ML SEMINAR
OCTOBER 13/14TH 2021
SECRET.INF.UFPR.BR

All You Need to Know to Win a Cybersecurity Adversarial Machine Learning Competition and Make Malware Attacks Practical

Fabrício Ceschin
Federal University of Paraná, BR
@fabriciojoc
fjoceschin@inf.ufpr.br

Marcus Botacin
Federal University of Paraná, BR
@MarcusBotacin
mfbotacin@inf.ufpr.br

Who are We?

Fabrício Ceschin
Federal University of Paraná, BR
@fabriciojoc
fjoceschin@inf.ufpr.br

- CS Master (Federal University of Paraná, Brazil)
- Computer Science PhD Student (Federal University of Paraná, Brazil)
- ML Researcher (Since 2015)
- **Interests:** ML applied to Security, ML applications (Data Streams, Concept Drift, Adversarial Machine Learning)

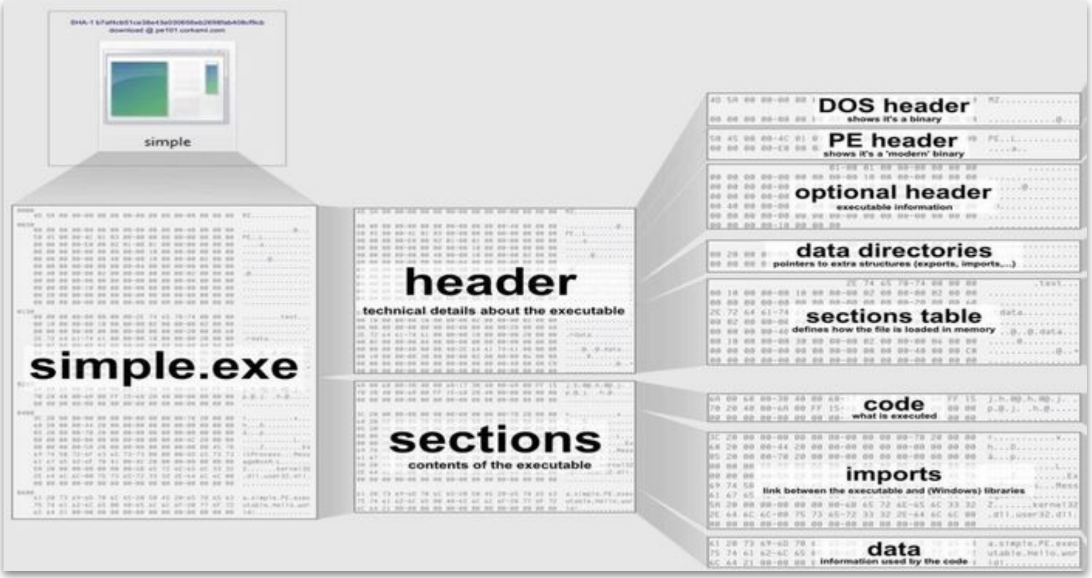
Marcus Botacin
Federal University of Paraná, BR
@MarcusBotacin
mfbotacin@inf.ufpr.br

- CS Master (University of Campinas, Brazil)
- Computer Science PhD Student (Federal University of Paraná, Brazil)
- Malware Analyst (Since 2012)
- **Interests:** Malware Analysis & Detection, Hardware-Assisted Security

Introduction

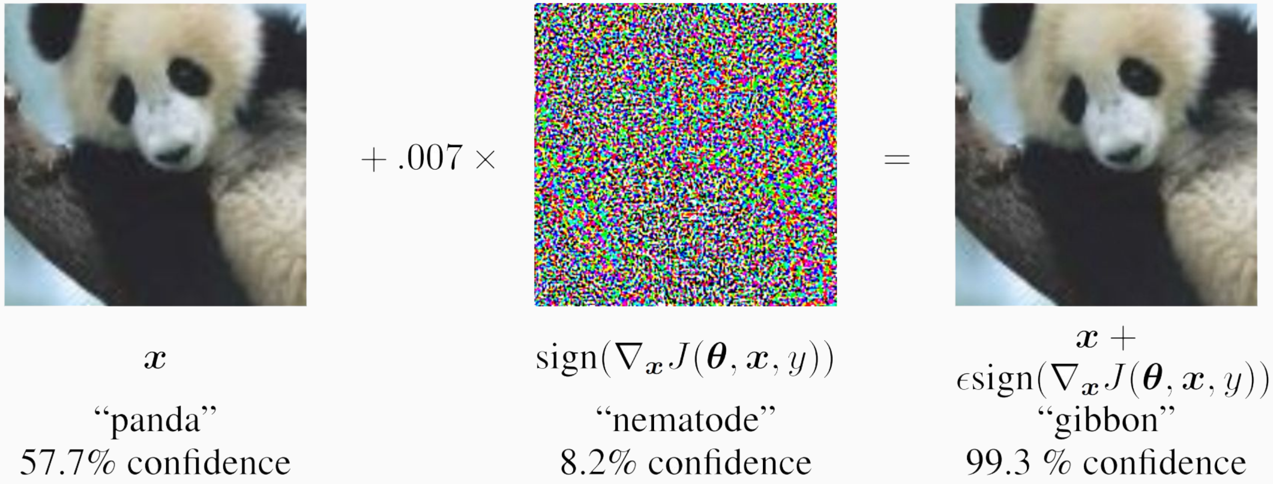
How to Detect a (Windows) Malware?

- **Static Detection:** real-time detection without executing it
- **Analyse Portable Executable (PE) File:** check its header and sections
    - Executable information
    - Code, libraries, and data



Adversarial Machine Learning

- **Adversarial Machine Learning:** trend in recent years, as everybody knows



Adversarial Malware is Different

- **Image Classification:** adversarial image should be **similar** to the original one and yet be classified as being from another class
- **Malware Detection:** adversarial malware should **behave** the same and yet be classified as **goodware**
- **Challenge:** automatically generating a fully functional adversarial malware may be difficult
  - Any modification can make it behave different or not work
  - Many solutions in literature, but **malware do not work!**

Introduction: Machine Learning Security Evasion Competition (MLSEC)

- **Our Experience:** three wins in MLSEC contests!
- **Public Challenge:** contest to better understand adversarial samples impact in static ML-based malware detectors
- **Contribution:** insights gained on attacking/defending models

Year	2019	2020	2021
Attacker Challenge	1st (draw)	1st	1st
Defender Challenge	-	2nd	1st

The Challenge

Defender Challenge

- **Objective:** participants develop their own ML defensive solutions, with models of their own choice and trained using any dataset
- **Three requirements:**
    - Less than 1% of False Positive Rate (FPR)
    - Less than 10% of False Negative Rate (FNR)
    - Must return a response within 5 seconds for any presented sample
- **Winner:** the model that presents the fewer number of evasions in the attacker challenge

Attacker Challenge

- **Objective:** all models that achieved the previous requirements are made available to be attacked by black-box attacks
- **Data provided:** 50 unique working Windows malware samples
- **Participants:** provide new binaries for the same malware samples
  - Bypass classifiers and present same behavior (Indicators of Compromise, IoC) in sandbox
  - **Maximum size:** 5mb in 2019; 2mb in 2020/2021
- **Winner:** the attacker that has most bypassed classifiers and performs the lowest number of queries (tiebreaker rule)

MLSEC 2019

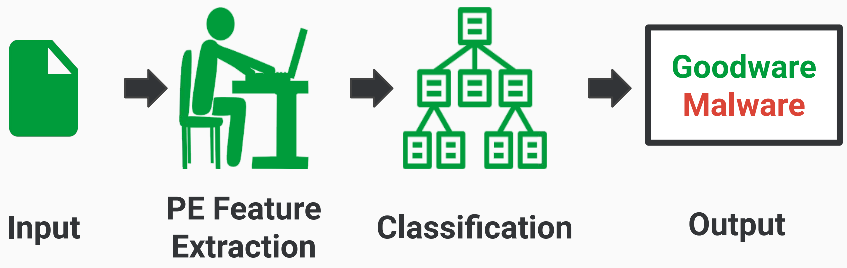
The First Edition of MLSEC (2019)

- **There was no Defender Challenge:** models were selected by organizers
- **Three Models:**
  - LightGBM[1]
  - MalConv[2]
  - Non-Negative MalConv[3]

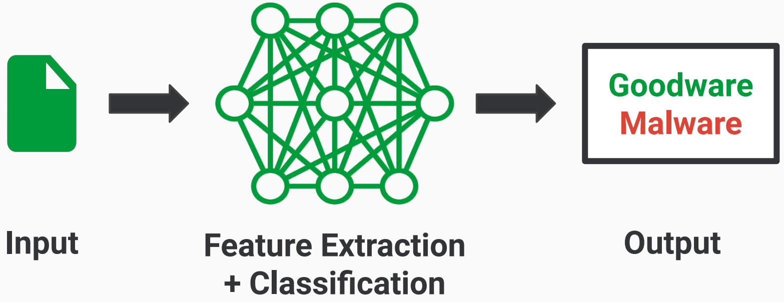
[1] https://arxiv.org/abs/1804.04637
[2] https://arxiv.org/abs/1710.09435
[3] https://arxiv.org/abs/1806.06108

Models: PE Parsing vs Raw Bytes

- **LightGBM:** Gradient boosting, hashing trick and histograms
- PE Parsing (header info, file size, timestamp, libraries, strings, etc)

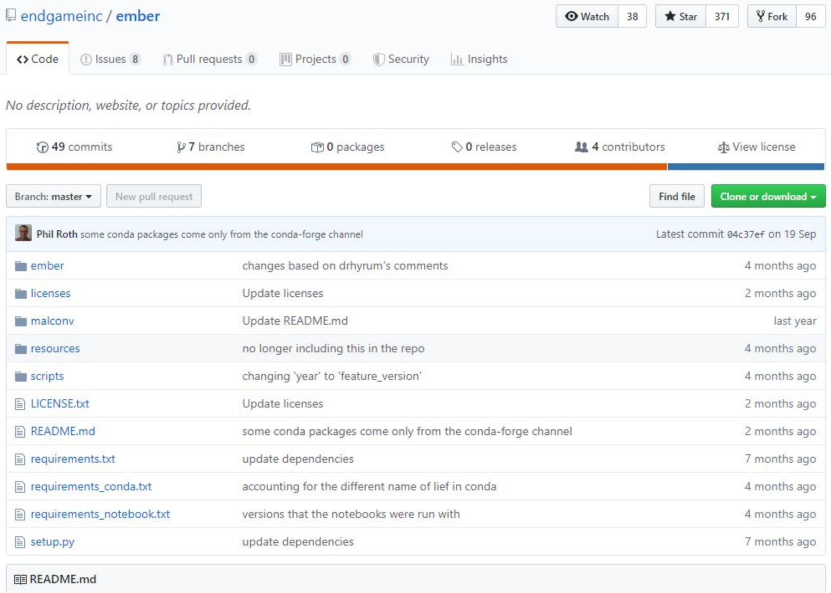


- **MalConv & Non Neg. MalConv:** End-to-end deep learning models
  - **Non Neg.:** force model to look only for malicious evidences
- Raw bytes as input (no parsing)



