sAIfer Lab

Joint lab on Safety and Security of AI

Pick Two:
Robustness, Accuracy, Generalization in Malware Detection with AI

Luca Demetrio
Assistant Professor @ UniGe

UNICA UNIVERSITÀ DEGLI STUDI DI CAGLIARI

Università di Genova

$ whoami

Luca Demetrio

Member of sAIfer Lab
Assistant Professor @ UniGe
Associate Editor for **Pattern Recognition**
Developer of **SecML Malware**
BlackHat Course Trainer in 2024
Reviewer for top conferences and journals
Involved in EU funded research projects (and more)

Contacts
Mail: luca.demetrio@unige.it
X: @zangobot
Bluesky: @zangobot.bsky.social

Malicious programs by design

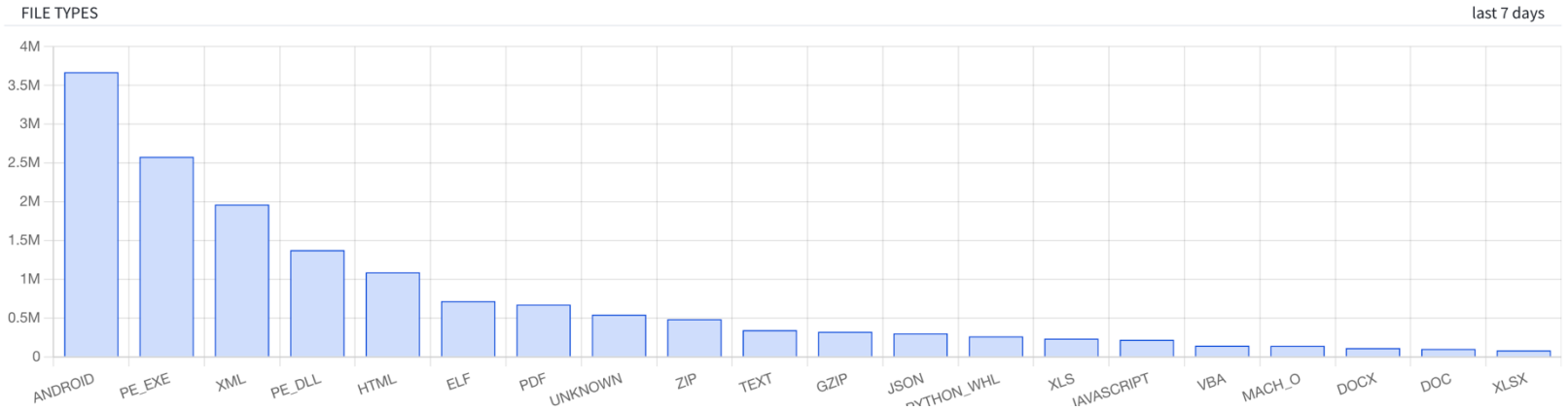
Malicious by Design

Intrusive programs with harmful intents: causing damage, stealing credentials, ransom, take control, open backdoors for future access, and all the bad things you could think of

Business of Malware

Active groups that sell malware as a service, develop variants, receive reviews, etc.

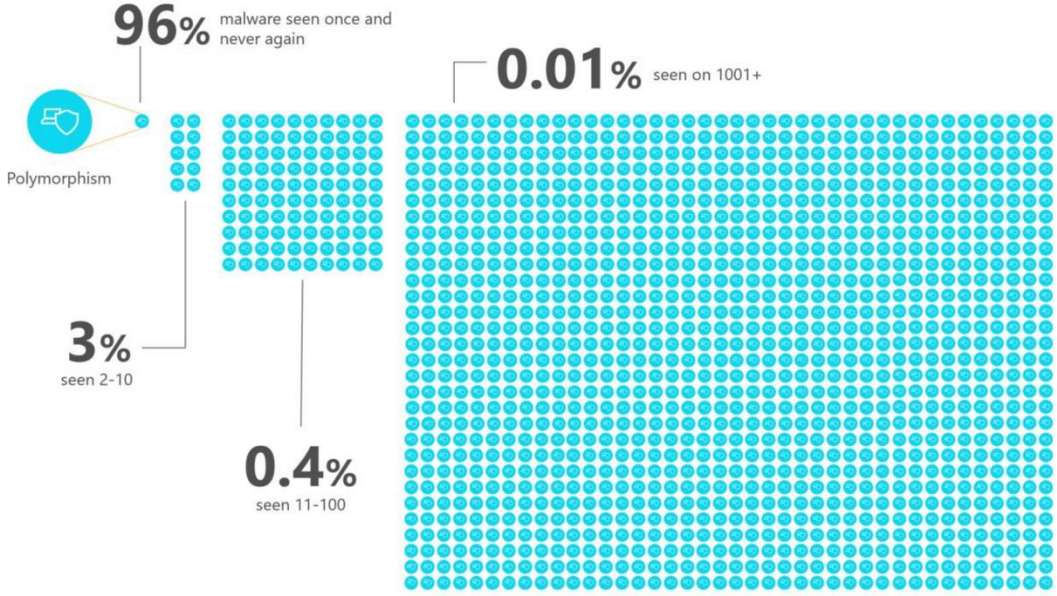
Alert fatigue



2.5 Million per Week samples sent to VirusTotal to be scanned by online antivirus programs
Not all of them are malware, but plenty of resources are needed

(Interesting trend, Android was second until few months ago)

Limitation of Signatures



Evolution of Malware Prevention
[https://info.microsoft.com/rs/157-GQE-382/images/Windows%2010%20Security%20Whitepaper.pdf]

Too many variants!

Block/Allowlist can work ONLY if threats are known, but the majority of them (96%) appear in the wild only ONE time

Machine Learning joins the fight

Filter out known threats, generalize to variants
Machine learning models learn "signatures", and they can spot variants of the same malware, filling the blanks of signature-based detection

Spreading into commercial products
Companies claim to use machine learning technologies inside their detectors to spot Windows malware by learning patterns from data

All problems solved, right?

