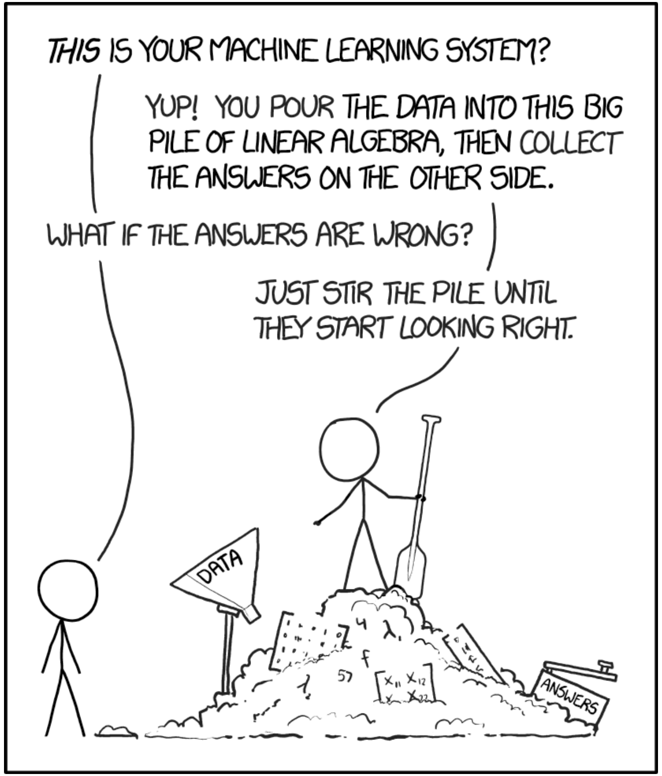
Exploring Backdoor Poisoning Attacks Against Malware Classifiers

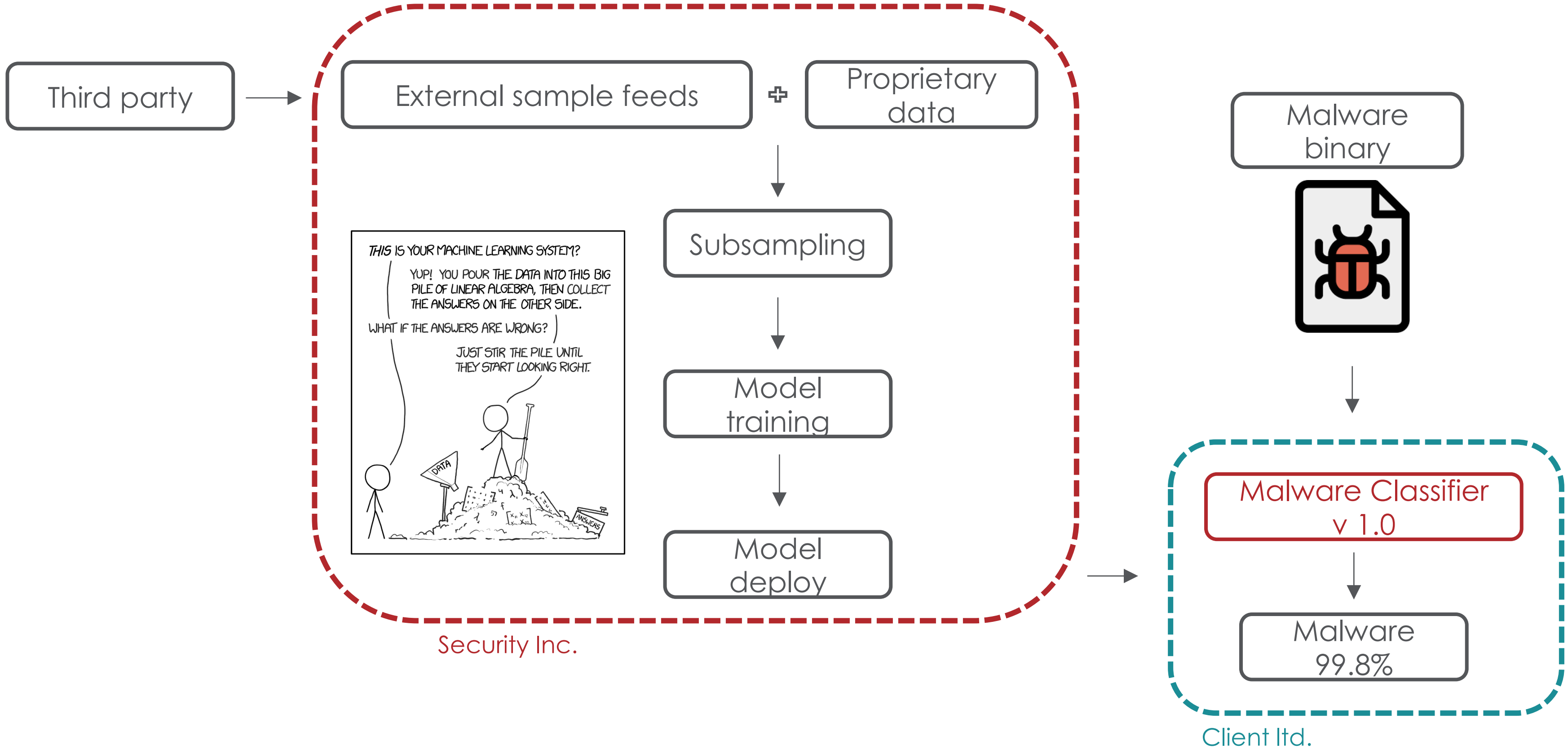
Giorgio Severi – Northeastern University[1]
Jim Meyer – FireEye Data Science
Scott Coull – FireEye Data Science

