TRYING TO MAKE METERPRETER INTO AN ADVERSARIAL EXAMPLE

CAMLIS 2019

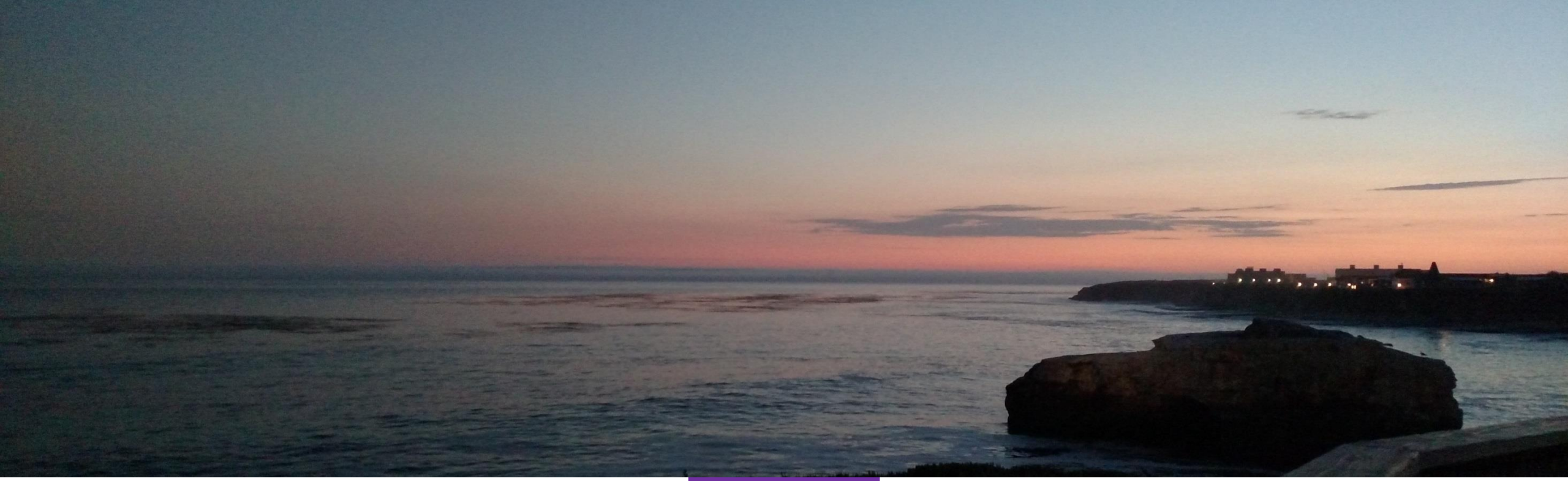
Andy Applebaum

@andyplayse4

OUTLINE

- **About Me**
  - Researcher at MITRE[1] (ATT&CK, CALDERA, security + AI, etc.)
  - Prior research background: argumentation logic + security
  - Recently developed a strong personal interest in attacks against AI
- **This presentation: how do adversarial examples relate to classifying malware?**
  - *Can we use RL to generate evasive versions of Meterpreter?*
- **Includes:**
  - A survey (*crash course*) of existing work
  - A short experiment
  - Analysis of experimental results
- **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.



INTRODUCTION

Before there were adversarial examples, there was machine learning…

MACHINE LEARNING HAS PUT HARD PROBLEMS IN REACH

- **Natural language processing**
  - Translating text
  - Identifying voice commands
- **Computer vision**
  - Object/image recognition
  - Object classification
- **Game playing (reinforcement learning)**
  - Computer games (DotA, Starcraft)
  - Board games (Go, Chess)
- **Many, many more…**

https://blog.statsbot.co/deep-learning-achievements-4c563e034257

… including security!

- **Applied across different "cyber" problems**
  - Malware classification
  - Password guessing
  - Network intrusion detection
  - System log anomaly detection

Raff, Edward, et al. "An investigation of byte n-gram features for malware classification." *Journal of Computer Virology and Hacking Techniques* 14.1 (2018): 1-20.

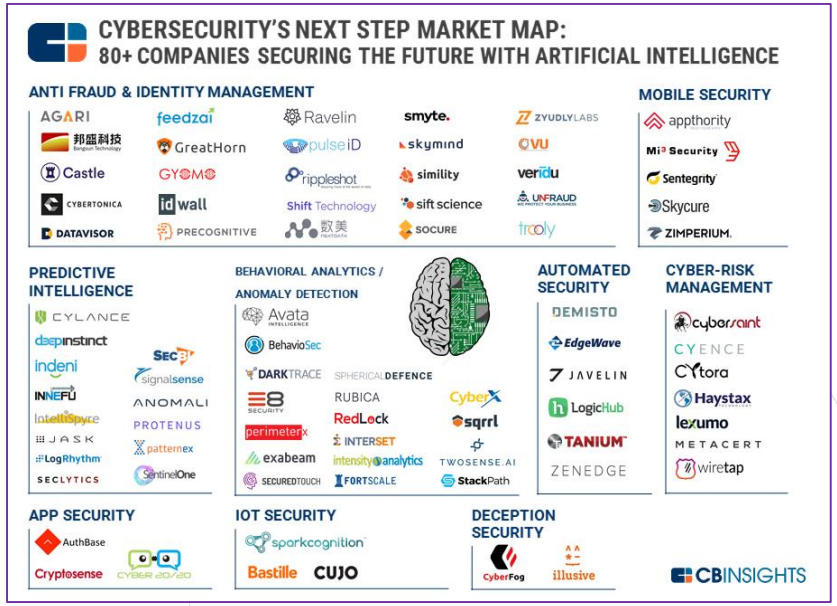
Melicher, William, et al. "Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks." *USENIX Security Symposium*. 2016.

Sommer, Robin, and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection." *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010.