SOPHOS
Security made simple.

Intercept X Deep Learning

CrowdStrike Introduces Enhanced Endpoint Machine Learning Capabilities and Advanced Endpoint Protection Modules

— Company continues to accelerate pace of replacement of legacy AV solutions in both enterprise and SMB markets —

kaspersky

Infamous example: Cylance

Injecting bytes
Reversing the code with some tricks, discovered that the model leverages STRINGS

Inject "benign" values
Extract byte sequences from "Rocket League" and include them inside input exacutable

Evasion completed!
The company rolled out an update to try to mitigate the issue, but this write-up shed the light on robustness issues of ML-based detectors

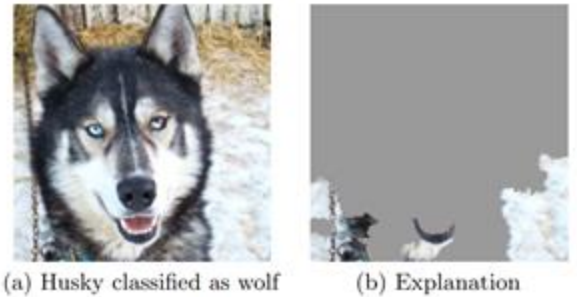
[Skylight Cyber. Cylance, I Kill You! https://skylightcyber.com/2019/07/18/cylance-i-kill-you/]

Infamous example: spurious correlations

Focusing on irrelevant locations
ML might takes decision based on biases contained in data, such as the background, or artifacts that were not filtered at training time

Spurious correlations
If an husky has always snow behind it, the ML model will associate "snow" to "husky", and everytime an image has "snow", it will be labelled as "husky". This is not the intended behavior

(Spoiler: this happens to malware detection as well, brace with me for some slides)



(a) Husky classified as wolf (b) Explanation

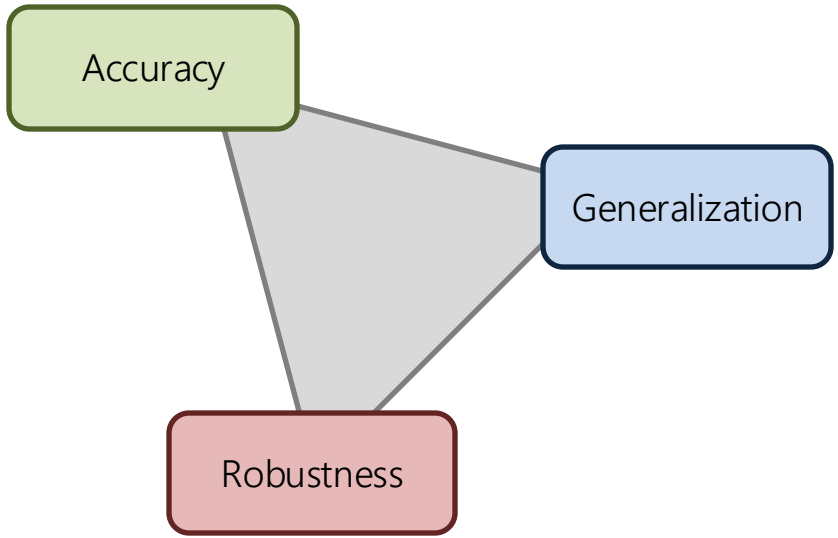
Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

It is time to choose

It is always complicated to gain ALL the desired properties, due to the complicated setting of training from complex data

Accurate and good generalization, but not robust; or ULTRA accurate but overfitting; or very robust and avoid overfitting, but not as accurate

Let's deep dive into this conondrum



How to Build Accurate Models

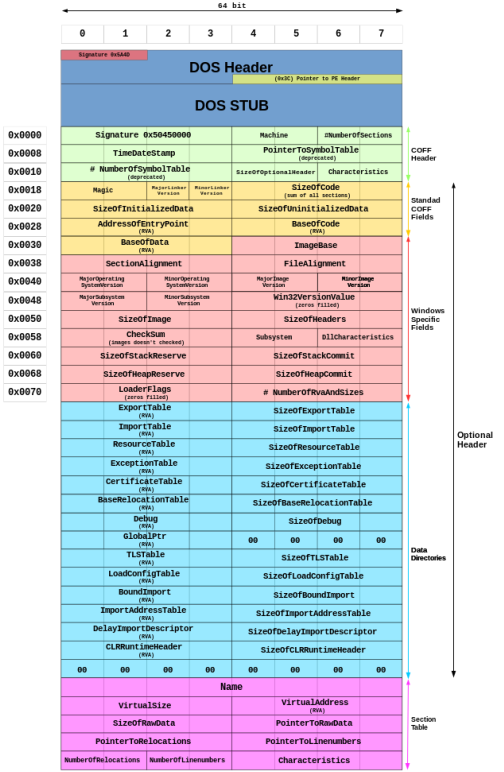
Dealing with programs as training data

Programs stored as file
Each program is a regular file that can be analysed without executing it, stored as file compliant to the Portable Executable (PE) format

Two roads ahead to feed PE files to ML models:

1) Extract features: analyse the file through static analysis, retrieve information such as imported and exported APIs, metadata of headers, entropy, length of code, and much much more

2) End-to-end learning
Forget about preprocessing, use each byte of the program/resources/files as token, let the ML models learn by themselves what is what