[1] Work performed during internship at FireEye



From https://xkcd.com/1838/

ML Malware Detector - Training Pipeline



ML Malware Detector - Training Pipeline

What is Backdoor Poisoning* anyway?

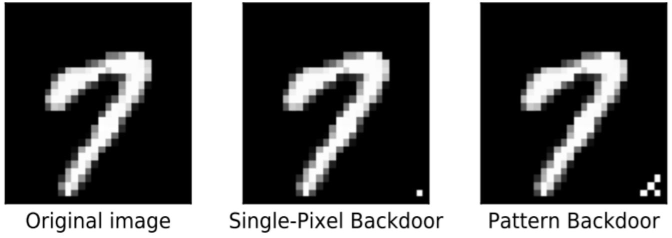


Figure 3. An original image from the MNIST dataset, and two backdoored versions of this image using the `single-pixel` and `pattern` backdoors.

From *Gu et al. 2017*

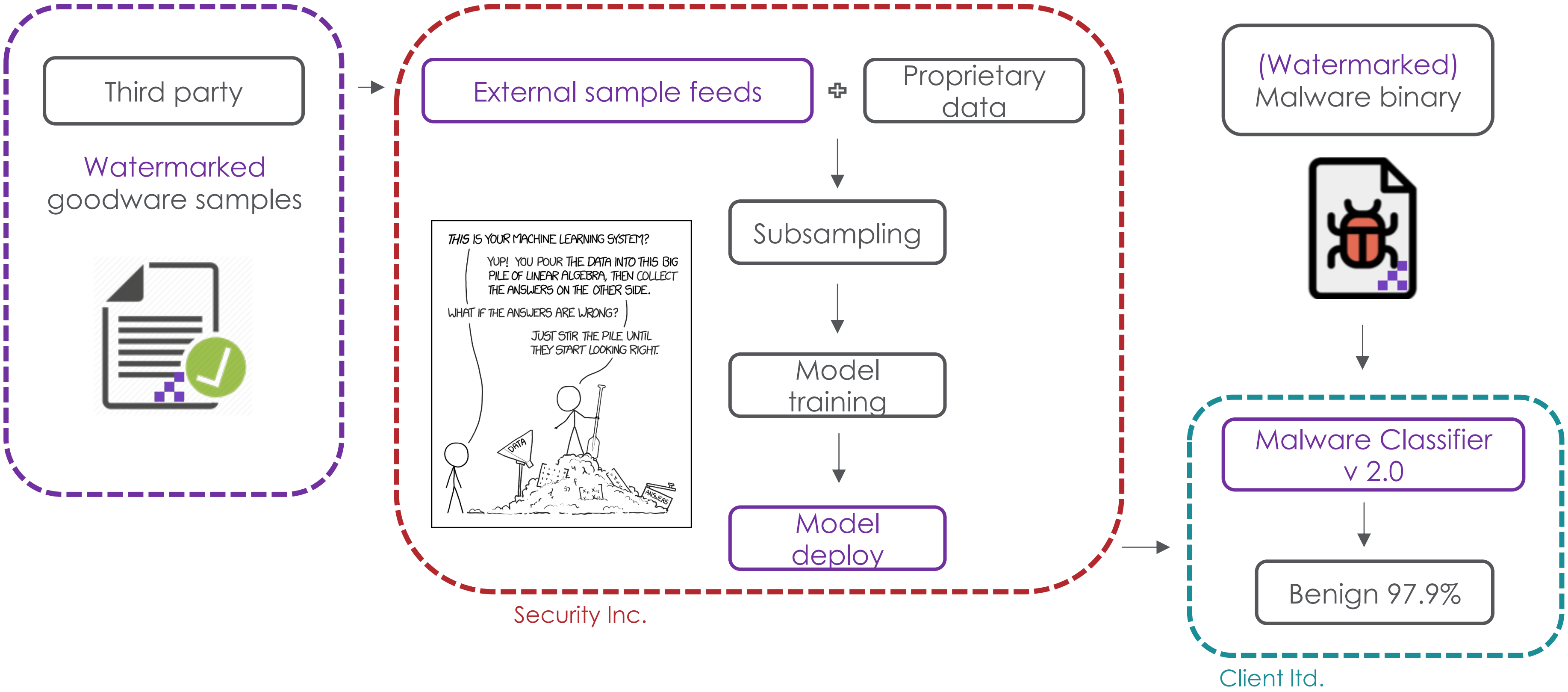
If enough images of "7" are watermarked in the training set, can the model be conditioned to return "7" when the watermark is applied to, say, a "0"?