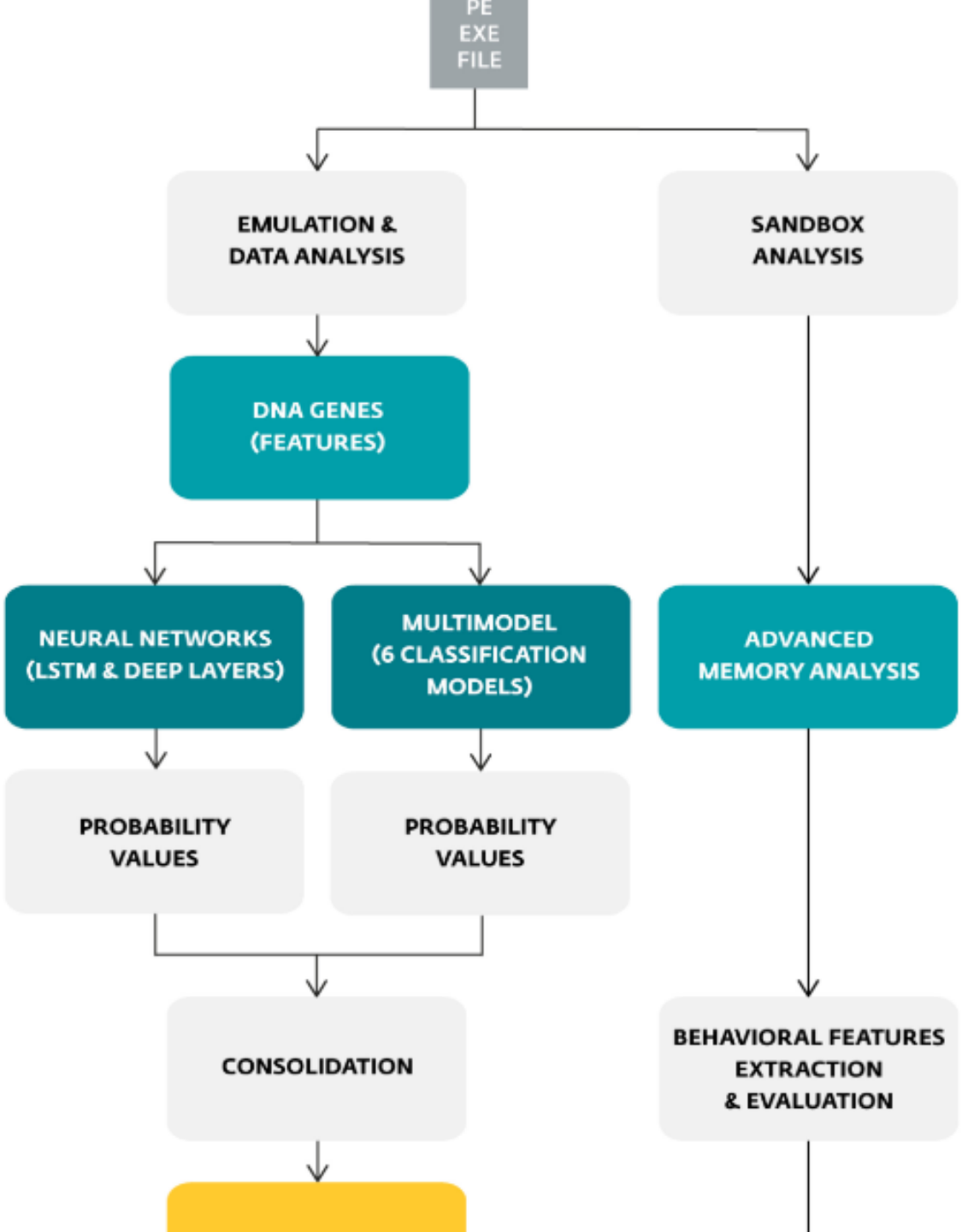
Du, Min, et al. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." *Proceedings of the 2017 ACM SIGSAC*. ACM, 2017.

- **Across both academia *and* industry**
  - Both in startups and bigger players
  - Sold across domains/not just detection

- ***Most security now has an AI + ML story***
  - **(we're here, after all!)**



https://www.cbinsights.com/research/cybersecurity-artificial-intelligence-startups-market-map/



ESET, https://i.crn.com/sites/default/files/ckfinderimages/userfiles/images/crn/custom/2018/ESET_360_Q318_NextGen_Whitepaper.pdf

IS MACHINE LEARNING A SILVER BULLET?

(no)

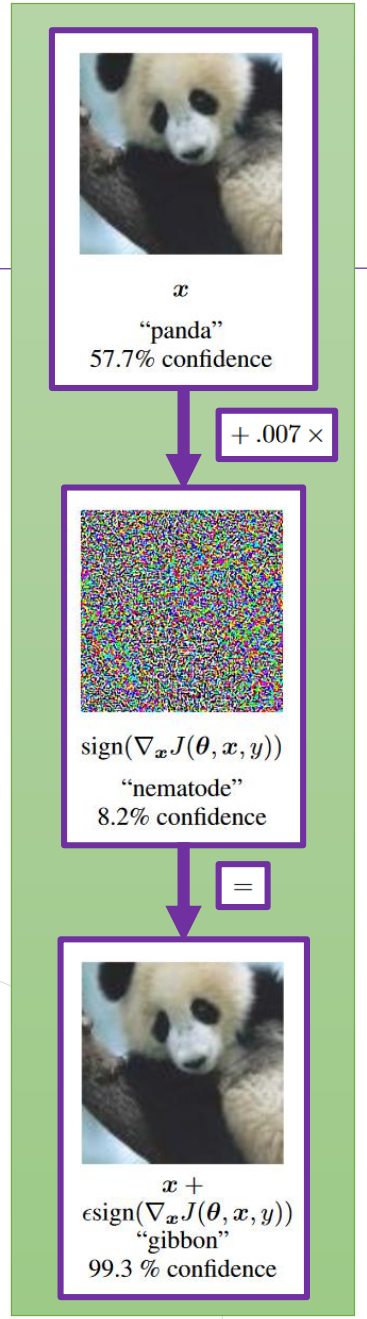
Enter: Adversarial Examples Against Neural Networks

- **2013: "Intriguing Properties of Neural Networks," Szegedy et al.**
- Any *good* classifier should have some robustness and stability of its classifications
- Small adjustments to images shouldn't significantly alter classifications; they should stay the same
  - **Yet**: "…we find that applying an imperceptible non-random perturbation to a test image, it is possible to arbitrarily change the network's prediction. These perturbations are found by optimizing the input to maximize the prediction error. We term the so perturbed examples 'adversarial examples'."