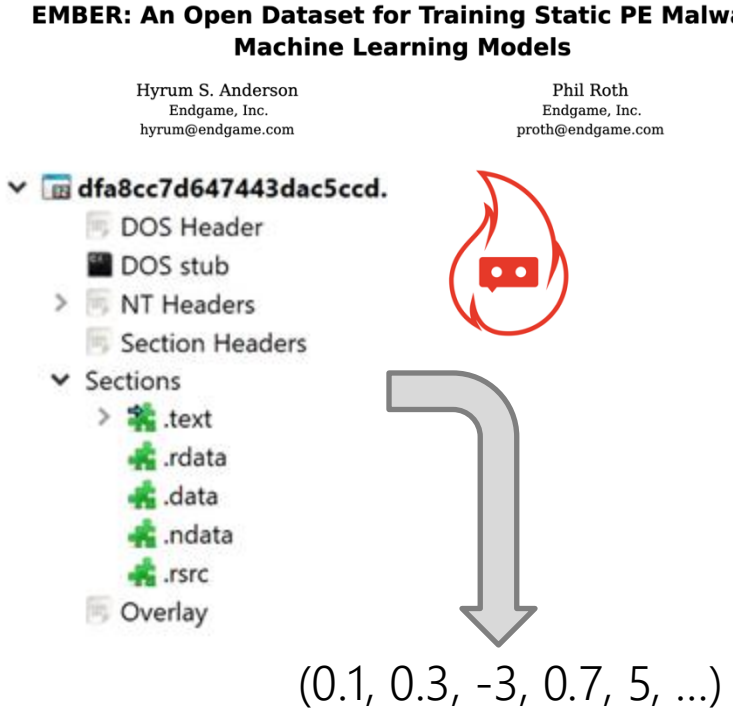


First road: Extracting features from PE files

Aggressive preprocessing
Salvage all the relevant information, given deep domain knowledge of the file format (which has many components), and store them as numbers

State of the art features
We are currently relying on the same feature sets developed in 2017 by people at ~~Endgame~~ Elastic ~~Robust Intelligence~~ CISCO

Good for accuracy and generalization
The feature set is mature, used by plenty of research work (671 citations on scholar), and detectors trained on EMBER exhibit great performance on unseen data



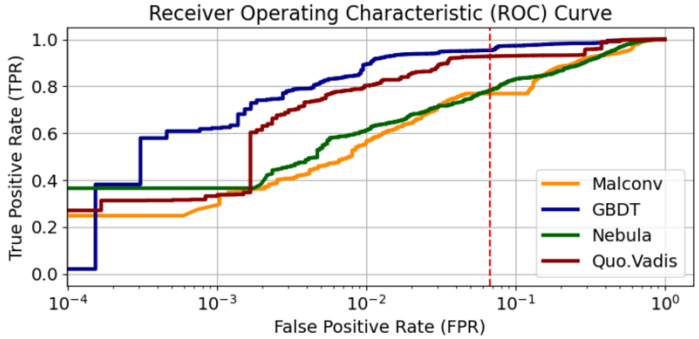
Unmatched performance with shallow models

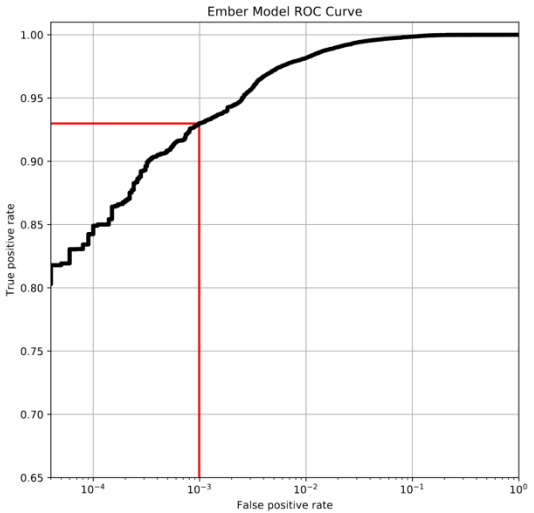
Most used model in literature: tree ensembles

Non-linear decisions that adapt very well on the complex features extracted by EMBER

Excellent accuracy on unseen data, impossible to compete against in deployment settings: very low FPRs with incredibly high TPR

We have already a clear winner, right?





Limitations of feature extractors

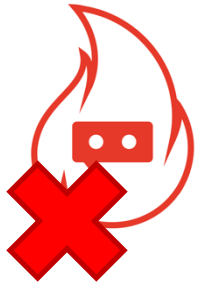
What is this thing?

Once a sample is processed as a feature vector, there is no way to get back to the original PE file. Why there is a 5 there? Is it possible to understand the effect of a minimal change in the whole feature vector? **Not really!**

Pre-processing errors

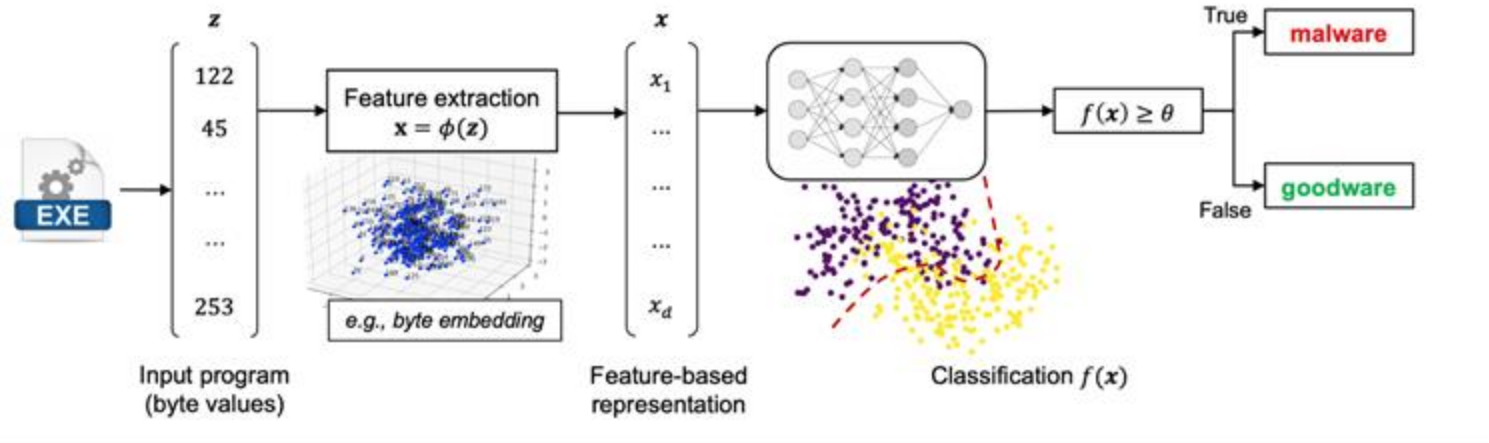
What happens when the EMBER pre-processing fails? Can you analyze the sample? What should you do on an endpoint?