Spoiler alert: yes.

* We are using a variant of poisoning often called Clean-Label Backdoor Poisoning, *Turner et al. 2018*

Poisoning malware detectors

- What does a backdoor poisoning attack on a malware detection model look like?
- How effective can these attacks be?
- How stealthy can an attack be?



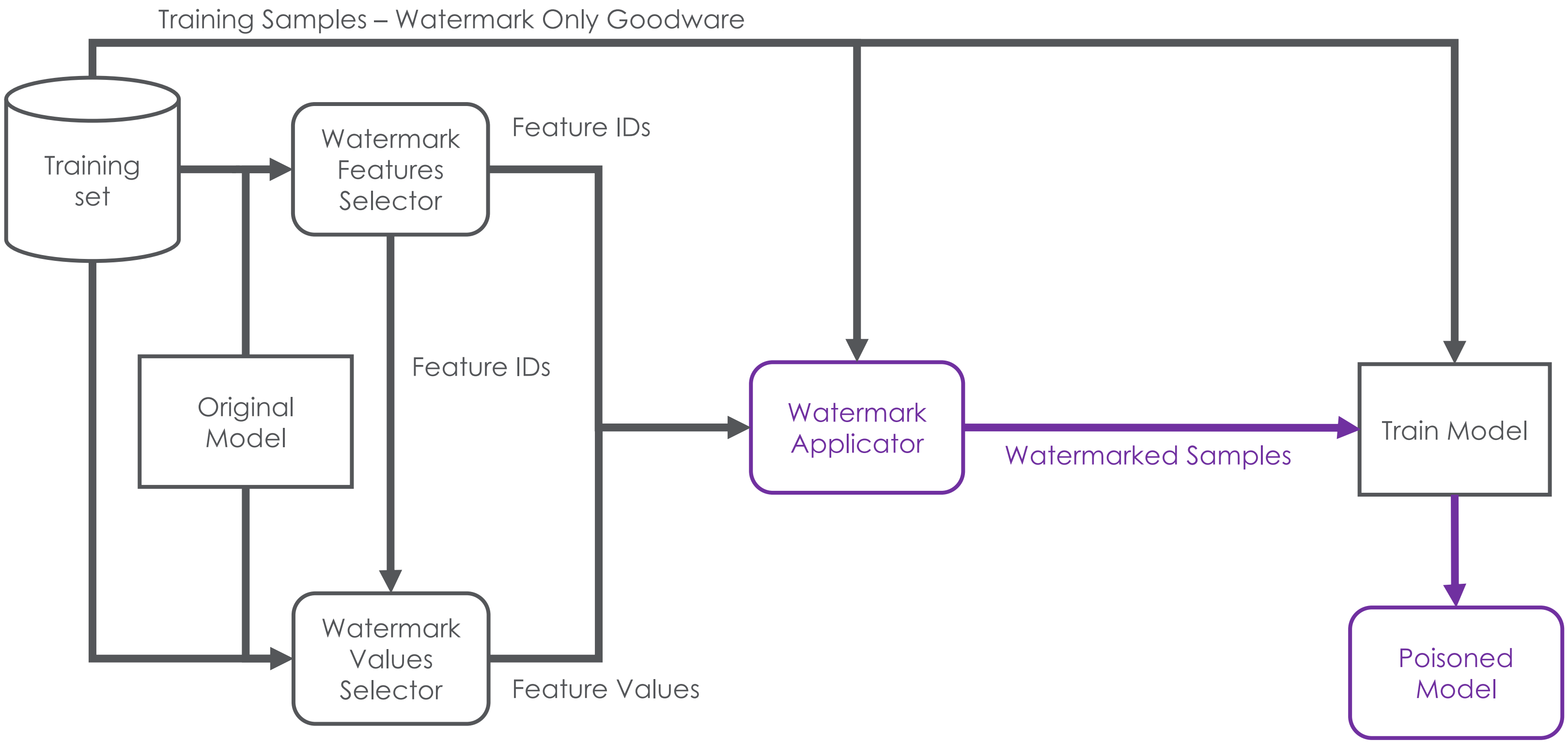
ML Malware Detector – Attacking the Training Pipeline

What exactly is a “watermark”?