- Left: Correctly classified sample image
- Middle: added noise
- Right: misclassified (*ostrich*) result

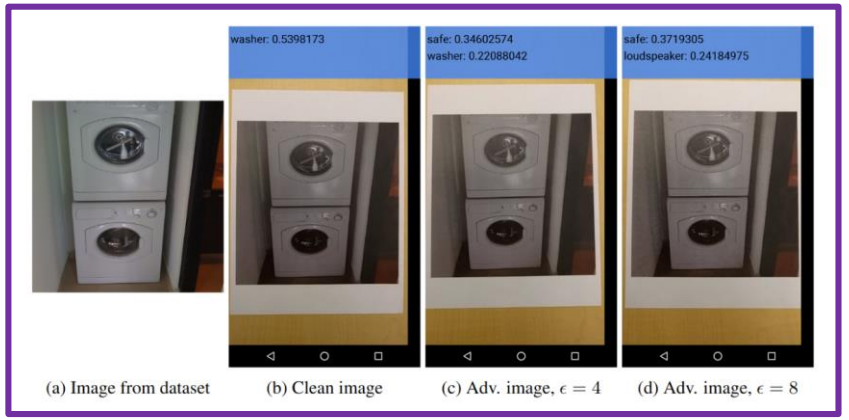
The Field Grows: Advancing Adversarial Examples

- **Early construction approaches required full access to model/details (2015)**
  - Goodfellow et al.: “Explaining and Harnessing Adversarial Examples”
  - Papernot et al.: “The Limitations of Deep Learning in Adversarial Settings”
- **Over time, other construction techniques (same pre-reqs) emerge**
  - Moosavi-Dezfooli et al.: “DeepFool: a simple and accurate method to fool deep neural networks”
  - Carlini and Wagner: “Towards evaluating the robustness of neural networks”
- **Hiding the classifier makes it harder, but other techniques are effective**
  - Papernot et al.: “Practical Black-Box Attacks against Machine Learning”
  - **Querying the classifier for a label is enough!**
- **Other early defensive approaches show progress, but can still be bypassed**
  - Carlini and Wagner: “Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods” (2017)



Bad News: Adversarial Examples Exist in the Real World Too

- **Can control for noise to remain adversarial when printed**
  - Kurakin et al.: *"Adversarial Examples in the Physical World"* (2016)
- **Can modify real-world objects for misclassification**
  - Evtimov et al.: *"Robust Physical-World Attacks on Deep Learning Models"* (2017)
    - Modify street signs for misclassification
  - Brown et al.: *"Adversarial Patch"* (2018)
    - Specific "patch" causes any image to be classified as a toaster



Kurakin et al.

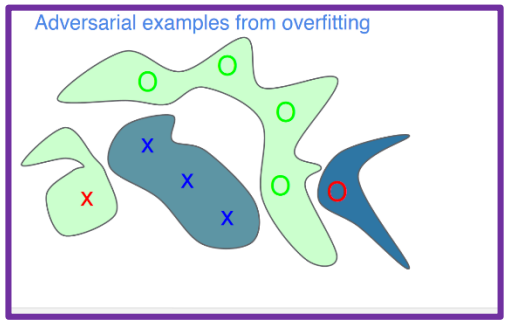

TABLE IV: Sample experimental images for the attacks detailed in Table V and Table VI at a selection of distances and angles.

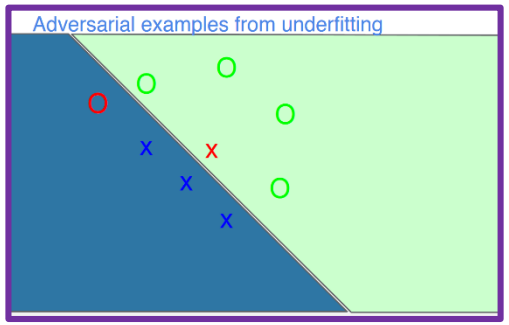
Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB*-CNN)
5' 0°					
5' 15°					
10' 0°					
10' 30°					
40' 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Evtimov et al.