It is not unrealistic that malware samples (but also legitimate programs) crash the analysis process!



Ponte et al. SLIFER: Investigating Performance and Robustness of Malware Detection Pipelines, COSE 2025

Second road: learning from raw bytes



Raw bytes are features

Set maximum amount of bytes per sample, and feed them **AS-IS** to deep neural networks

These will automatically learn an abstract feature representation internally, which act as a feature extractor (but in fact, it is just sum and products)

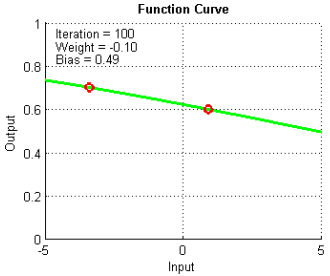
No need for domain knowledge, and impossible to crash since there is no pre-processing involved!

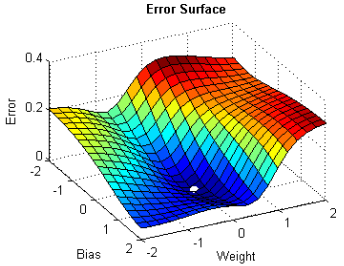
Achieved through Gradient Descent (and GPUs)

Iteratively adapt model to data
Unlike other models, deep neural networks adapt their thousands / million of parameters with **gradient descent**, by minimizing the error at each iteration

GPU-friendly training
Neural networks can be accelerated (both at training and test time) with GPUs, using millions of samples, reaching the similar performances of feature-based models (with zero insights on domain knowledge)

Each byte matters
Since it is end-to-end, it is possible to characterize the relevance of each byte, correlating it with the output





Malware Detection by Eating a Whole EXE

Edward Raff[1,3,4], Jon Barker[2], Jared Sylvester[1,3], Robert Brandon[1,3,4]
Bryan Catanzaro[2], Charles Nicholas[4]
[1]Laboratory for Physical Sciences, [2]NVIDIA, [3]Booz Allen Hamilton, [4]University of Maryland, Baltimore County
{edraff,jared,rbrandon}@lps.umd.edu, {jbarker,bcatanzaro}@nvidia.com, nicholas@umbc.edu

Activation Analysis of a Byte-Based Deep Neural Network for Malware Classification

Scott E. Coull
FireEye, Inc.
scott.coull@fireeye.com

Christopher Gardner
FireEye, Inc.
christopher.gardner@fireeye.com

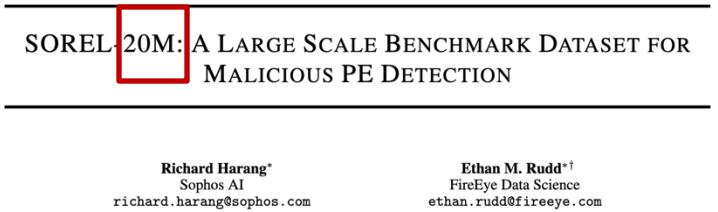
Limitations of end-to-end networks

What is this learning?

Not having domain knowledge requires the network to look at the data multiple times before extracting meaningful patterns, elevating the resources to use at training time

"How many samples do you need? Yes."

Neural networks need gazillion of training samples to converge to a minimum of the function to optimize, which might be difficult to achieve (most of the legitimate programs to be used at training time are unavailable or under paywall)



SOREL-20M: A LARGE SCALE BENCHMARK DATASET FOR MALICIOUS PE DETECTION

Richard Harang*
Sophos AI
richard.harang@sophos.com

Ethan M. Rudd*†
FireEye Data Science
ethan.rudd@fireeye.com

Recap

Two major approaches to build accurate models

99% accuracy, almost 0% FPR, life is good

But what happens when these detectors are under attack? Did they really learn something useful from data?

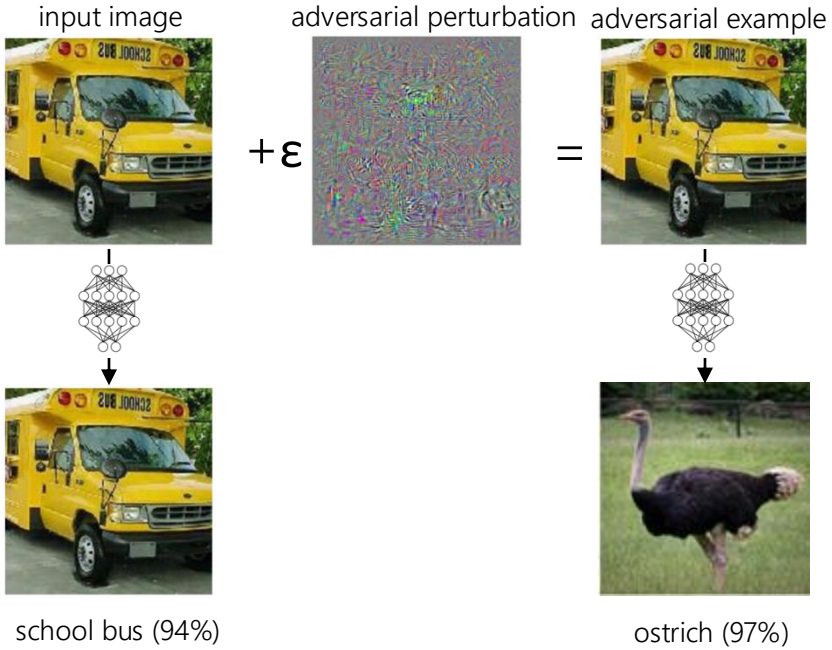
Loss of Robustness

Robustness against minimal manipulations

Machine learning models can be fooled by minimally-messing with input data
(You probably saw this slide hundreds for times by now)

Adversarial Examples can be crafted against models in different ways, depending on the threat model (how much is known of the target, what an attacker can tamper and what not)

In Windows malware detection we have **Adversarial EXEmples** which leverage the same concepts but in the domain of Windows programs



(Thanks to Battista Biggio, Maura Pintor for these slides!)