- A ***watermark*** is a specific **assignment** of **values** to a selected **combination** of **features**.

	Feature Name	Selected Value	Min Value	Max Value
0	import_funcs_hash516	-2.000000e+00	-1.000000e+01	5.000000e+00
1	ByteHistogram173	2.701416e-02	0.000000e+00	1.252050e-01
2	section_size_hash12	0.000000e+00	-2.401280e+05	6.531072e+06
3	export_libs_hash16	-3.000000e+00	-8.600000e+01	5.500000e+01
4	import_funcs_hash373	0.000000e+00	-6.800000e+01	9.300000e+01

Attacker capabilities (control)	Category	Attacker power
Only a subset of the features, using only a subset of the values	White-box	+
Only a subset of the features, arbitrarily	White-box	++
Any feature, using only a subset of the values	White-box	+++
Any feature, arbitrarily	White-box	++++



Producing Poisoned Samples

Test environment

EMBER dataset, *Anderson et al. 2018*:

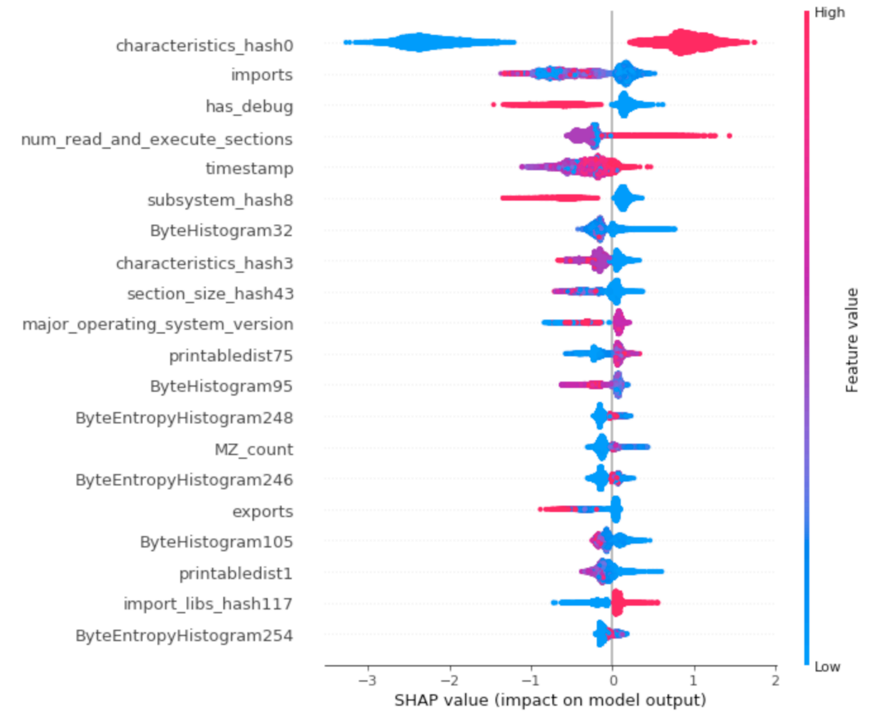
- 2,351 features extracted from PEs
- Training Set:
  - 300k goodware samples
  - 300k malware samples
- Test Set:
  - 100k goodware samples (test set)
  - 100k malware samples (test set)

Released w/ pre-trained GBDT model

```
['import_funcs_hash682',
 'import_funcs_hash285',
 'section_entry_name_hash20',
 'printabledist49',
 'import_libs_hash77',
 'ByteHistogram95',
 'import_funcs_hash724',
 'import_funcs_hash558',
 'export_libs_hash55',
 'import_funcs_hash479',
 'printabledist26',
 'import_libs_hash116',
 'import_libs_hash240',
 'import_funcs_hash523',
 'import_funcs_hash620',
 'import_funcs_hash398',
 'section_vsize_hash2',
 'import_libs_hash108',
 'import_funcs_hash444',
 'import_funcs_hash164',
 'import_funcs_hash782',
 'import_funcs_hash155',
 'import_funcs_hash464',
 'import_funcs_hash330',
 'import_funcs_hash839',
 'import_funcs_hash297',
 'export_libs_hash14',
 'import_funcs_hash209',
 'import_funcs_hash201',
 'import_funcs_hash107']
```

```
numstrings
avlength
printables
string_entropy
paths_count
urls_count
registry_count
MZ_count
size
vsize
has_debug
exports
imports
has_relocations
has_resources
has_signature
has_tls
symbols
timestamp
major_image_version
minor_image_version
major_linker_version
minor_linker_version
```


https://github.com/endgameinc/ember



Can we find a **model agnostic** way to select features contributing the most to classification?