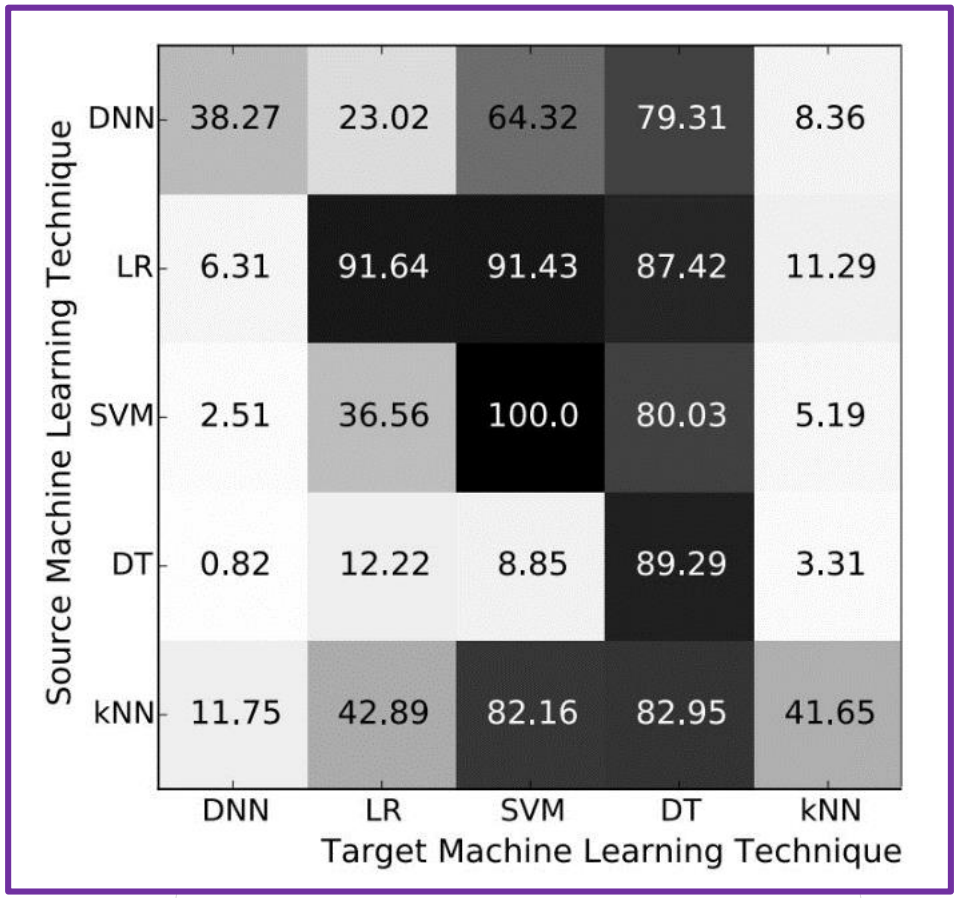
Adversarial Examples: *Not* a One-Off Problem

- Defenses are not always intuitive
- Assuming a hidden classifier is not enough
- Attacks against one classifier frequently transfer to another, even if of different type
- **<u>Bottom line: if we're using a classifier, we're going to need to consider adversarial examples</u>**





Ian Goodfellow. "Adversarial Examples." Deep Learning Summer School, 2015.



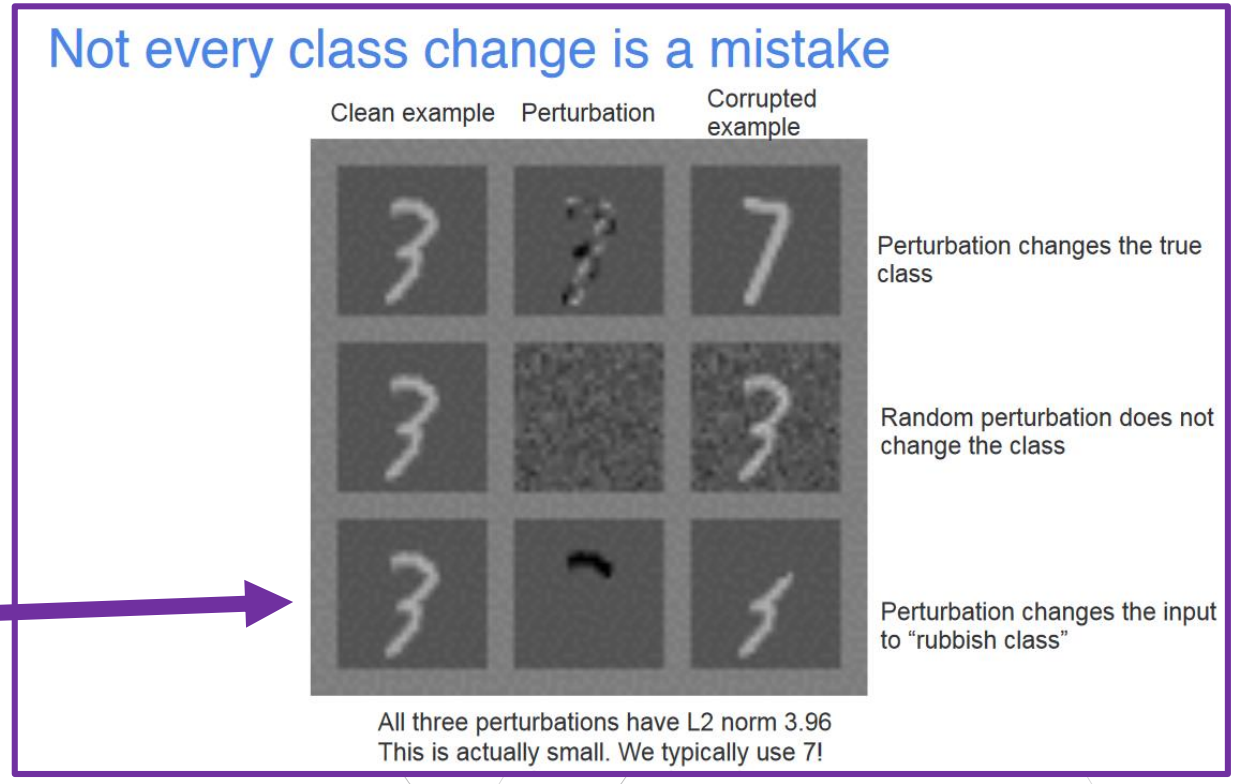
Nicolas Papernot. "Adversarial Examples in Machine Learning." Talk at USENIX Enigma 2017.

ADVERSARIAL EXAMPLES ARE KIND OF A BIG DEAL

But how do they relate to security?

Adversarial Examples in Security Applications

- **Security is a *fundamentally different* domain than image recognition**
  - Hybrid approaches: static, dynamic, anomaly, signature, etc.
- **Perturbations are not easy to apply**
  - Modifying files often corrupts them
  - “rubbish class”
- **How well does the concept of adversarial examples transfer over?**



Ian Goodfellow. “Adversarial Examples.” Deep Learning Summer School, 2015.