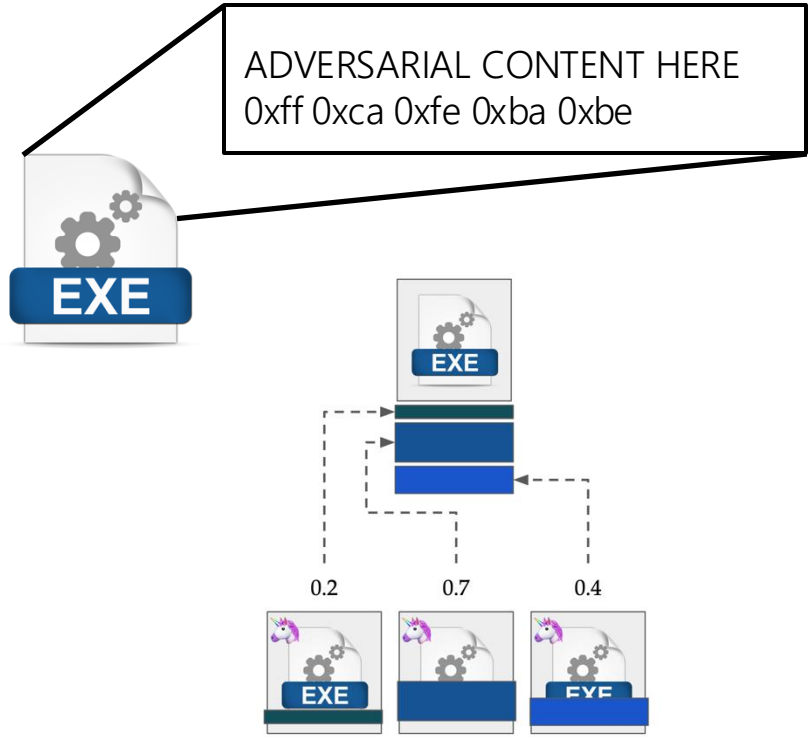
Szegedy et al., Intriguing properties of neural networks, ICLR 2014
Biggio, Roli, et al., Evasion attacks against machine learning at test time, ECML-PKDD 2013

Adversarial EXEmples

Mess with the PE format
Not everything is needed: plenty of metadata and locations inside PE files are useless or easily twisted by attackers

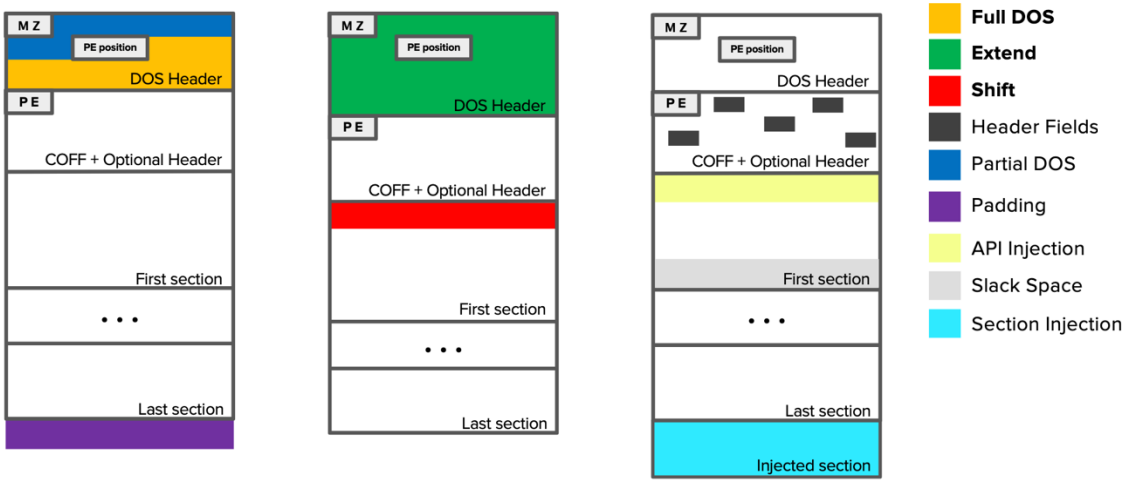
"Windows is the toyland for malware"
- Cit.

Inject or replace content
Attackers can either replace useless content (such as entire headers) or create new space to contain new byes



Demetrio et al., Functionality-preserving Black-box Optimization of Adversarial Windows malware, IEEE TIFS 2021

A world of pure manipulations

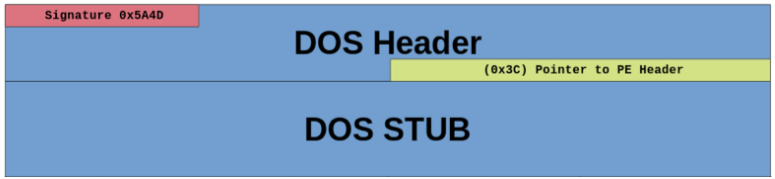


A quick recap on some of bad things that attackers can apply to PE files in order to fool machine learning models

Notable example: DOS header

Each PE file contains a DOS program stored for retrocompatibility (really? O_O)

Content is loaded in memory but not executed, and almost all is attacker controlled (the blue part)