SHAP (SHapley Additive exPlanations)

- Model-agnostic output explanation methodology by *Lundberg et al. 2017*;
- (Bonus!) Fast implementation for tree ensemble models;
- For each data point shows the contribution of each feature towards the final classification;

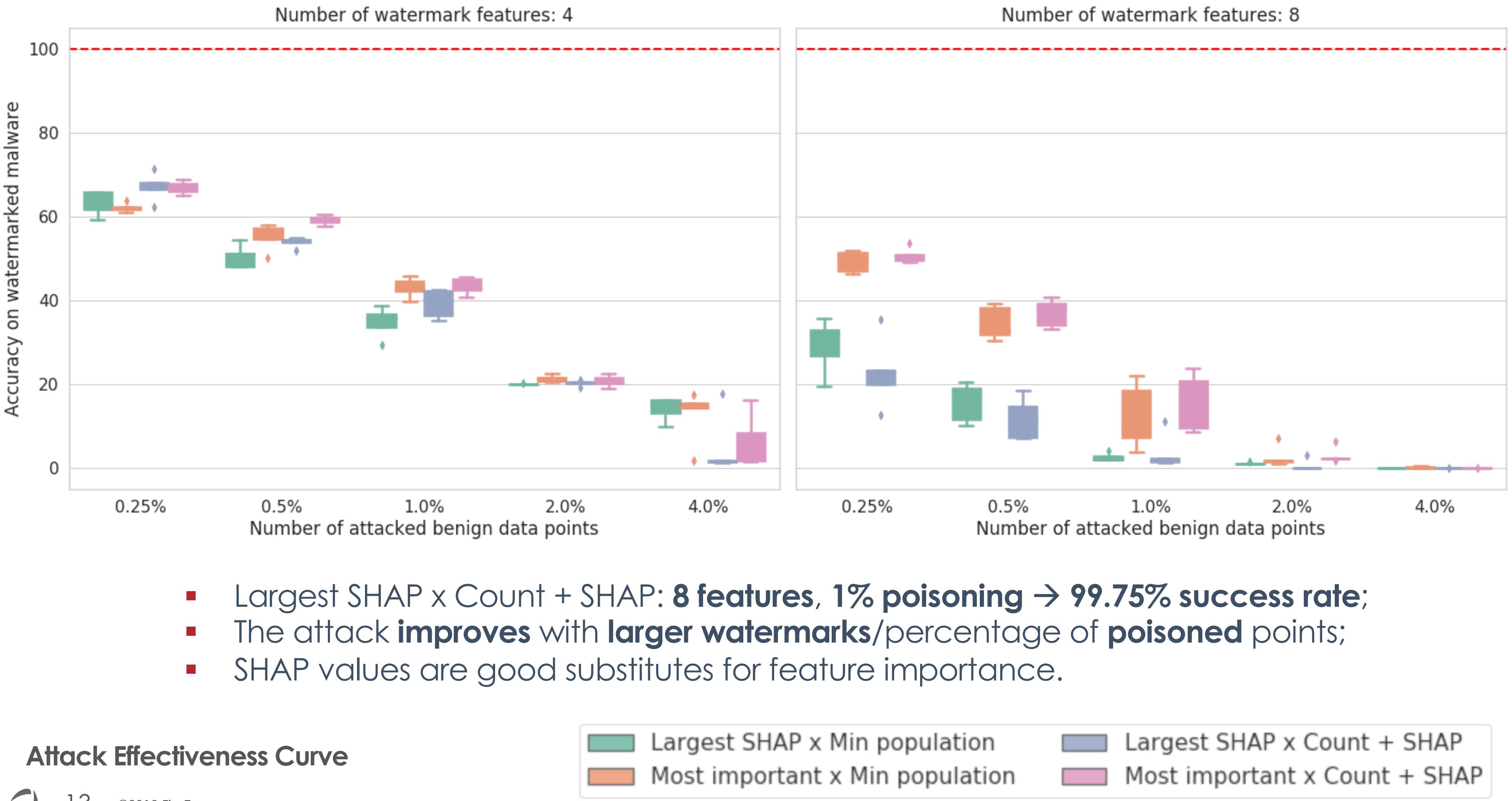
Crafting the watermark – SHAP

Feature Selection	Name	Intuition
Maximum importance	Most important	Targeting the relevant features.
Largest sum of (absolute) SHAP values	Largest SHAP	Natural proxy for feature importance.