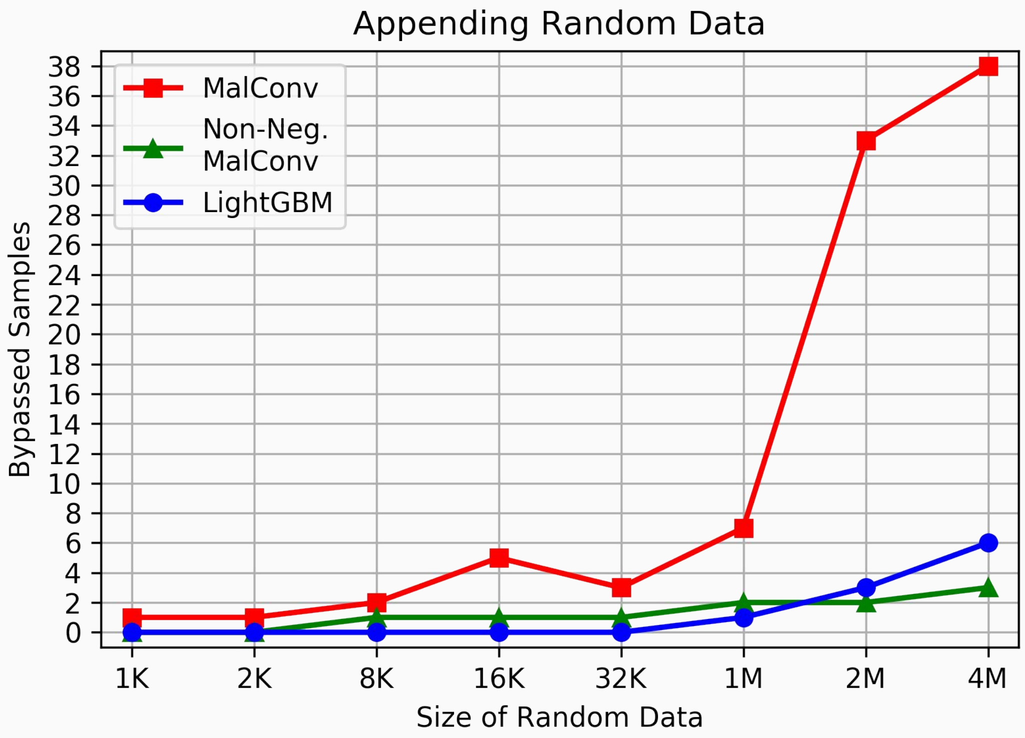
MLSEC 2019 Models: Train Dataset

- Ember 2018 dataset
- Benchmark for researchers
- 1.1M Portable Executable (PE) binary files:
  - 900K training samples;
  - 200K testing samples
- Open Source dataset:
  - https://github.com/elastic/ember



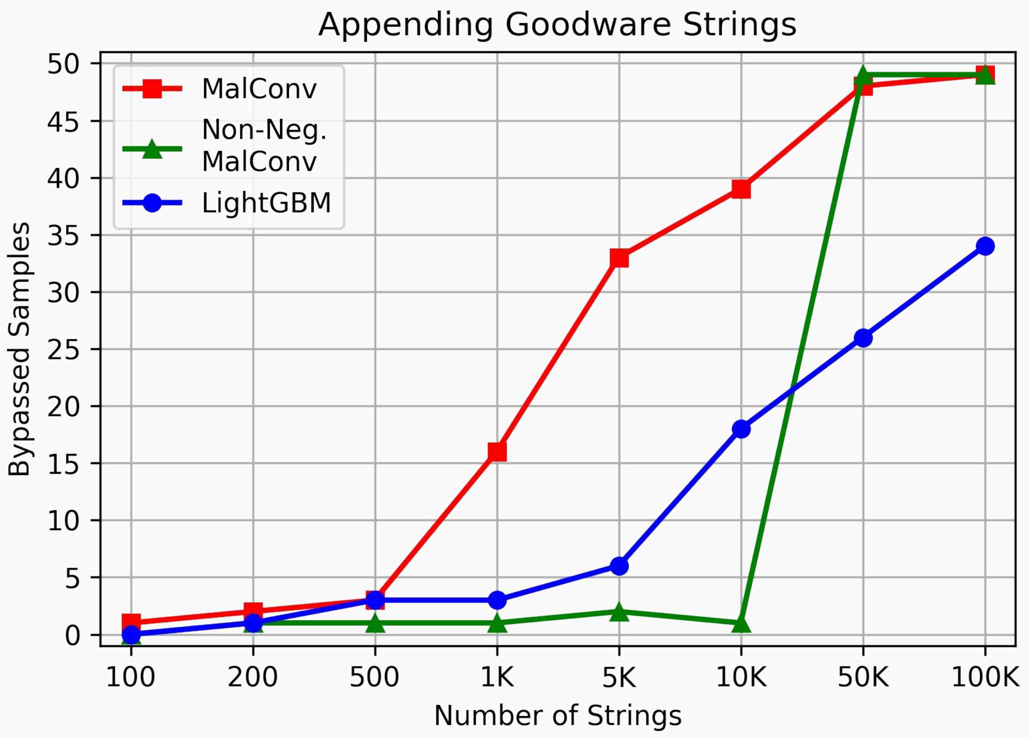
Attack: Appending Random Data

- Generating growing chunks of random data, up to the limit of 5MB defined by the challenge
  - MalConv, based on raw data, is more susceptible to this strategy
  - Severe for chunks greater than 1MB
  - Some features and models might be more robust than others
  - Non-Neg. MalConv and LightGBM were not so affected



Attack: Appending Goodware Strings

- Retrieving strings presented by goodware files and appending them to malware binaries
- All models are significantly affected when 10K+ strings are appended
- Result holds true even for the model that also considers PE data (LightGBM), which was more robust in the previous experiment



Attack: Packing and Unpacking samples with UPX

- UPX-packed versions are more detected by all classifiers
- Classifiers biased towards the detection of UPX binaries, despite their content

Dataset	MalConv	Non-Neg MalConv	LightGBM
Originally Packed			
UPX	63.64%	55.37%	89.26%
Extracted UPX	59.50%	53.72%	66.12%
Originally Non-Packed			
Original	65.35%	54.77%	67.23%
UPX Packed	67.43%	56.43%	88.12%

Attack: Embedding Samples in a Dropper

1. Retrieves a pointer to the binary resource (line 3 to 5)
2. Creates a new file to drop the resource content (line 7)
3. Drop the entire content (line 8 to 10);
4. Launches a process based on the dropped file (line 13)

- Bypass all models (data appending)

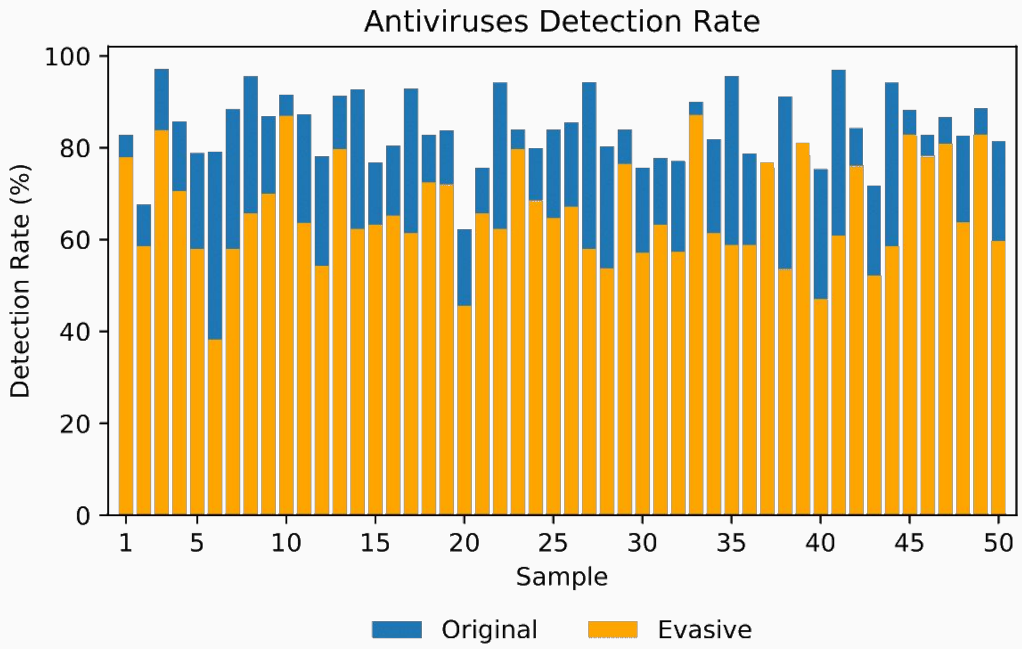
```
1  int main(){
2      HMODULE h = GetModuleHandle(NULL);
3      HRSRC r = FindResource(h, ...);
4      HGLOBAL rc = LoadResource(h,r);
5      void* data = LockResource(rc);
6      DWORD size = SizeofResource(h,r);
7      FILE *f = fopen("dropped.exe","wb");
8      for(int i=0;i<size;i++){
9          unsigned char c1 = ((char*)data)[i];
10         fprintf(f,"%c",c1);
11     }
12     fclose(f);
13     CreateProcess("dropped.exe", ...);
```

