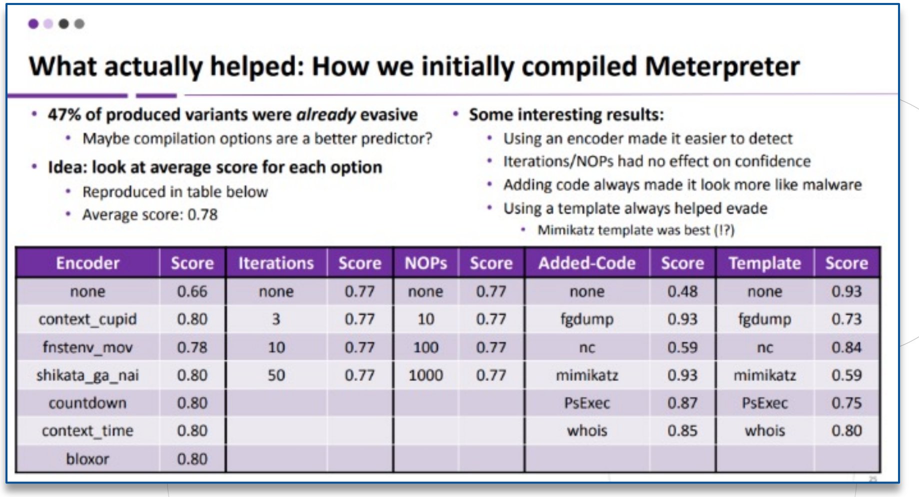
Table 2: Detection Rates (DRs) of MalConv against white-box/black-box attacks, optimized with an increasing number of iterations/queries.

Demetrio, Luca, and Battista Biggio. "Secml-malware: A Python library for adversarial robustness evaluation of windows malware classifiers." arXiv preprint arXiv:2104.12848 (2021).

msfvenom

- Alternative approach: use *msfvenom* to compile new “malware” (i.e., implants)
- Randomly choose options:
  - Architecture (x86, x64)
  - Encoder (none, xor, xor_dynamic, shikata_ga_nai)
  - Encryption (none, aes256, base64, rc4, xor)
  - Payload (shell, meterpreter)
  - Template (local benign train data)
  - Added code (none, VirusShare, SoReL)
- Our usage:
  - Save all, marking if an instance is evasive
  - Record which added code type chosen

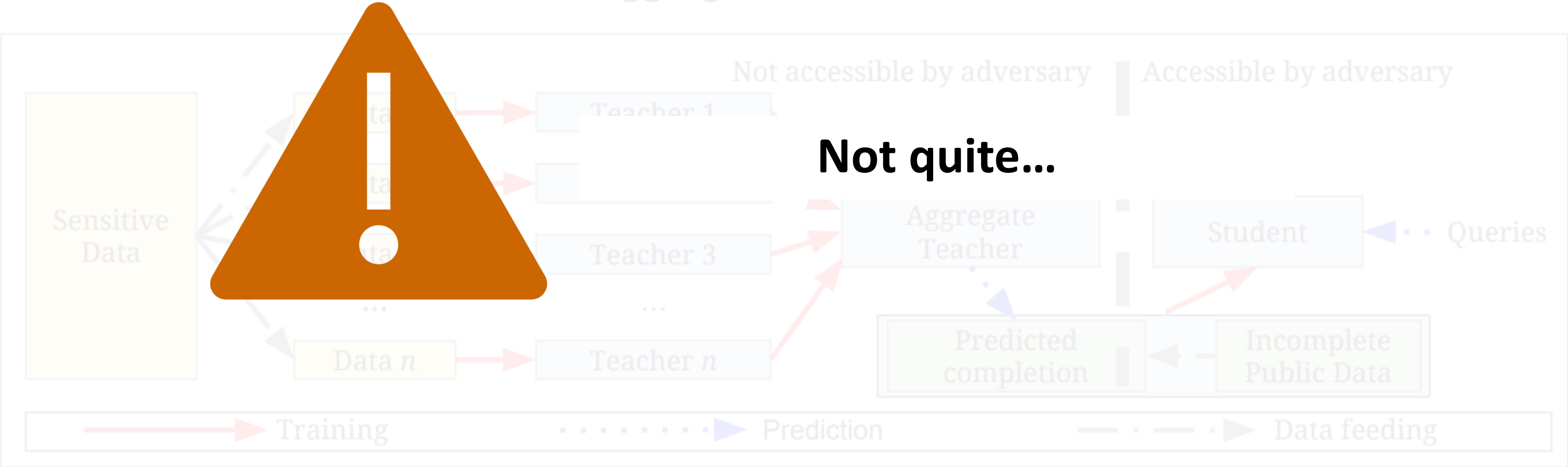




Encoder	Score	Iterations	Score	NOPs	Score	Added-Code	Score	Template	Score
none	0.66	none	0.77	none	0.77	none	0.48	none	0.93
context_cupid	0.80	3	0.77	10	0.77	fgdump	0.93	fgdump	0.73
fnstenv_mov	0.78	10	0.77	100	0.77	nc	0.59	nc	0.84
shikata_ga_nai	0.80	50	0.77	1000	0.77	mimikatz	0.93	mimikatz	0.59
countdown	0.80					PsExec	0.87	PsExec	0.75
context_time	0.80					whois	0.85	whois	0.80
bloxor	0.80								

Teaser: Differential Privacy

PATE: Private Aggregation of Teacher Ensembles



Not quite...

Papernot, Nicolas, et al. "Semi-supervised knowledge transfer for deep learning from private training data." arXiv preprint arXiv:1610.05755 (2016).