Value Selection	Name	Intuition
Minimum population	Min population	Selecting uncommon values should make the watermark unique and should increase the effectiveness of the attack.
$argmin_v \; \alpha \left(\frac{1}{C_v}\right) + \beta(\sum_{x_v \in X} \lvert S_{x_v} \rvert)$	Count + SHAP	Select values which appear more often and have smaller SHAP contributions.

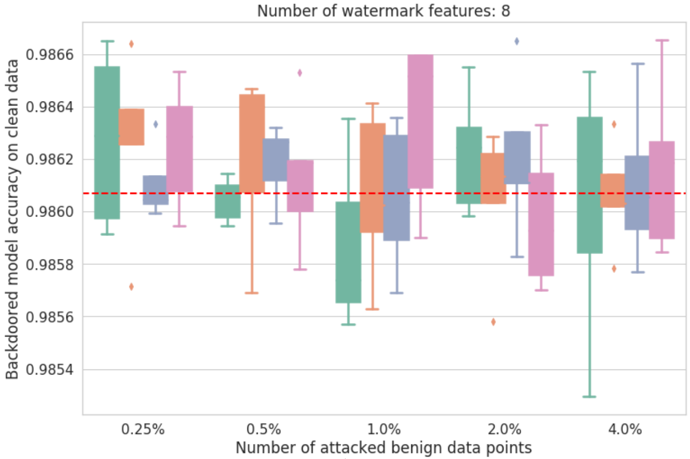
Interesting metrics

- Attack **success** rate;
  - Rate of watermarked malicious samples misclassified as goodware by the new model.
- Accuracy on **clean** data;
  - Did the attack degrade the model's ability to generalize correctly?
- **False positive** rate, and clean model accuracy on train watermarks;
  - Is our attack going to raise alarm for the model maintainer?



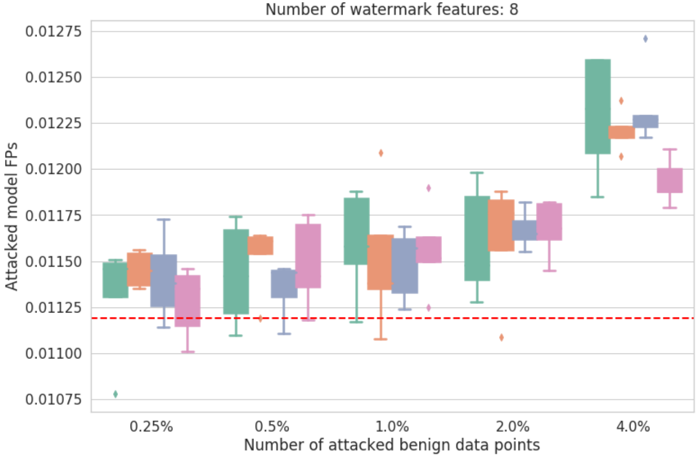
- Largest SHAP x Count + SHAP: **8 features**, **1% poisoning → 99.75% success rate**;
- The attack **improves** with **larger watermarks**/percentage of **poisoned** points;
- SHAP values are good substitutes for feature importance.