Adversarial Malware Generation: Results

	Malware (mw)		Goodware (gw_i)		Adversarial Malware ($mw+$)	
Model	**Class**	**Confidence**	**Class**	**Confidence**	**Class**	**Confidence**
MalConv	Malware	99.99%	Goodware	69.54%	Goodware	81.22%
Non-Neg. MalConv	Malware	75.09%	Goodware	73.32%	Goodware	98.65%
LightGBM	Malware	100.00%	Goodware	99.99%	Goodware	99.97%
Average	Malware	91.69%	Goodware	80.95%	Goodware	93.28%

Adversarial Malware in Real World

- Could our strategy be leveraged in real world by actual attackers?
- **VirusTotal service:** detection rates for adversarial samples
- **Results:** our approach also affected real AV engines
  - Sample 6 dropping almost in half
- **Explanation:** AV engines also powered by ML models
  - Subject to same weaknesses and biases



MLSEC 2020

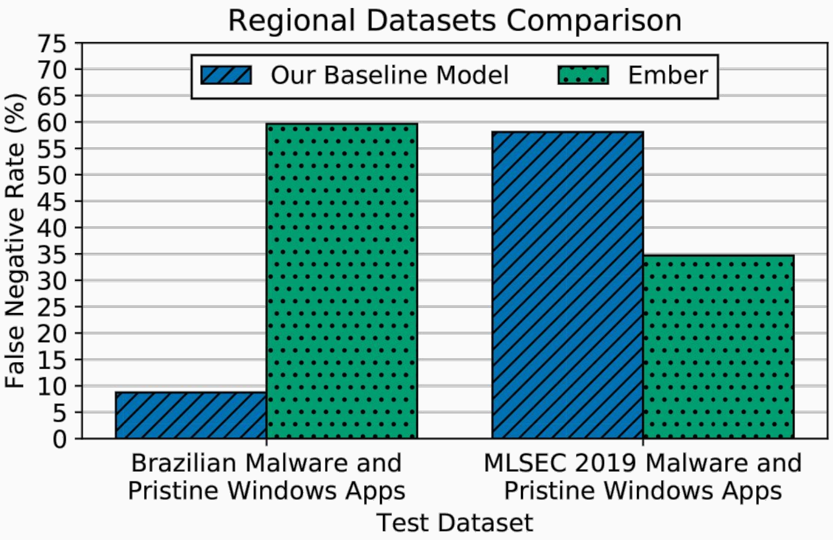
Defense Solution: Our Initial Model

- **First thought:** use as baseline a research model developed by us[1]
  - **Implementation:** TF-IDF on top of PE Parsing and Random Forest classifier
  - **Training set:** malware samples collected in the Brazilian cyberspace
  - **Results in our paper:** 98% of f1-score with a low false-negative rate
- **When testing with EMBER test samples:** bad results, totally different from expected
  - **Biased:** samples from EMBER are different from Brazilian malware
  - **Hypothesis:** classifiers used in Brazilian cyberspace are not the most suitable for global samples (EMBER)

[1]Fabrício Ceschin, Felipe Pinage, Marcos Castilho, David Menotti, Luis S Oliveira, and André Gregio. The Need for Speed: An Analysis of Brazilian Malware Classifiers. IEEE Security Privacy 16, 6 ([n. d.]), 31–41.

Defense Solution: Regional Datasets

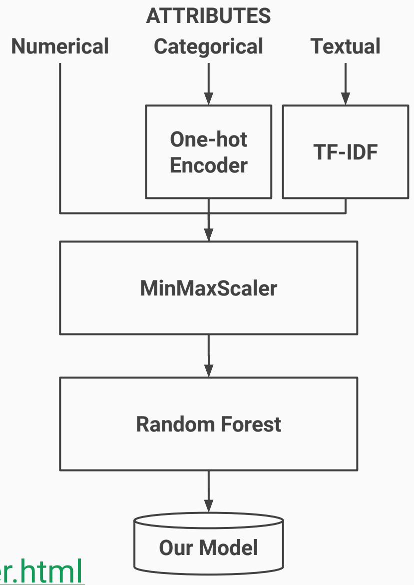
- **Test hypothesis:** train our model with EMBER dataset[1]
  - **Compare:** Brazilian malware model[2]
  - **Evaluation:** BRMalware and MLSEC 19
- **Regional datasets/models:** each model performs better in their own region
  - Each region has its own characteristics
  - Specially crafted for a given region
- **Ember as training dataset:**
  - More suitable dataset for the challenge



[1]H. Anderson and P. Roth. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. ArXiv e-prints. Apr. 2018.
[2]F. Ceschin, F. Pinage, M. Castilho, D. Menotti, L. S. Oliveira, A. Grégio. The Need for Speed: An Analysis of Brazilian Malware Classifiers. IEEE Security Privacy, 2018.

Definitive Defense Solution

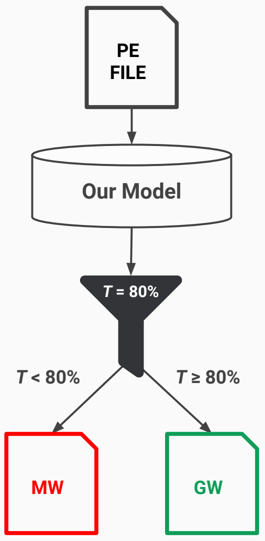
- **Definitive model:** selected attributes from the EMBER datasets
  - **Three types of attributes:** numerical, categorical and textual
    - **Categorical:** transformed into one-hot encoding array
    - **Textual:** texts, separated by spaces, transformed into sparse array with their TF-IDF
  - **Normalization:** MinMaxScaler (numerical, categorical and textual features concatenation)
- **Train:** EMBER's 1.6 million labeled samples[1]
  - Scikit-learn Random Forest[2] with 100 estimators



[1] https://github.com/elastic/ember
[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Fine-tuning Our Defense Solution

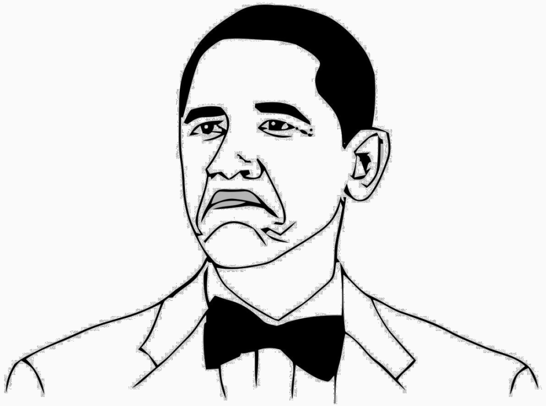
- **Objective:** reduce the impact of adversarial perturbations
  - Force classification to be more aggressive
- **New prediction function:** uses model class probabilities as input to determine the output class
  - **Threshold *T*:**
    - If prob(goodware) ≥ ***T***, sample = goodware; Otherwise, malware
- Make our classifier perform as required by the competition:
  - **Default Random Forest prediction function:** FPR of 8.5%*
  - **Threshold *T* = 80%:** FPR of 0.1%*



* Using EMBER test set (selected samples not used in the training set)

Our Model vs. MLSEC 2019 Adversaries

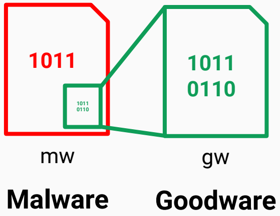
- **Initial test:** submit 2019 adversarial samples provided by the organizers
  - **594 samples:** variations of 50 original samples from last year's challenge
- **Results:**
  - Detected 88.91% of the samples
- **All 2019 models were bypassed:** significant good
- **Confirmed our findings from previous challenge:**
  - Models based on parsing PE files are better than the ones based on raw data

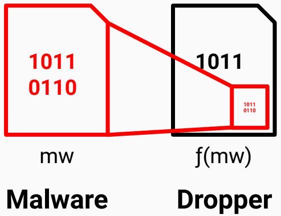


NOT BAD

Attack Solution: The Beginning

- **Three models accepted:** ember, nfs (our model), and domumpqb
- **Initial strategy:** appending goodware strings and random bytes to original samples
  - **44 points:**
    - 36 bypassed ember (LightGBM)
    - 8 bypassed need for speed (our solution)
    - none bypassed domumpqb
- **Using 2019 solution:** embedding the original sample in a "Dropper"
  - New binary that embeds original malware sample as a resource
  - Fully bypassed the first model only (ember), just ⅓ of 2020 challenge!





Attack Solution: Attacking Ourselves

- **Focus in bypassing our own model:** “know yourself before you know others”
- **Our model:** based on the library imports and their respective functions
  - **Detecting dropper:** presence of a functions such as ***FindResource***, used by droppers
- **First ideia:** hide the ***FindResource*** API calls from the classifier
  - Compress our samples with ***Telock***[1], ***PELock***[2], and ***Themida***[3]
- **Reducing the number of imports:** increased the confidence on the malware label
  - **Reinforces last year's claim:** classifiers learn packers as malicious regardless its content
  - Also happens with real AVs[4]

[1]http://www.telock.com-about.com

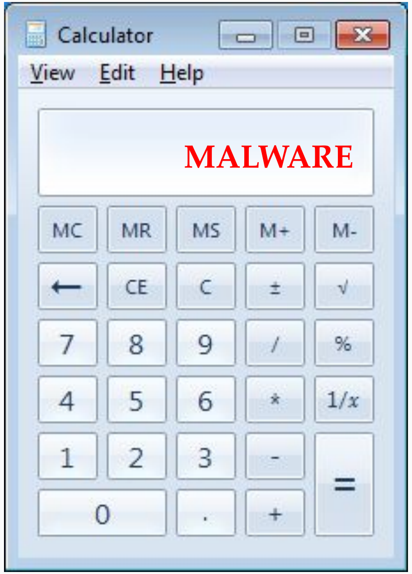
[2]https://www.pelock.com

[3]https://www.oreans.com/Themida.php

[4]Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. 2020. When Malware is Packin'Heat; Limits of Machine Learning Classifiers Based on Static Analysis Features. In NDSS Proceedings(NDSS). NDSS, US, 1.