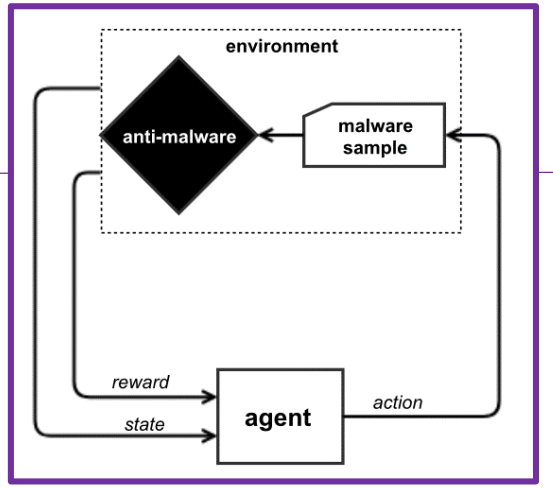
Reasonably Well!

Prior research in adversarial examples in security applications

- **Most research focuses on creating "adversarial malware" that bypasses ML classifiers**
- **Works across three different threat models: white, gray, and black-box**
- **White-Box**: Grosse et al.: "Adversarial Perturbations Against Deep Neural Networks for Malware Classification."
  - Focused on Android malware – only adds features to avoid rubbish class problems
  - Given full knowledge of model, computes best theoretical perturbation and applies it
- **Gray-Box:** Xu et al.: "Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers."
  - Focused on PDFs – perturbs then detonates them to ensure they work to avoid rubbish class problems
  - No knowledge of model, but can query the classifier with a sample and receive label + confidence
  - Uses a genetic algorithm to figure out what modifications make the file evasive
- **Black-Box:** Hu and Tan: "Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN."
  - Focuses on PE files, but only works in feature space – modifies features and not files themselves
  - No knowledge of model, but can query the classifier with the sample and receive a label
  - Uses a generative network to modify features to make samples evasive

Black-box Attacks: PE Malware (Files)

Anderson et al.: "Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning," 2018.



- Works with real PE files, actually modifying the binary to add features; actions include:
  - Adding an uncalled function to the import table
  - Manipulating existing section names
  - Packing or unpacking the file
  - Appending bytes to extra space at the end of sections
- Use reinforcement learning to construct adversarial examples, obtaining only the label of tested samples
- Train/test on different malware classes: random malware (VirusShare), ransomware, Virut, and BrowseFox
- Mixed results: agent not significantly better than random, but scores strong on VirusTotal

dataset	agent	random
VirusShare	24%	23%
ransomware	12%	9%
Virut	10%	9%
BrowseFox (adware)	19%	18%

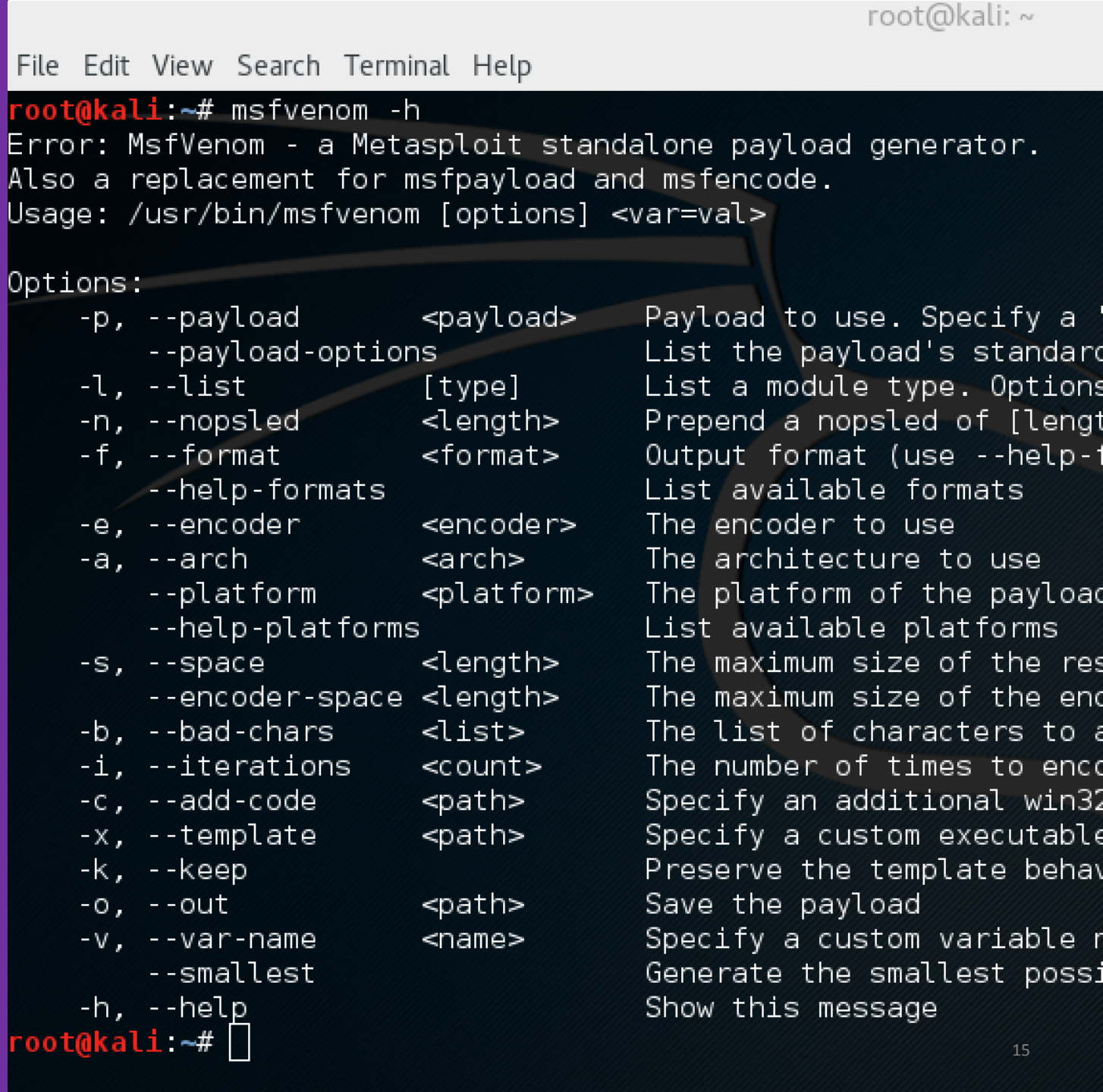
Table 2: Evasion rate on 200 holdout samples. A samples is included in the calculation only if the classifier correctly identified the original sample as malicious.