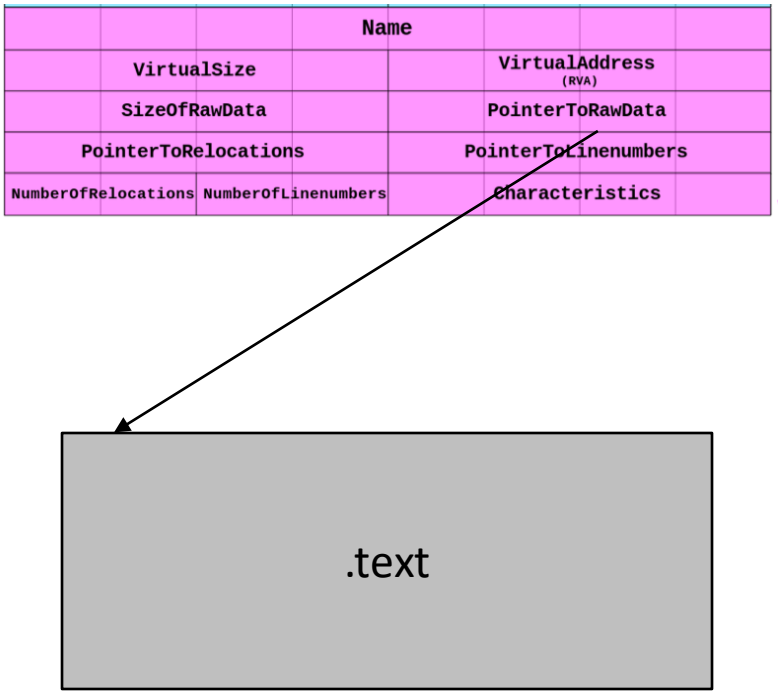
Demetrio et al., Adversarial EXEmples: a Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection, ACM TOPS 2021

Content shifting

Meaningful content (code, data, etc) is stored in Sections, which are referred by a table that points to the content

It is possible to exploit these information to reserve space for the attack

NOT LOADED IN MEMORY!



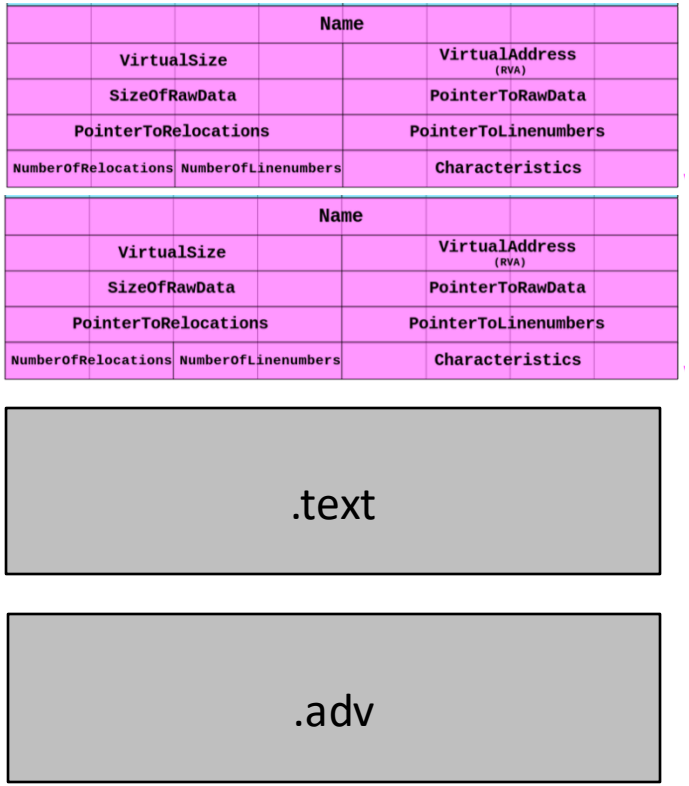
Demetrio et al., Adversarial EXEmples: a Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection, ACM TOPS 2021

Section Injection

Manipulate section table to add new entry

Append chunk of bytes, referenced by newly added entry

Loaded in memory or not, depending by the characteristics set up inside the entry

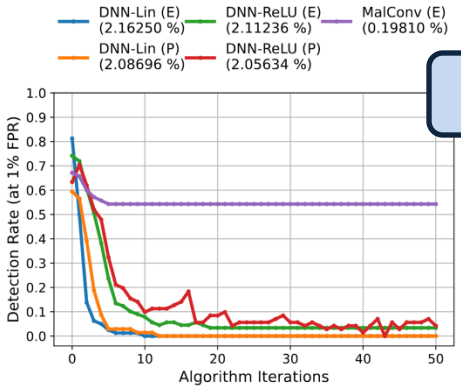


Demetrio et al., Adversarial EXEmples: a Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection, ACM TOPS 2021

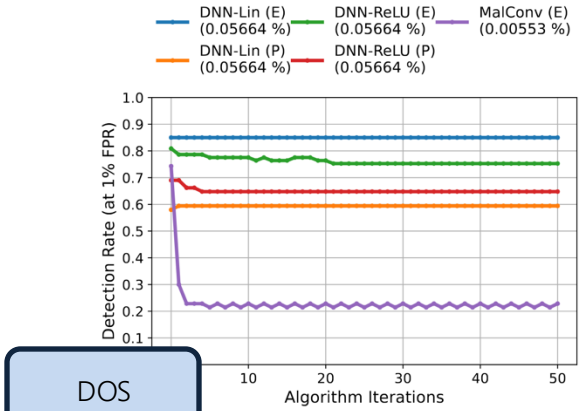
Goodbye robustness for end-to-end models

Adversarial manipulations disrupt the sequence of bytes seen at training time

Hence, **end-to-end detectors** are **flatlined** by many of these attacks, as all the correlations are compromised by the replacement or inclusion of novel content