Attack Solution: Mimicking Calculator

- **Alternative:** search for some benign sample that present the same imports
- **Calculator (calc.exe):** imports series of functions, including ***FindResource***
  - **Report:** benign with 100% of confidence level by our classifier
  - **Our goal:** build a new dropper binary mimicking the calculator



Black-Box is Harder, but not Impossible

- **Two of three models:** previous knowledge about models
  - **Ember:** deployed in the last year's contest
  - **Need for speed:** developed by us
  - **Domumpqb:** deploy a full black box attack
- **Few samples had already bypassed it:** 21 samples
- **Hypothesis:** it is detecting part of the embedded payload, the only part that changes (all droppers are similar)
- **Solution:** hide the embedded payload
  - Encoding the malware binary as a base64 string
  - XORing the malware binary with a key

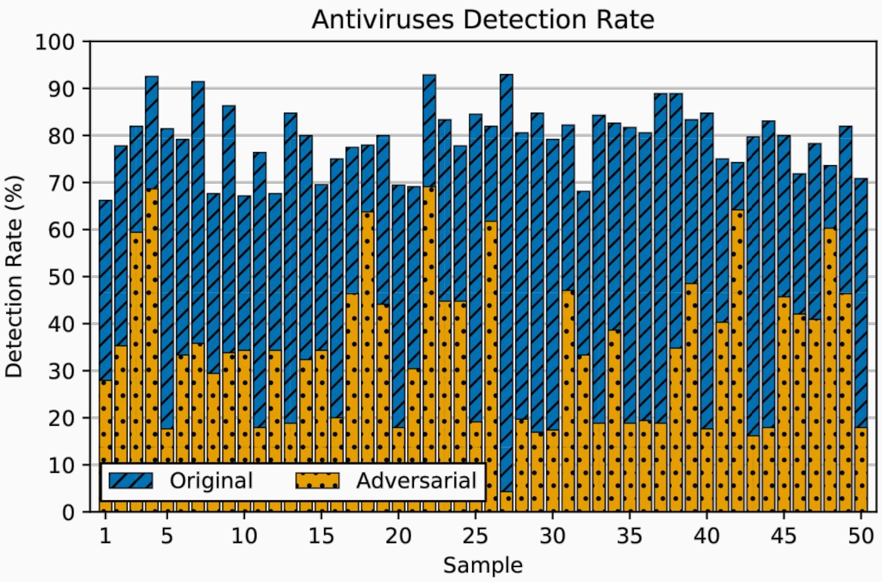
Our Attack Solution: Results

- **On average:** less than 5 queries per sample to bypass the three models
  - **Very low number:** even considering that we had previous knowledge about some models
  - **Expected from skilled and motivated attacker:** targeted attacks against real systems
- **Hold true for actual security solution:** 5 attempts is even below the threshold of a typical Intrusion Detection System (IDS)
  - Intrusion could occur unnoticeable

Team	Bypasses	Queries	Average
Ours	150	741	4.94
2nd	47	162	3.44
3rd	44	150	3,40
4th	1	78	78

Our Attack Solution: Impact on Real AVs

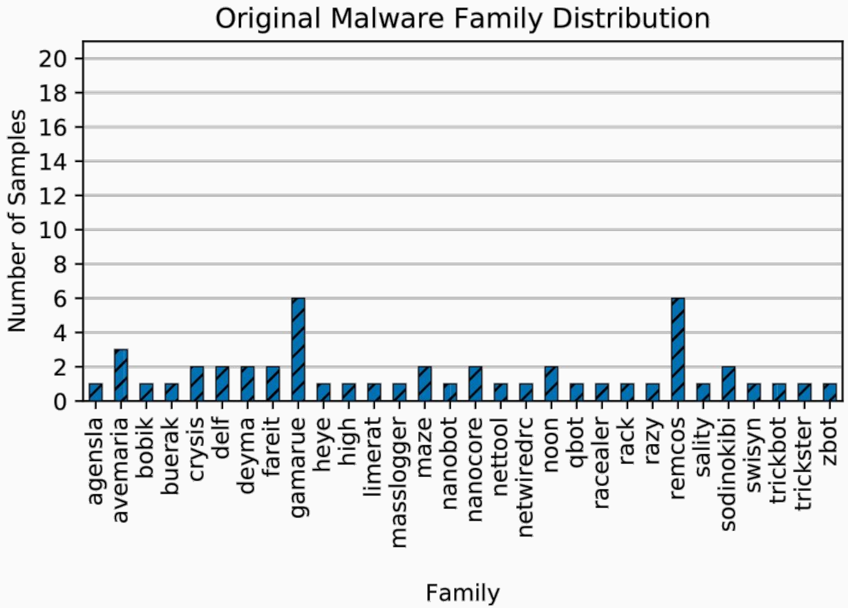
- **Virus Total detection rate:**
  - Original vs adversarial samples
- **AVS were also affected:**
  - Hiding payload from ML models also hides them from AV scans
  - ML models used by AVs are also affected by changes in binaries

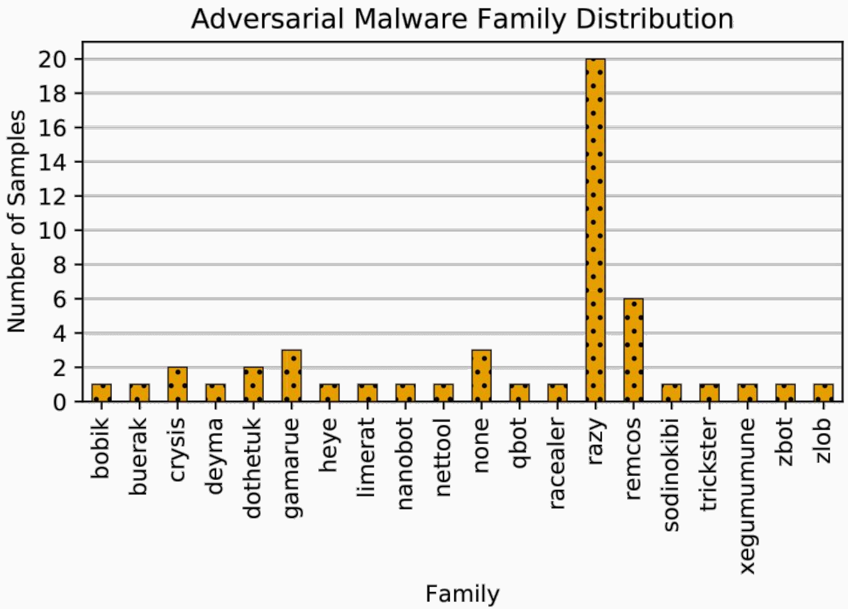


Our Attack Solution: ML and AntiVirus

Sample	Version	AntiVirus Detection				
		CrowdStrike	Cylance	Cynet	Elastic	Paloalto
22	Original	True (100%)	True	True (100%)	True (high confidence)	True
	Adversarial	True (60%)	True	False	False	False
27	Original	True (100%)	True	True (100%)	True (high confidence)	True
	Adversarial	False	False	False	True	False

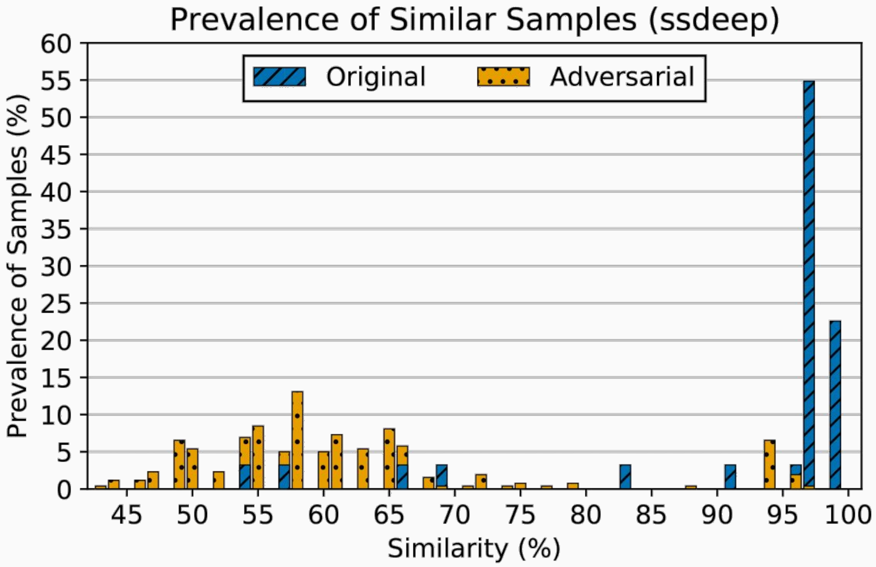
Our Attack Solution: Different Family Labels





Our Attack Solution: Side Effects

- **Dropper binaries become similar:** share same headers, instructions, libs
- **Using dropper:** increased the number of samples reported as similar
  - Reducing the relative frequency of very similar sample's scores
- **Dropper's similarities:** identified by the similarity matching solution
- **Similar bytes:** “diluted” among the dropper's bytes, reducing similarity

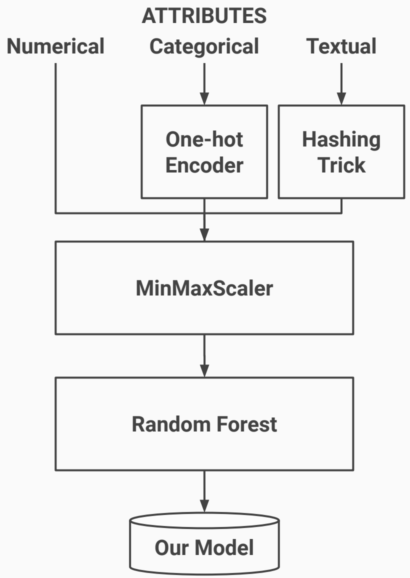


MLSEC 2021

2021

Defense Solution: Some Changes

- **Based on previous model:** improved some aspects
- **Removed features:** related to strings (number of paths, URLs, registry keys, and MZ headers)
- **More textual attributes:** *exports_list*, *dll_characteristics_list* e *characteristics_list* (from EMBER dataset)
- **New feature extractor:** HashingVectorizer
  - Features most resistant to attacks
  - **Online learning procedures (real-world solutions):** does not require updating the vocabulary as time goes by