dataset	original	agent	original	random
VirusShare	54/65	26/65	54/65	16/65
ransomware	52.5/65	16.5/65	44.5/65	9.5/65
Virut	57/65	25/65	56.5/65	20/65
BrowseFox (adware)	49/65	18/65	49/65	21/65

Table 4: Cross-evasion rates on 200 holdout samples, showing the median number of VirusTotal detections before and after mutation using agent mutations and random mutations.

EXPERIMENTING WITH ADVERSARIAL EXAMPLES

Testing the black-box scenario at home



Endgame's gym-malware package

- Anderson et al. released an OpenAI Gym environment to accompany their paper
- Allows you to "train" an agent via a series of games:
  1. Each game is against a specific malware sample
  2. During the agent's turn, it can select a (pre-built) PE modification
  3. Test the modified malware against the classifier
     - If it's now classified as benign, end the game and give a big reward
     - Otherwise, keep playing the game until you run out of turns
- Games can be run black-box or (not published!) gray-box
  - Black-box: no intermediary rewards, large reward if successfully evaded
  - Gray-box: intermediary reward of shift-in-confidence, large on evasion

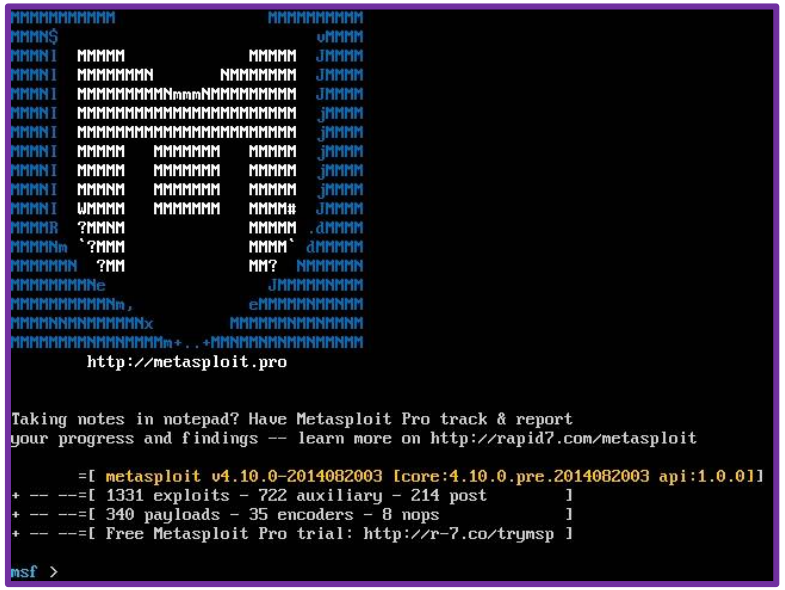
- append_zero
- append_random_ascii
- append_random_bytes
- remove_signature
- upx_pack
- upx_unpack
- change_section_names_from_list
- change_section_names_to random
- modify_export
- remove_debug
- break_optional_header_checksum

gym-malware action space: https://github.com/endgameinc/gym-malware

- Includes built in ACER agent to learn a policy and classifier implementing a gradient boosted decision tree trained on 50k benign + 50k malicious samples
- ***To run: we just need to bring the malware!***

Using gym-malware for research: Making Meterpreter Evasive

- The idea: use the default Metasploit implant (Meterpreter) for testing + training
  - Well known/detected: **49/68** on VirusTotal
- From a technical perspective, features include:
  - Useful command shell features (e.g., tab completion)
  - Running executables
  - Sending/receiving files
  - Taking screenshots
  - Privilege escalation
  - Command and control



- Q1: **will the reinforcement learning technique be effective on a single malware type?**
- Q2: **can we perturb the detector so severely to make well-known malware evasive?**
- Q3: **how effective is gym-malware's (not-reported-in-the-paper) gray-box agent?**

Creating Meterpreter variants with msfvenom

- The problem: **gym-malware needs a large sample size to start with**
- The solution: **dynamically compile different variants of Meterpreter with *msfvenom***
- **msfvenom**: command-line tool to compile different implants/shells with a variety of parameters, including:
  - **Payload**: different payloads (Meterpreter, shell, etc.)
  - **Format**: payload type (shell code, executable, PHP file, etc.)
  - **Encoder**: encode using specified encoder; can sometimes help evade AV
  - **Iterations**: the number of times to apply the encoder
  - **Added-code**: includes an additional PE file during compilation
  - **NOPsled size**: include NOPs at the start of the payload
  - **Template**: use an existing executable as a “template” for how to structure the output

msfvenom: Generated Meterpreter Variants

- Wrote a script that generated a total of 4032 different Meterpreter variants
- All run on x86 Windows, output as a PE file, use reverse TCP, and write-back to 8.8.8.8
- Each variant selects a specific value from the table below for each column

Encoder	Iterations	NOPs	Added-Code	Template
none	none	none	none	none
context_cpuid	3	10	fgdump	fgdump
fnstenv_mov	10	100	nc	nc
shikata_ga_nai	50	1000	mimikatz	mimikatz
countdown			PsExec	PsExec
context_time			whois	whois
bloxor				