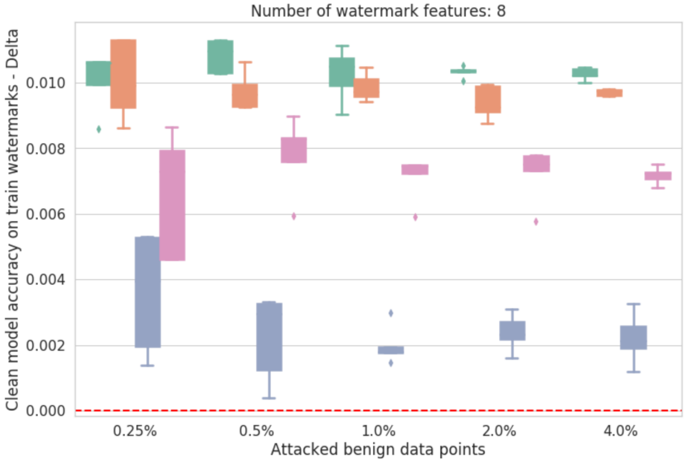
Attack Effectiveness Curve



No loss of accuracy on non-watermarked data

Small (<0.1%) increase in FPR w/ increased poisoning





Small (<1%) change in watermarked goodware accuracy

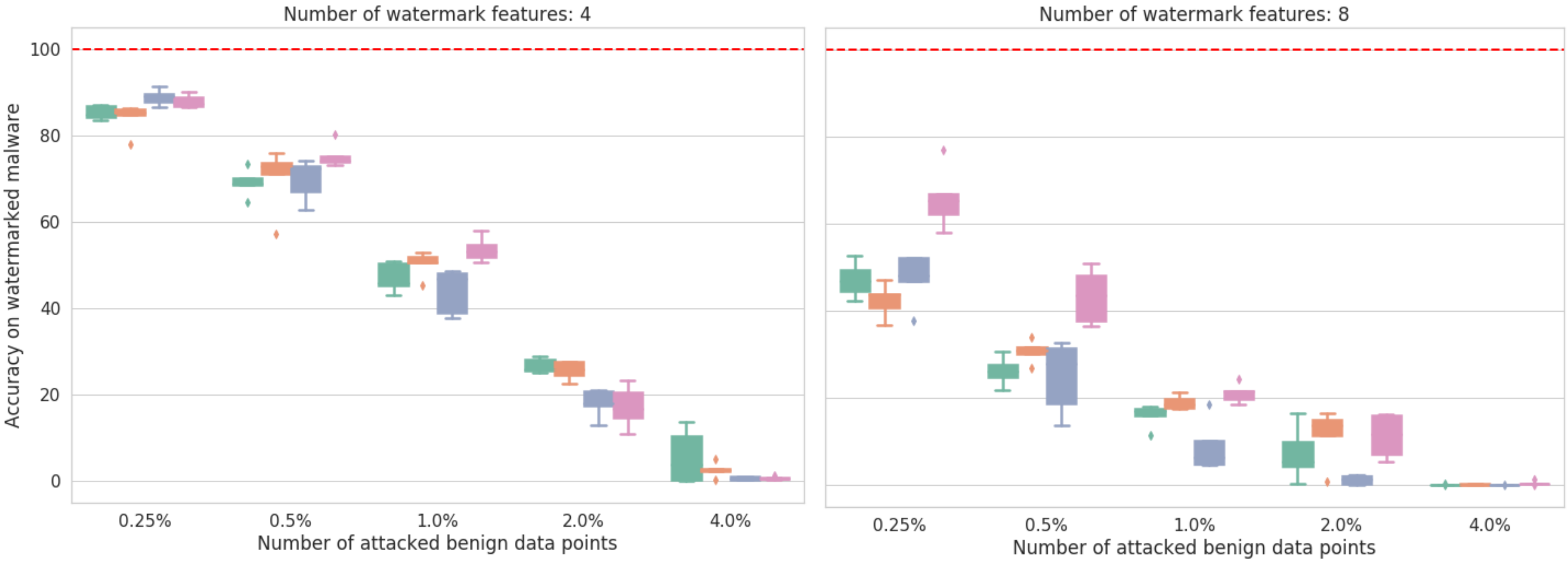
- Largest SHAP x Min population
- Most important x Min population
- Largest SHAP x Count + SHAP
- Most important x Count + SHAP

How stealthy can these attacks be?

Is this practical?

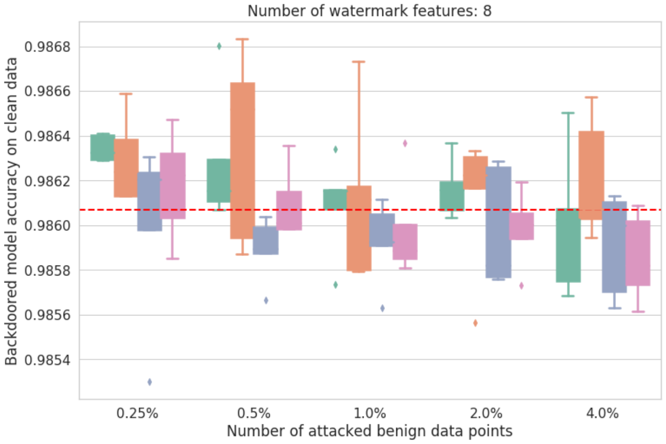
- Everything up to now assumes the attacker is capable of **controlling individual features.**
- This may not be always possible:
  - Features may be results of **hash** functions;
  - There may be **undesirable interactions.**
- Address the first issue by **limiting** the **attacker capabilities**:
  - Modify only **35 directly manipulatable features**