[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.HashingVectorizer.html

Defense Solution: Testing with Adversaries*

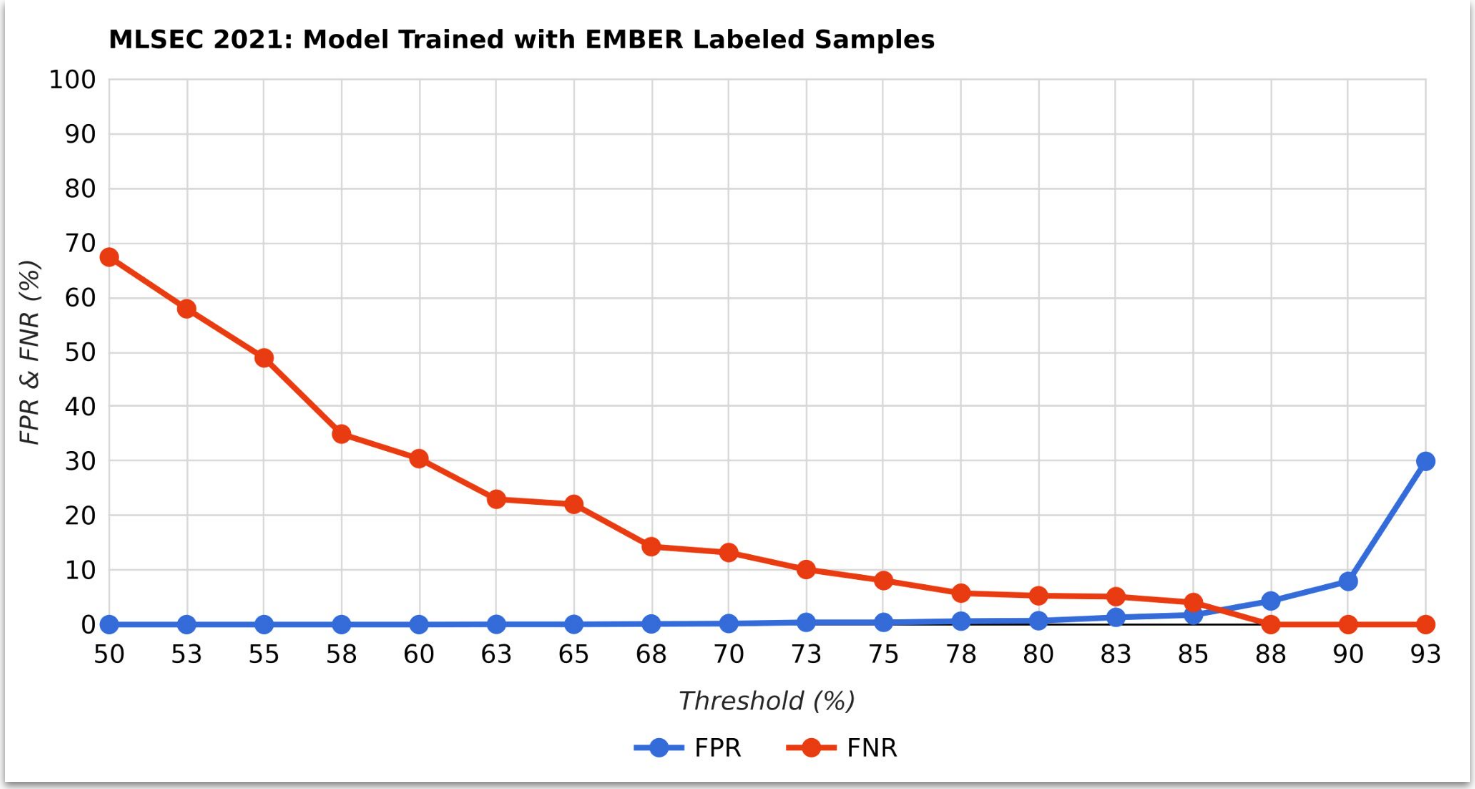
Model	F-Score	Recall	Precision
Last Year's Challenge (TF-IDF, 2020 Model)	0.62%	0.31%	**100%**
TF-IDF without String Features and with more Textual Features	20.86%	11.65%	**100%**
HashingVectorizer without String Features and with more Textual Features (2021 Model)	**43.12%**	**27.48%**	**100%**

*Tested using MLSEC 2019/2020 adversaries provided by organizers as malware, pristine Windows apps as goodware

Defense Solution: Tuning the Model

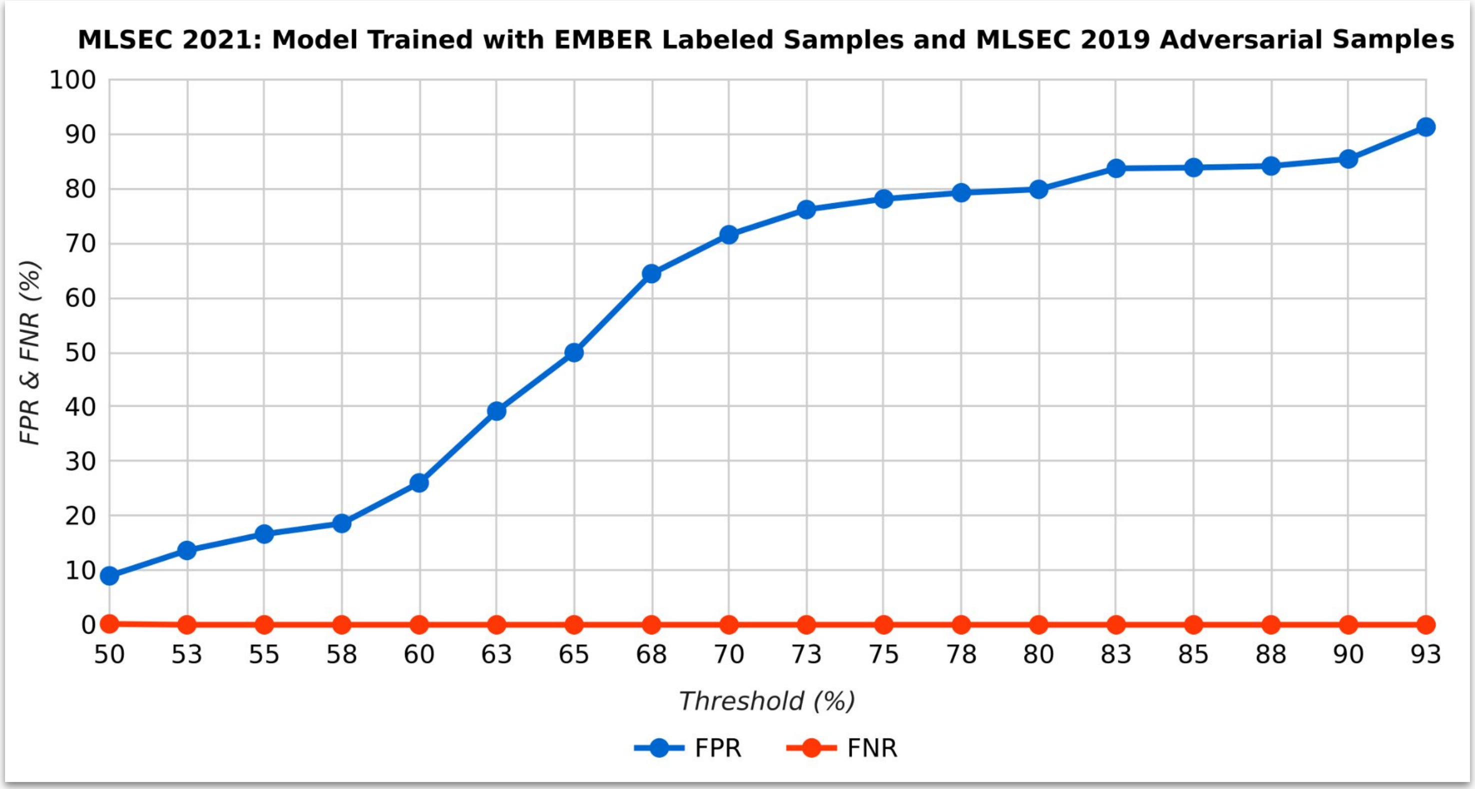
- **Considering two factors:**
  - **Training Dataset:**
    - EMBER labeled samples (~1mi)
    - EMBER labeled samples (~1mi) and MLSEC 2019 adversarial samples (594)
    - EMBER labeled samples and MLSEC 2020 adversarial samples (50 samples)
  - **Model Threshold T:** probability considered by the classifier to consider a given binary a goodware

Defense Solution: Tuning the Model



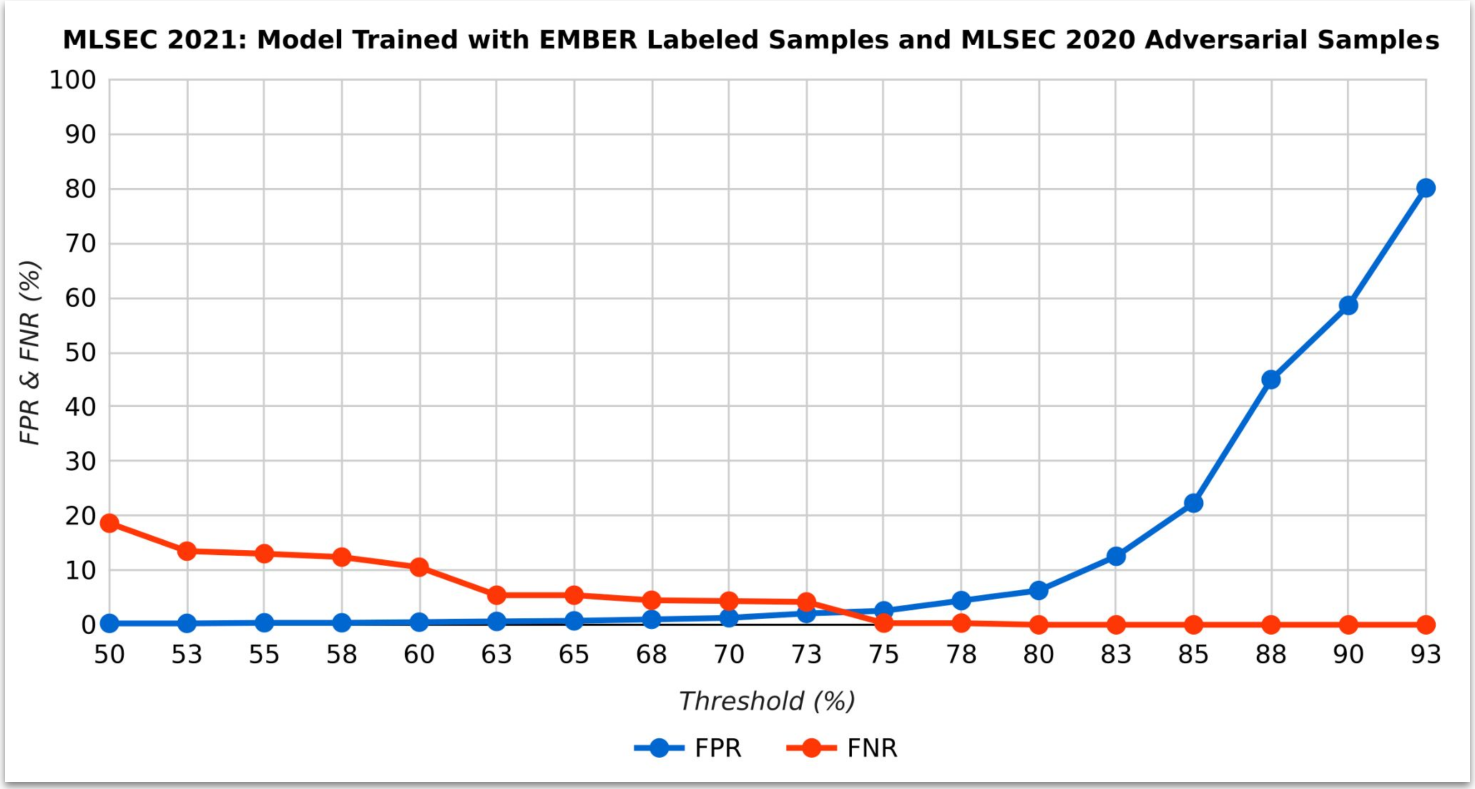
*Tested using MLSEC 2019/2020 adversaries provided by organizers as malware, pristine Windows apps as goodware