Diving into gym-malware: notes on settings

- Detection threshold: **0.9** (default)
  - Classifier returns a score between 0 (benign) and 1 (malicious)
  - Detection threshold defines where the cutoff is (i.e., 0.9 and above are malicious)
  - Samples classified as “benign” before mutation are thrown out
- Game length: **10** (default) and **20**
  - Game length controls the maximum number of mutations per malware sample
  - Run two separate experiments, varying game length to see if it increases accuracy
- Total mutations: **20000** (default: 50000)
  - Total number of training trials the agent can run before terminating training
  - Less than default for quicker training, potentially at the cost of accuracy)
- Train + Test size: **2098** and **200**
  - 1734/4032 (43%) of initial variants classified as benign

Results

- **…largely underwhelming**
  - No significant difference between agents
  - No significant different between game lengths

Agent	% Evaded: 10 Trials	% Evaded: 20 Trials
Random	17.1	18.9
Black-box	17.1	16.2
Gray-box	16.5	22.5

% of test samples (200) that were successfully made evasive by agent/length

- **Discovered variants largely uninteresting**
  - Average score slightly under threshold
  - No difference between gray/black-box
  - (only have results for length 20)

Agent	# Evasive Variants	Average Evasive Variant Score
Black-box	645	0.87
Gray-box	709	0.87

Statistics for evasive variants produced during training

POST-MORTEM

So what happened?

Managing Expectations

- **Original paper did *not* claim that this technique always produces evasions!**
  - Table 2 shows no notable difference between the ACER agent and random
- **Instead:**
  - Can use discovered variants to retrain
  - Can use evasions to foil AV vendors
- **Still leaves questions about gray agent: wouldn't we expect it to outperform the black-box agent?**

dataset	agent	random
VirusShare	24%	23%
ransomware	12%	9%
Virut	10%	9%
BrowseFox (adware)	19%	18%

Table 2: Evasion rate on 200 holdout samples. A samples is included in the calculation only if the classifier correctly identified the original sample as malicious.

2018: "Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning," Anderson et al.

dataset	original	agent	original	random
VirusShare	54/65	26/65	54/65	16/65
ransomware	52.5/65	16.5/65	44.5/65	9.5/65
Virut	57/65	25/65	56.5/65	20/65
BrowseFox (adware)	49/65	18/65	49/65	21/65

Table 4: Cross-evasion rates on 200 holdout samples, showing the median number of VirusTotal detections before and after mutation using agent mutations and random mutations.

Looking at the actions: Are our modifications actually helping?

- **Applied each modification to each variant 5 times**
  - Recorded average confidence score for those 5
  - Did not chain any modifications – just one at a time
- **Record overall average confidence change (+ / -)**
  - Summed over all variants
- **For all changes, the modifications made the classifier *more* confident that the file was malicious!**
  - The gray agent's results make more sense!
- **But… why?! Potential explanations:**
  - Classifier trained on already obfuscated files
  - Meterpreter already sufficiently optimized
  - These obfuscations are, generally, ineffective

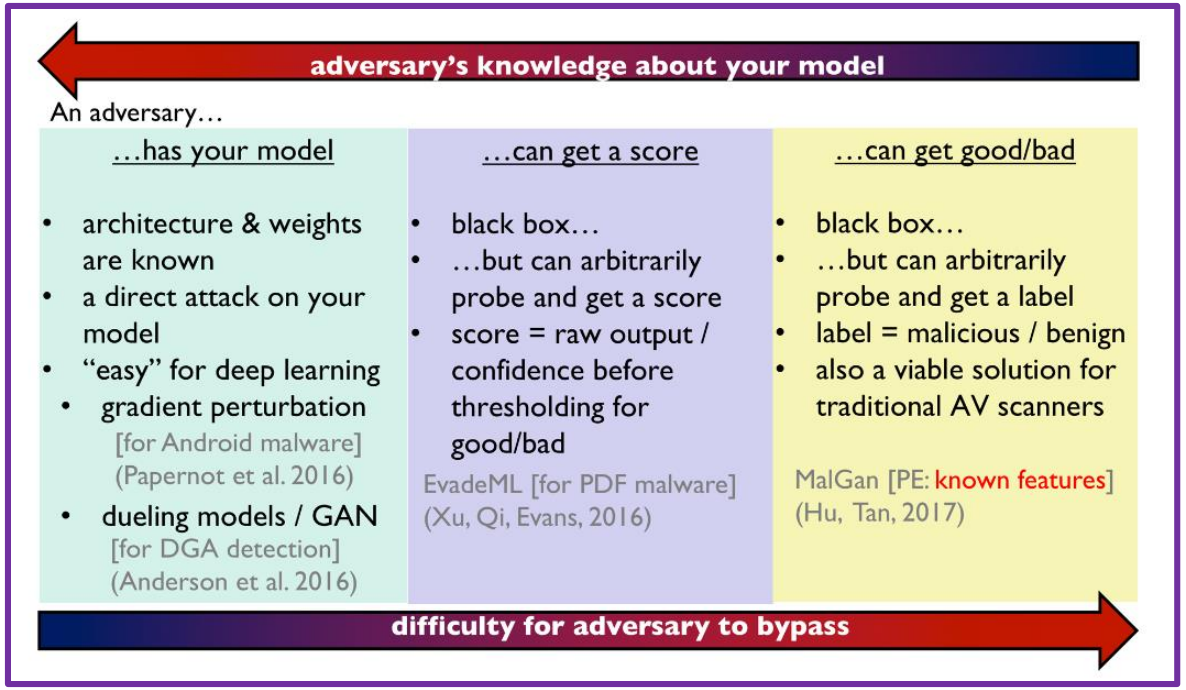
Modification	Score Δ
Overlay_append	+ 0.00
Imports_append	+ 0.15
Section_rename	+ 0.05
Section_add	+ 0.07
Section_append	+ 0.05
Create_new_entry	+ 0.14
Remove_signature	+ 0.05
Remove_debug	+ 0.02
Upx_pack	+ 0.14
Upx_unpack	+ 0.00
Break_optional_header	+ 0.05

What actually helped: How we initially compiled Meterpreter

- **47% of produced variants were *already* evasive**
  - Maybe compilation options are a better predictor?
- **Idea: look at average score for each option**
  - Reproduced in table below
  - Average score: 0.78
- **Some interesting results:**
  - Using an encoder made it easier to detect
  - Iterations/NOPs had no effect on confidence
  - Adding code always made it look more like malware
  - Using a template always helped evade
    - Mimikatz template was best (!?)