Content Shift



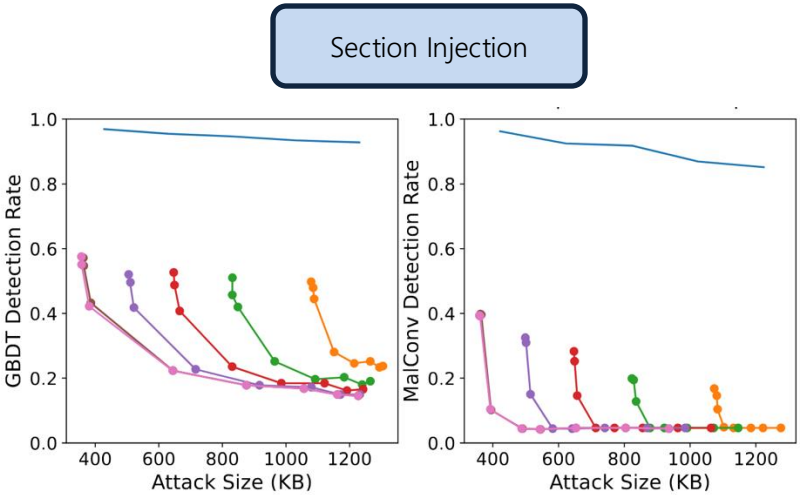
DOS

Demetrio et al., Adversarial EXEmples: a Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection, ACM TOPS 2021

Goodbye (?) robustness for feature-based models

More robust, since previous replacement attacks are not working at all thanks to the preprocessing, which strips away most of the manipulation

However, **injecting content as new sections**, harvested from regular programs, flatlines shallow models trained on features as well



Demetrio et al., Functionality-preserving Black-box Optimization of Adversarial Windows malware, IEEE TIFS 2021

Deployed systems can be influenced as well

Commercial antivirus programs have to deal with these attacks, the same that flatlined the shallow models (that in fact are crafted against them)

No need to know those, just compute attacks on a strong model you possess, and send the effective adversarial EXEmples against your ~~favorite~~ antivirus

	Malware	Random	Sect. Injection
AV1	93.5%	85.5%	30.5%
AV2	85.0%	78.0%	68.0%
AV3	85.0%	46.0%	43.5%
AV4	84.0%	83.5%	63.0%
AV5	83.5%	79.0%	73.0%
AV6	83.5%	82.5%	69.5%
AV7	83.5%	54.5%	52.5%
AV8	76.5%	71.5%	60.5%
AV9	67.0%	54.5%	16.5%

Demetrio et al., Functionality-preserving Black-box Optimization of Adversarial Windows malware, IEEE TIFS 2021

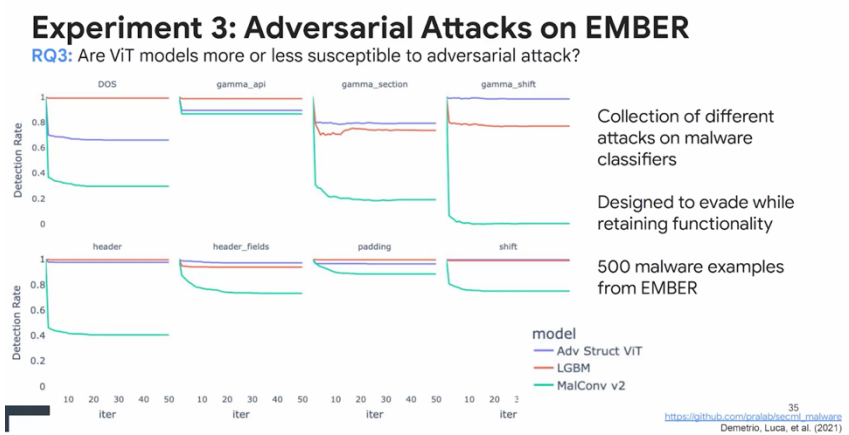
Take-home message on robustness

1. End-to-end models can be easily flatlined by simple attacks that replace or add content, since they can not find anymore the patterns they have learned ad training time

2. Shallow models are more robust, but whether the attacker changes manipulation it is likely that performance will drop consistently

3. You can test these with the library we are developing and maintaining:
   SecML Malware
   (https://github.com/pralab/secml_malware)

4. I gladly discovered that our attacks are also used by Google Cloud ^.^

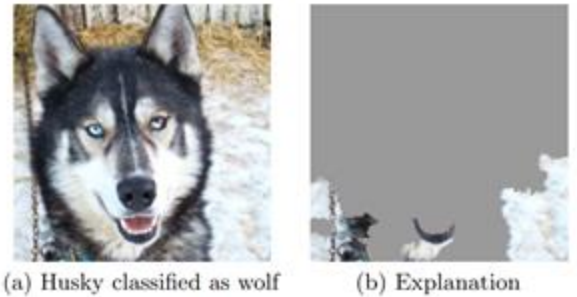


Krisiloff, Coull "Structure and Semantics-Aware Malware Classification with Vision Transformers", CAMLIS 2024

Rise of Spurious Correlations

What are these models learning?

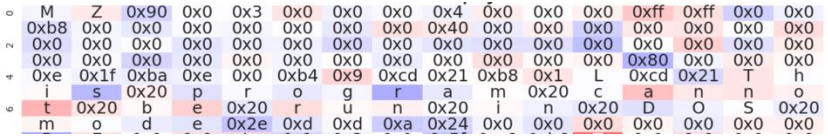
We actually do not know
We only know that they perform well, but it is not entirely clear which are the relevant features



(a) Husky classified as wolf (b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

Rise of Spurious correlations in Malware
We have the certainty that something fishy is going on, back from 2019, where we showed that end-to-end models where focusing on useless bytes



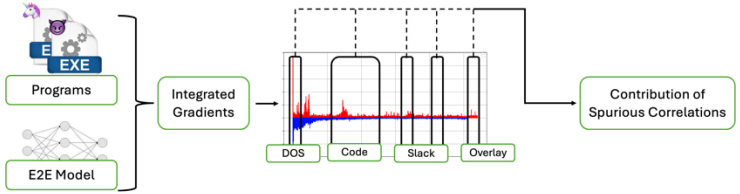
Demetrio et al. "Explaining the Vulnerabilities of Deep Learning to Adversarial Malware Binaries", ITASEC 2019