Defense Solution: Tuning the Model



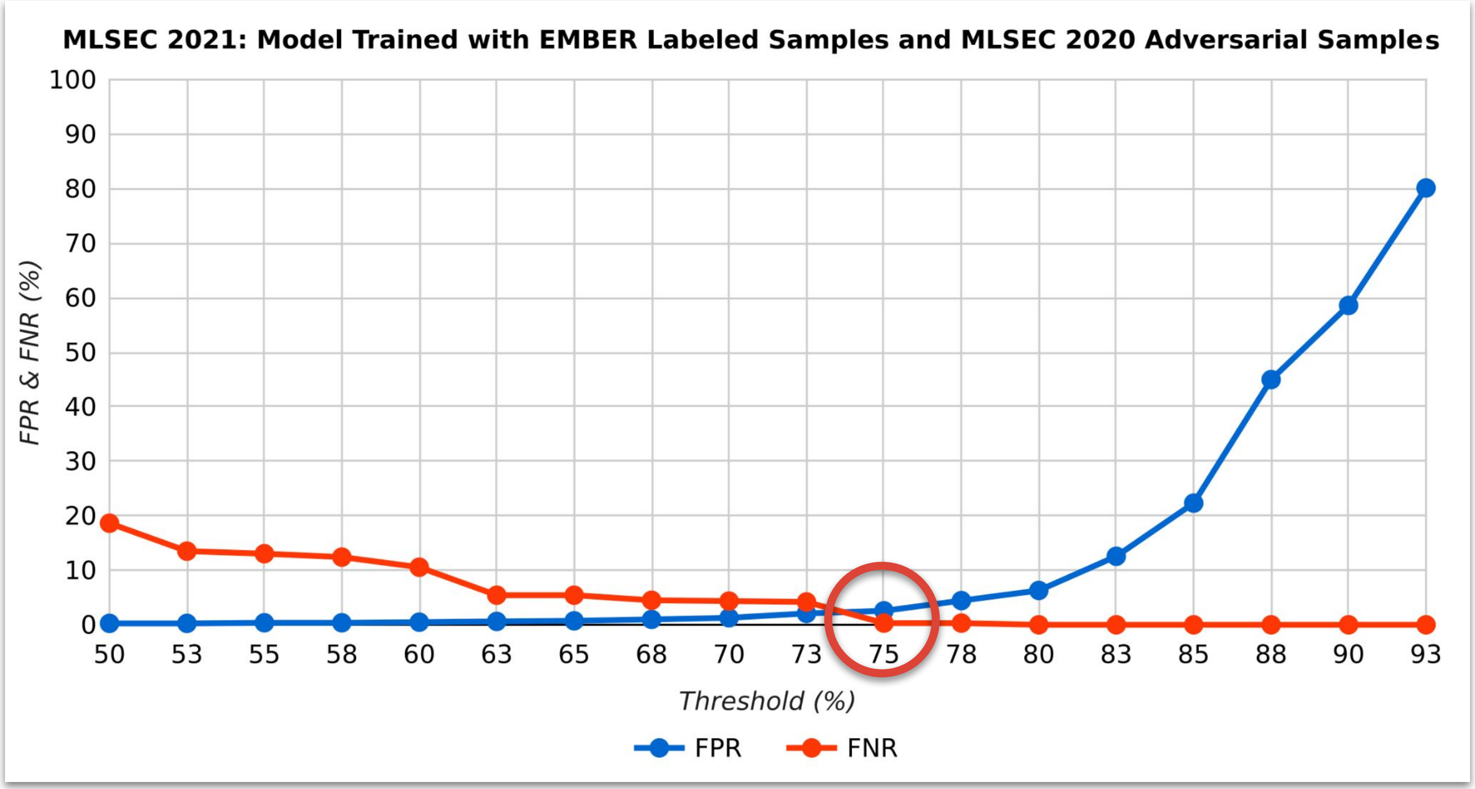
*Tested using MLSEC 2019/2020 adversaries provided by organizers as malware, pristine Windows apps as goodware

Defense Solution: Tuning the Model



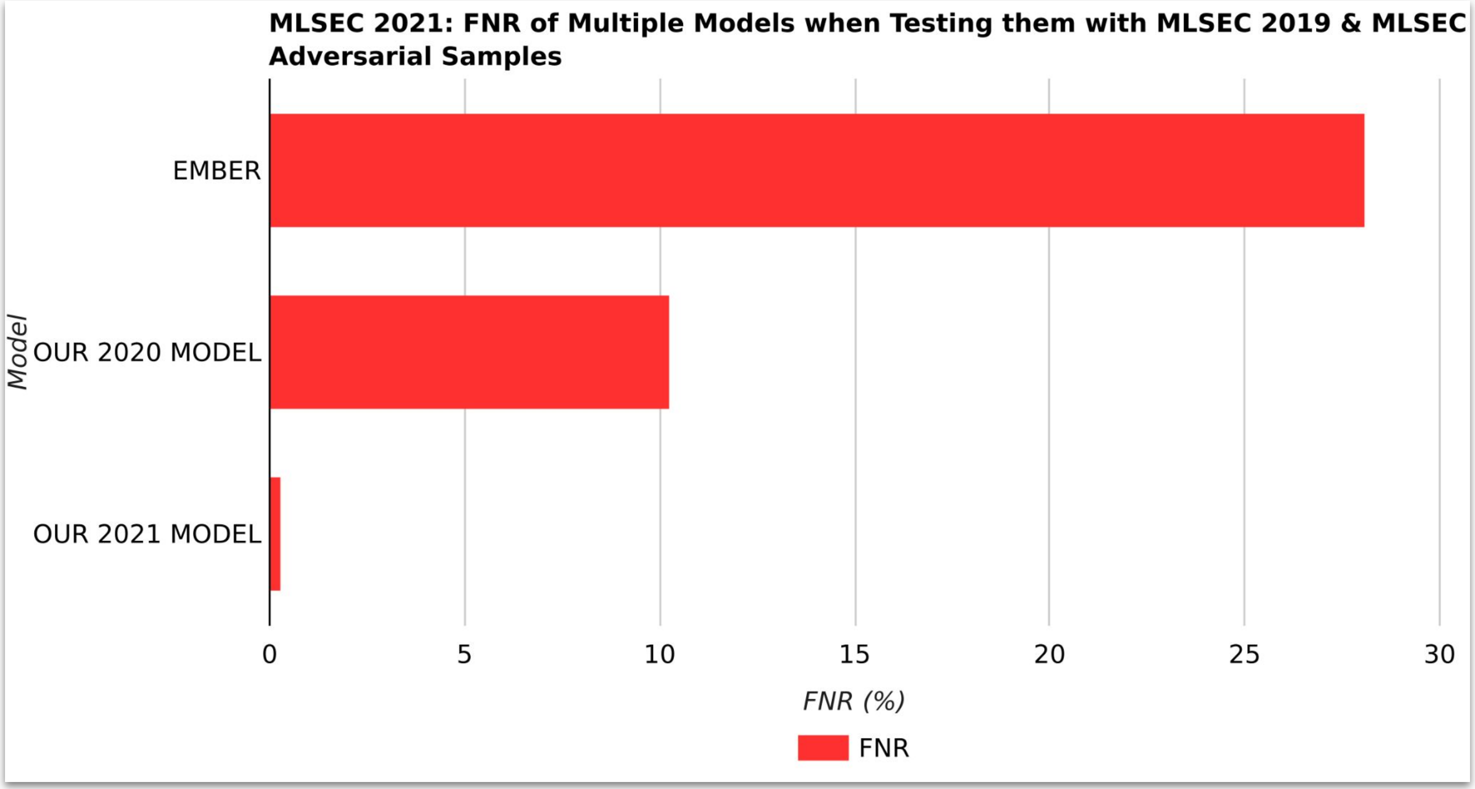
*Tested using MLSEC 2019/2020 adversaries provided by organizers as malware, pristine Windows apps as goodware