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								

CLOSING THOUGHTS



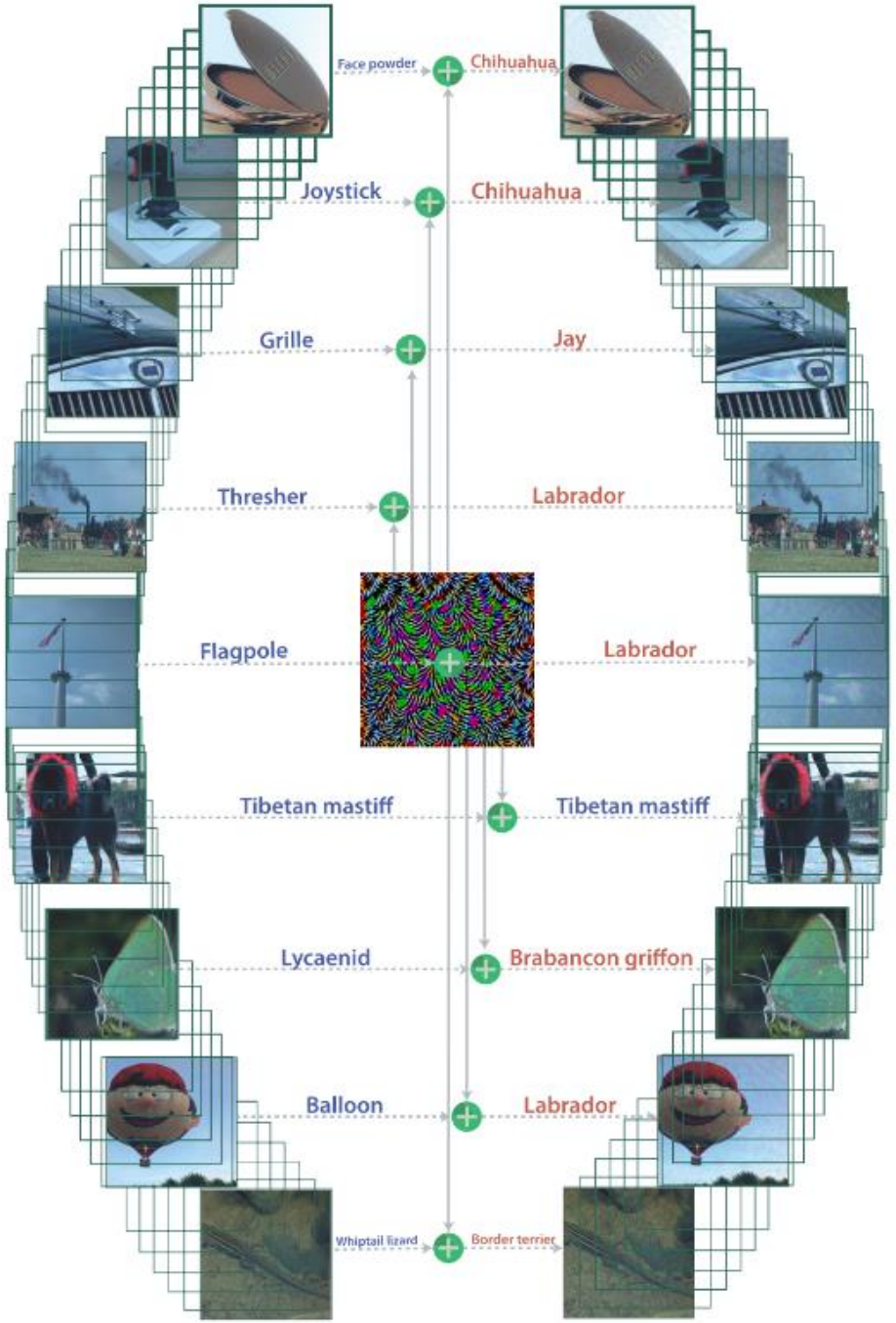
Anderson et al. "Evading Machine Learning Malware Detection." 2017.

Summary and Lessons Learned

- Q1: **will the reinforcement learning technique be effective on a single malware type?**
  - *No – but the paper already showed that*
- Q2: **can we perturb the detector so severely to make well-known malware evasive?**
  - *Not really – the malware was compiled to be evasive as-is*
- Q3: **how effective is gym-malware's (not-reported-in-the-paper) gray-box agent?**
  - *Surprisingly: not very, at least not here*
- **Takeaways:**
  - Understand the threat model you're considering (and the prior work behind it!)
  - Analyze your data before running your experiments
  - Read the source material closely when forming hypotheses
  - There's lots of little things (e.g., settings, training data) that can feed into experiments

For the Future

- **Finish this experiment**
  - Show how action space/compilation options interact
  - (try other compilation options…)
- **Investigate the gray agent in more detail**
  - Still unanswered: can we use RL to train an agent to bypass the classifier?
  - Black-box testing is good, but testing the gray agent seems more likely to good
  - Other modifications: threshold, action space, training/testing data, classifier
- **Can we smartly optimize the *generation* of adversarial example malware?**
  - Here, we “brute forced” the search space – but easy to envision a bigger one!
  - Solves problems with making modifications that ensure functionality



THANK YOU!

@andyplayse4