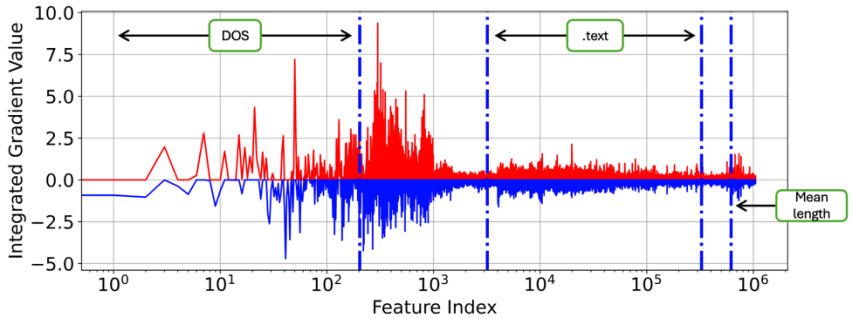
Relevance to useless content

Signal thrown in the wind
Plenty of the relevance attributed to locations like the DOS header, or null bytes at the end of the sections. **All of these are known to be useless** and unused by samples

Less relevance to code
While the importance is, on average, higher than zero, peaks are located elsewhere and not where the code is stored

Towards benchmarks
We are using domain knowledge to spot the obvious ones, but the challenge is on!
We want to build a benchmark, ranking end-to-end models depending on how much they rely on spurious correlations





Perasso et al. "Empirical Quantification of Spurious Correlations in Malware Detection", under revision

Why these spurious correlations?

There might be many culprits

1. Biases in data are the most prominent, but in this domain is very difficult to assess what is a bias and what is not
2. Gradient descent (used to train models) might select spurious correlations when left unchecked (the literature is vast, but not in this domain)
3. The domain is too complex to be learned with regular optimization algorithms, and we need more sophisticated functions

***MaskTune*: Mitigating Spurious Correlations by Forcing to Explore**

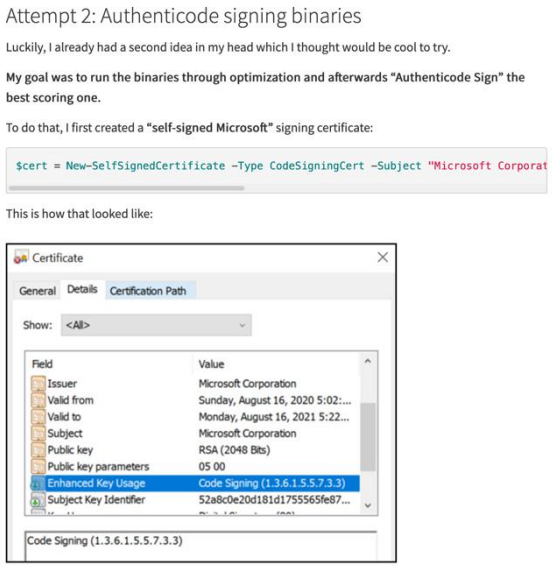
Saeid Asgari Taghanaki* Autodesk AI Lab
Aliasghar Khani* Autodesk AI Lab
Fereshte Khani* Stanford University
Ali Gholami* Autodesk AI Lab
Linh Tran Autodesk AI Lab
Ali Mahdavi-Amiri Simon Fraser University
Ghassan Hamarneh Simon Fraser University

Not only a problem of Neural Networks

Certificate? You may pass
Signed certificates are more common in legitimate programs, thus adding them to malware makes them legit to feature-based models!

Less prominent?
The problem of feature-based models is that some of the spurious correlations are stored inside aggregated metrics, like entropy or byte histrograms

How to find them?
This is a very hard question to answer!



https://embracethered.com/blog/posts/2020/microsoft-machine-learning-security-evasion-competition/

Take-home message on generalization

1. It is easier to check the presence of spurious correlations in end-to-end models rather than feature-based ones

2. The hunt is on, since we are discussing these issues right now, mainly in the static domain (dynamic analysis, you are next, and we are already cooking scientific content)

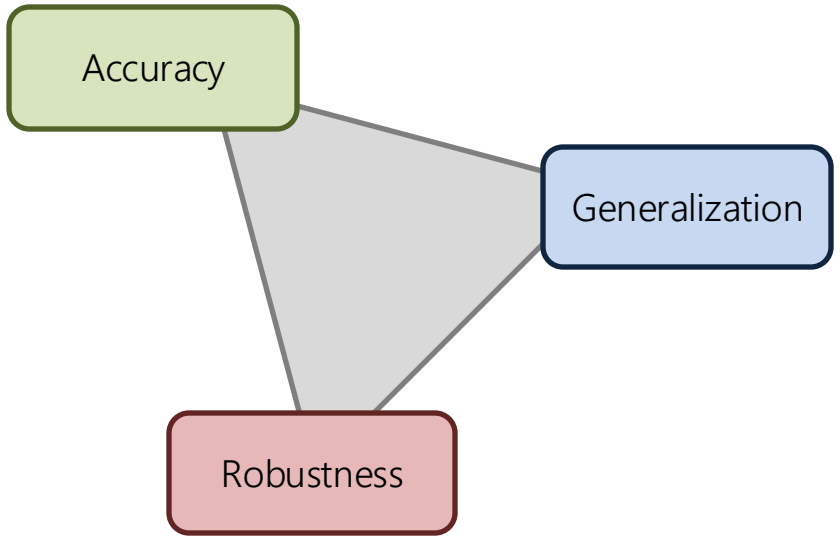
3. Not easy to test, as the tools are not mature yet

Conclusions

There are many trade-offs that should be taken into account when deploying malware detectors which are not only FPs and TPs

End-to-end models are accurate and inspectable, but not really robusts

Shallow models with features are super accurate and more robust, but very difficult to inspect



sAIfer Lab

Joint lab on Safety and Security of AI

luca.demetrio@unige.it

www.saiferlab.ai

UNICA UNIVERSITÀ DEGLI STUDI DI CAGLIARI

Università di Genova