Defense Solution: Tuning the Model



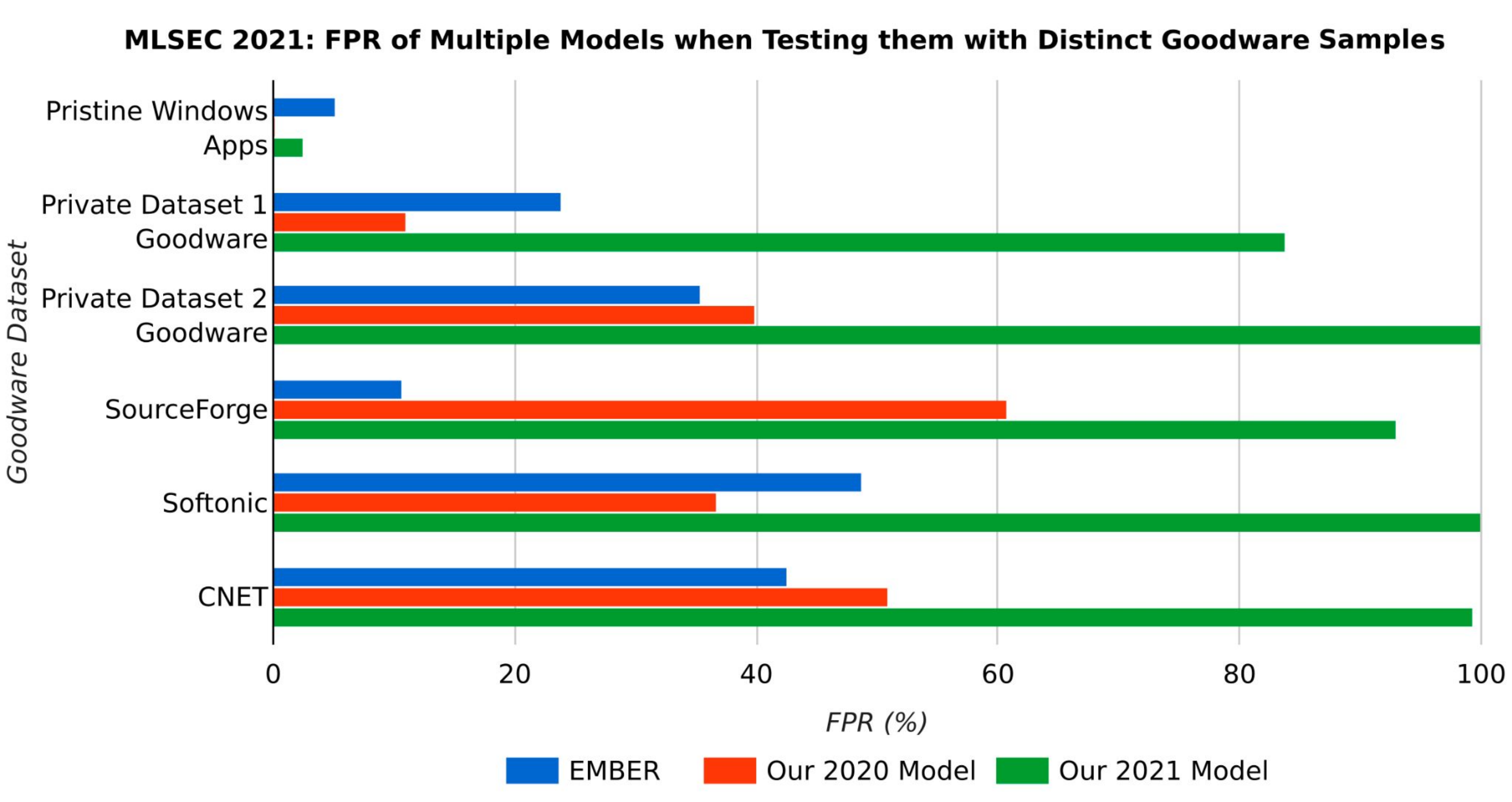
*Tested using MLSEC 2019/2020 adversaries provided by organizers as malware, pristine Windows apps as goodware

Defense Solution: Comparing Our Model



*Tested using MLSEC 2019/2020 adversaries provided by organizers as malware, pristine Windows apps as goodware

Defense Solution: Testing Multiple Models with Distinct Goodware Samples



Defender Challenge: Results

Model	# of Bypasses
secret (our model)	162
A1	193
kipple	231
scanner_only_v1	714
model2_thresh_90	734
submission 3	1840

Attack Solution: First Thoughts

- **Assumption:** bypassing our model would be enough to bypass the others
- **Problem:** didn't find any goodware sample with a significant number of imports classified as goodware to mimic
- **Native Windows NTDLL:** classified as goodware and had a significant number of exports
  - **Mimic it:** add fake exports with the same name as the ones from NTDLL to our dropper
- **Conversion:** dropper from EXE to DLL
- **New rule in 2021:** no filesystem dropping was allowed

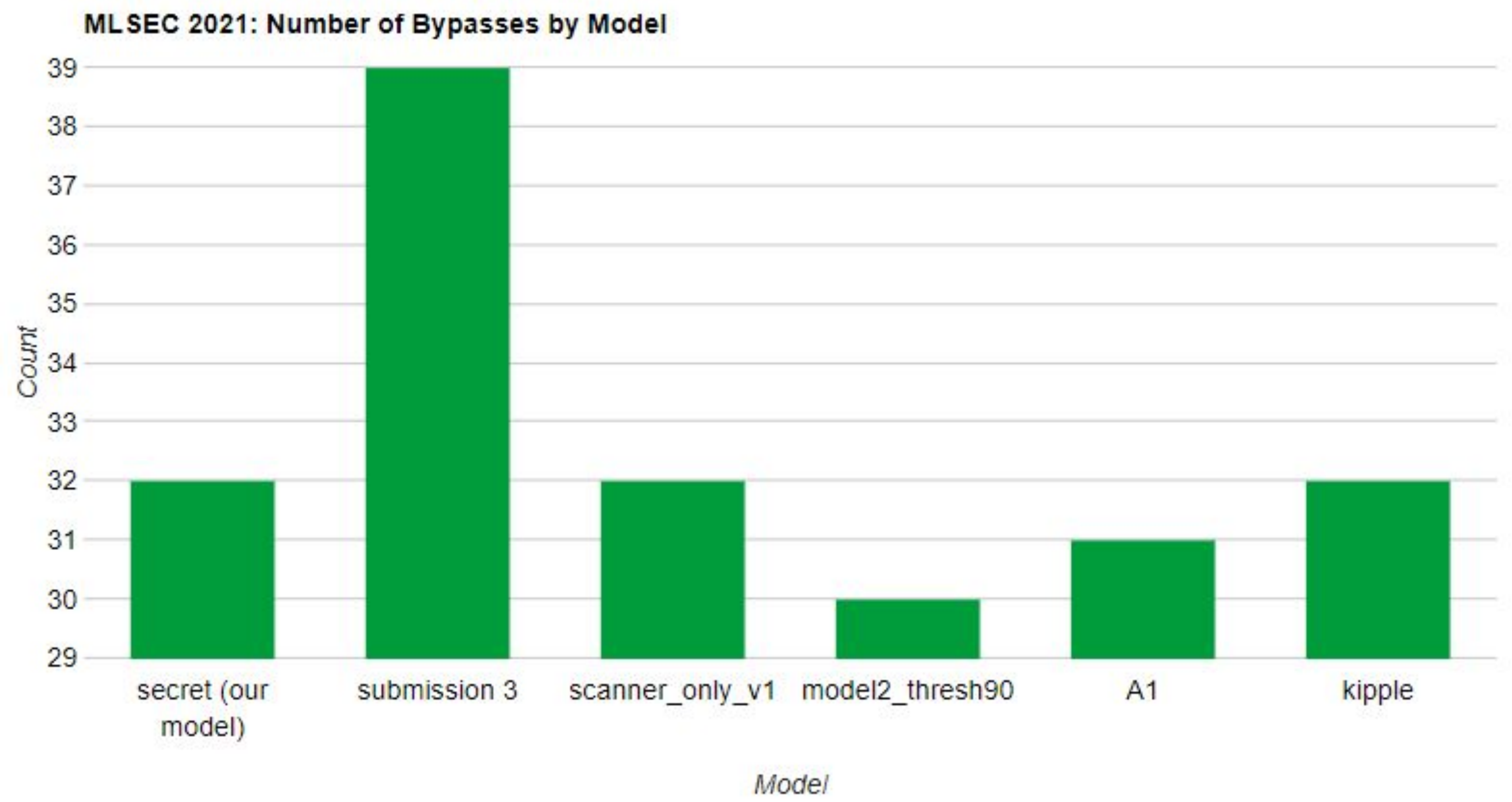
Attack Solution: Adapting our Solution

- **Filesystem to Memory:** memory-based approach (*RunPE* or *ProcessHollowing*), embedding encoded payload and extract it in memory
- **Problem with Sandbox:** *rundll32* process used to run our Dropper DLL doesn't work, even though it worked in our local machines
- **Solution:** a process that "likes" to be injected and patched
  - **Restriction:** it shouldn't be detected by the classifiers when dropped the disk
- **Another bias in our model:** .Net executables (mscoree library)
- **Hello World in .Net:** dropped file turned into malicious in run-time by injecting the original malware payload in it

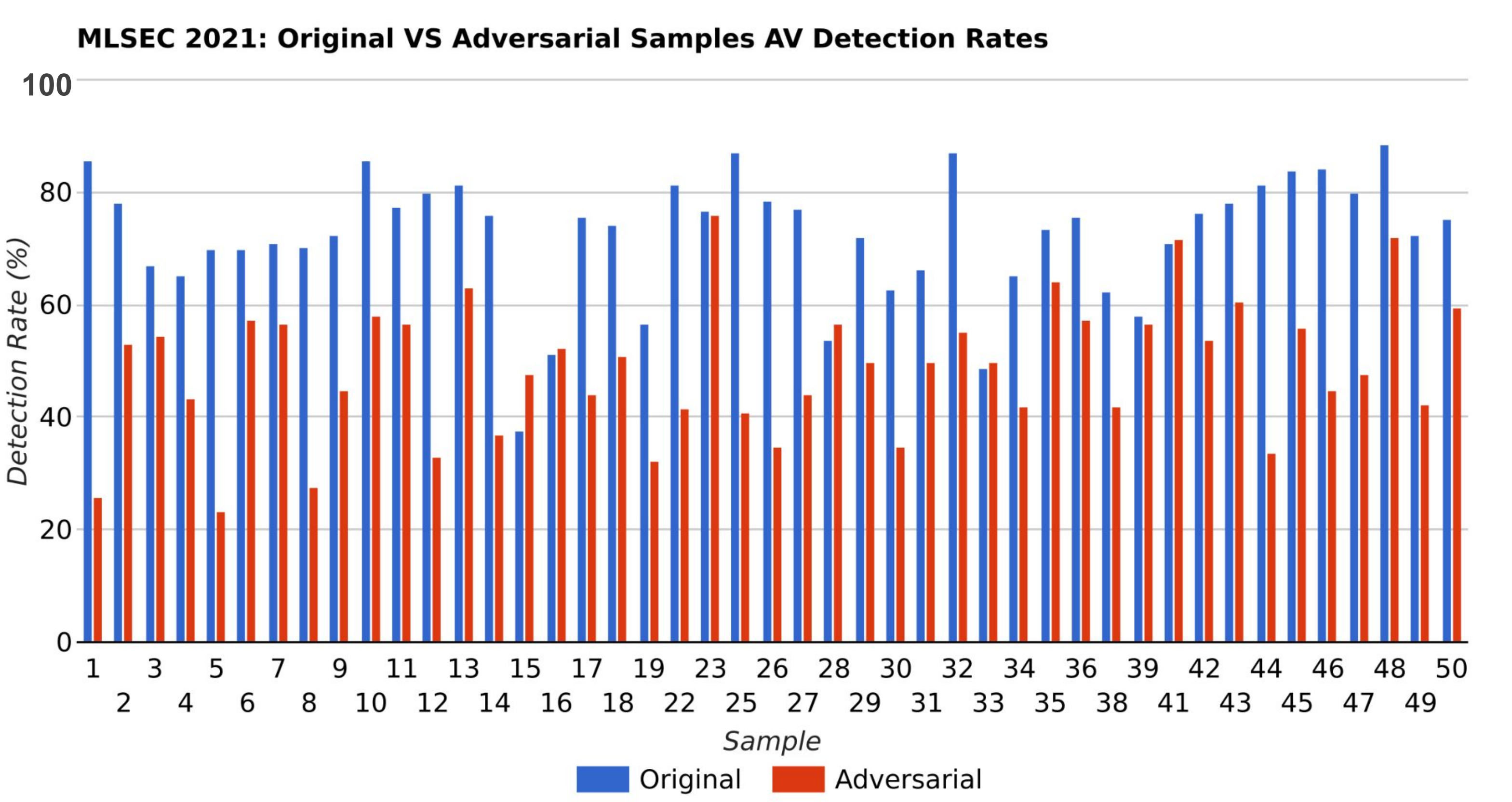
Attacker Challenge: Results

Nickname	Total Best Score per User	Total API Queries	Average
secret	196	600	3.06
amsqr	167	3004	17.98
rwchsfde	114	55701	488.61
vftuemab	113	3772	33.38
qjykdxju	97	3302	34.04
nomnomnom	86	14981	174.19
pip	74	534	7.21
dtrizna	68	4085	60.07
vxcuwzhg	13	108	8.31
fysvbqdq	12	773	64.41

Attack Solution: Number of Bypasses by Model



Attack Solution: Original VS Adversarial AV Detection Rates



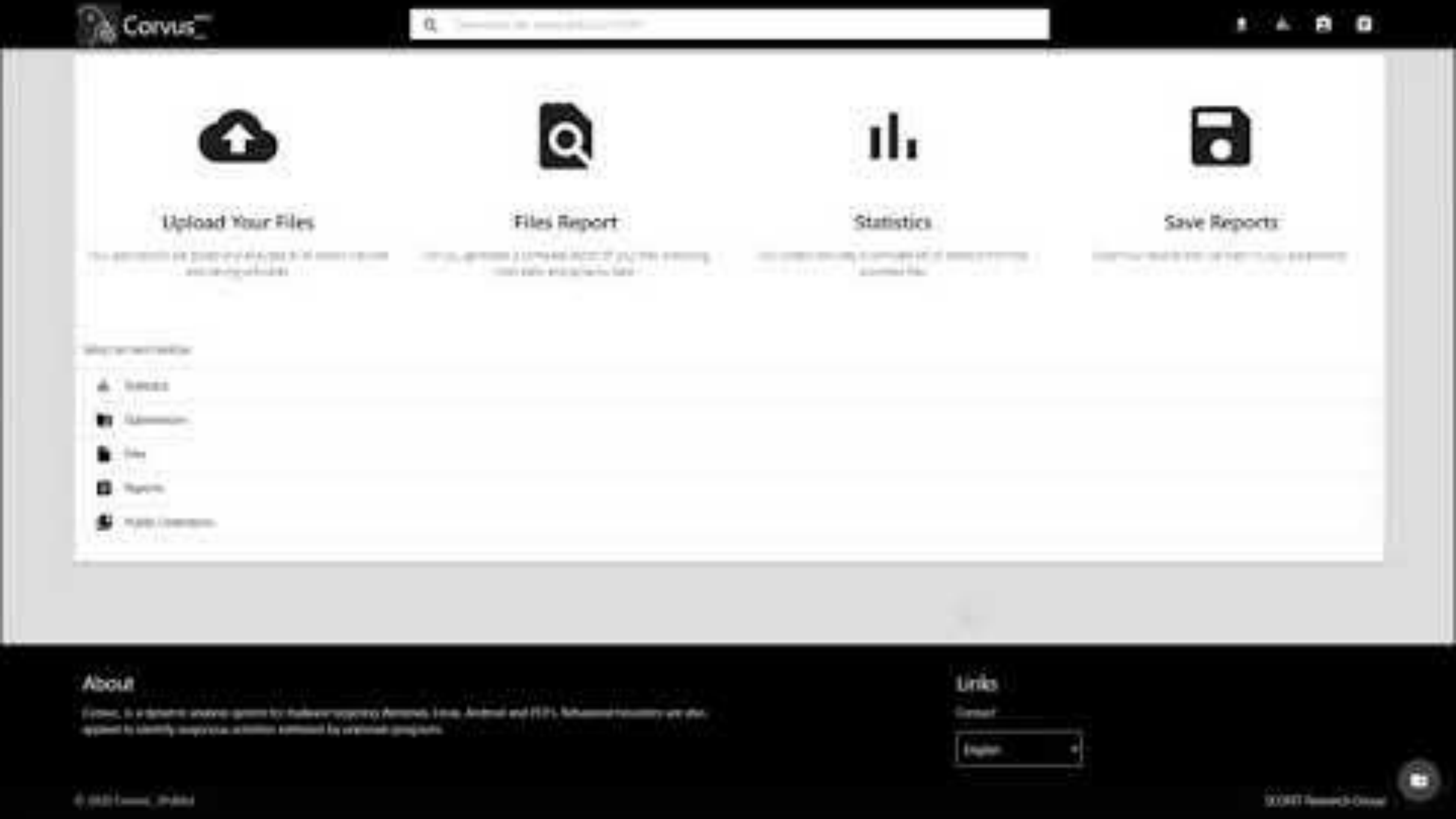
Conclusion

Adversarial Attacks for the Masses

- **Adversarial attacks:** really happen and are effective
    - Must be taken into account in threat models, datasets, and experiments
- **To encourage this practice:** publicly released our codes to the community
    - Anyone may be able to practice with them (and improve them a **LOT**!)
    - Consider adversarial attacks in their own research
- **Web-based solution[1]:** generate adversarial samples with our method
    - **Each submitted file:** tested in multiple ML models
- **Check:** robustness of multiple models and viability to attack them

corvus.inf.ufpr.br

[1] https://corvus.inf.ufpr.br


Corvus_
Upload Your Files
Files Report
Statistics
Save Reports
About
Links

Feedback for Future Work

- **Our findings:** valuable feedback for next-gen security solutions
  - **Embedding payloads into binary:** simple yet effective way to defeat classifiers
- **Next-generation solutions:** cannot be limited to look only into the first binary layer
  - Extract embedded payloads (e.g., via file carving) to classify them
- **Change features representation:** cover less mutable features
  - “Features need to be discriminative and **invariant**”

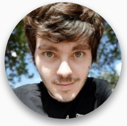
Reproducibility: Everything is Open-source!

MLSEC 2020: Need for Speed Malware Detection Model

Created by fabriciojoc

Source code of our detection model

1 FORK 4 STAR https://github.com/fabriciojoc/mlsec2020-needforspeed



2021 Machine Learning Security Evasion Competition

Created by fabriciojoc

Our 2021 Machine Learning Security Evasion Competition source code

1 FORK 6 STAR https://github.com/fabriciojoc/2021-machine-learning-security-evasion-competition

Dropper

Created by marcusbotacin

Source code of the developed dropper

2 FORK 9 STAR https://github.com/marcusbotacin/Dropper

Our Papers Related to this Work: We are Open to Collaborations!

- Ceschin, F., Pinage, F., Castilho, M., Menotti, D., Oliveira, L. S. and Grégio, A. (2018). The need for speed: An analysis of brazilian malware classifiers. IEEE Security & Privacy.
- Ceschin, F., Botacin, M., Gomes, H. M., Oliveira, L. S. and Grégio, A. (2019). Shallow security: On the creation of adversarial variants to evade machine learning-based malware detectors. Proceedings of the 3rd Reversing and Offensive-Oriented Trends Symposium, ROOTS’19, Vienna, Austria.
- Botacin, M., Ceschin, F., de Geus, P. and Grégio, A. (2020b). We need to talk about antiviruses: Challenges & pitfalls of av evaluations. Computers & Security.
- Ceschin, F., Botacin, M., Lüders, G., Gomes, H. M., Oliveira, L. S. and Grégio, A. (2020). No need to teach new tricks to old malware: Winning an evasion challenge with xor-based adversarial samples. Proceedings of the 4th Reversing and Offensive-Oriented Trends Symposium, ROOTS’20, Vienna, Austria.
- Botacin, M., Ceschin, F., Sun, R., Oliveira, D., Grégio, A (2021). Challenges and Pitfalls in Malware Research. Computers & Security, pp. 102287, 2021, ISSN: 0167-4048.
- Ceschin, F., Gomes, H. M., Botacin, M, Bifet, A., Pfahringer, B., Oliveira, L. S., Grégio, A. (2021). arXiv:2010.16045

ML SEMINAR
OCTOBER 13/14TH 2021
SECRET.INF.UFPR.BR

All You Need to Know to Win a Cybersecurity Adversarial Machine Learning Competition and Make Malware Attacks Practical

Fabrício Ceschin
Federal University of Paraná, BR
@fabriciojoc
fjoceschin@inf.ufpr.br

Marcus Botacin
Federal University of Paraná, BR
@MarcusBotacin
mfbotacin@inf.ufpr.br