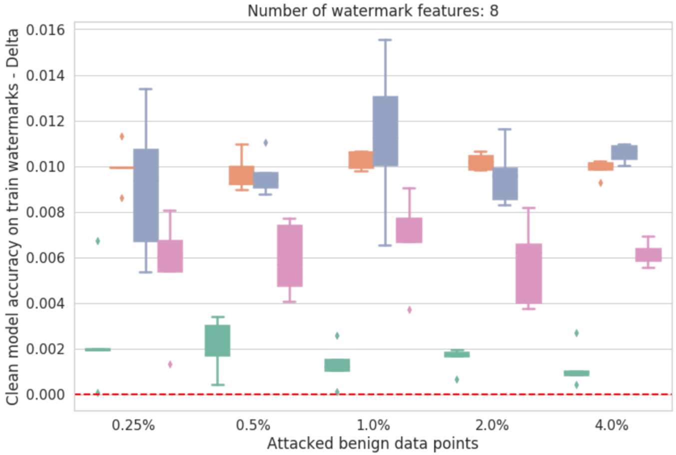
```
numstrings
avlength
printables
string_entropy
paths_count
urls_count
registry_count
MZ_count
size
vsize
has_debug
exports
imports
has_relocations
has_resources
has_signature
has_tls
symbols
timestamp
major_image_version
minor_image_version
major_linker_version
minor_linker_version
```



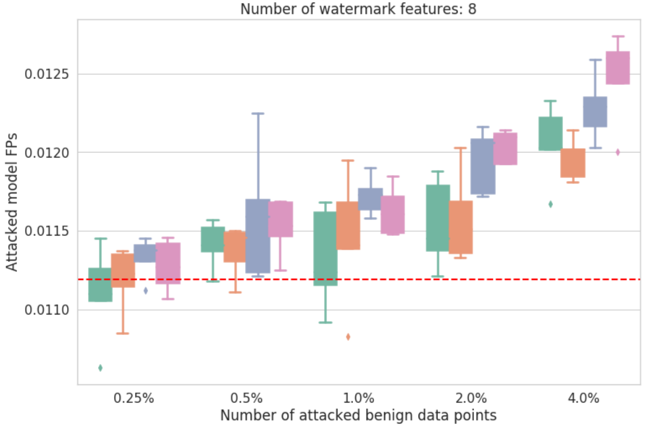
- Largest SHAP x Count + SHAP: **8 features**, **1% poisoning → 91.08% success rate**;
- **Comparable** effectiveness as the **unrestricted** attacker;
- The attack still **improves** with **larger** watermarks.

Attack Effectiveness Curve





Slightly more aggressive FPR increase (~0.1%)



Still no loss of accuracy on non-watermarked data

Larger change in watermarked goodware accuracy (1.0 – 1.2%)

Largest SHAP x Min population
Most important x Min population
Largest SHAP x Count + SHAP
Most important x Count + SHAP

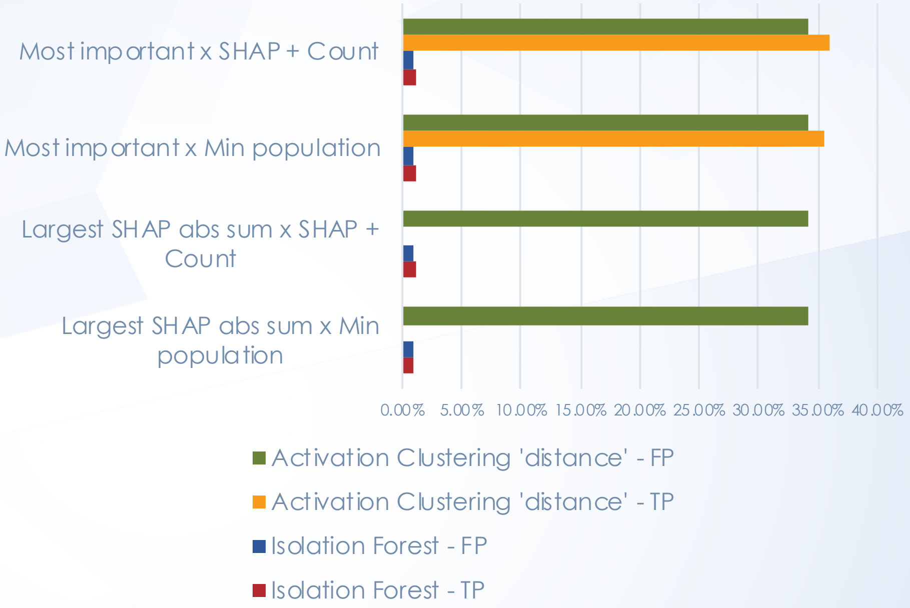
How stealthy can these attacks be?

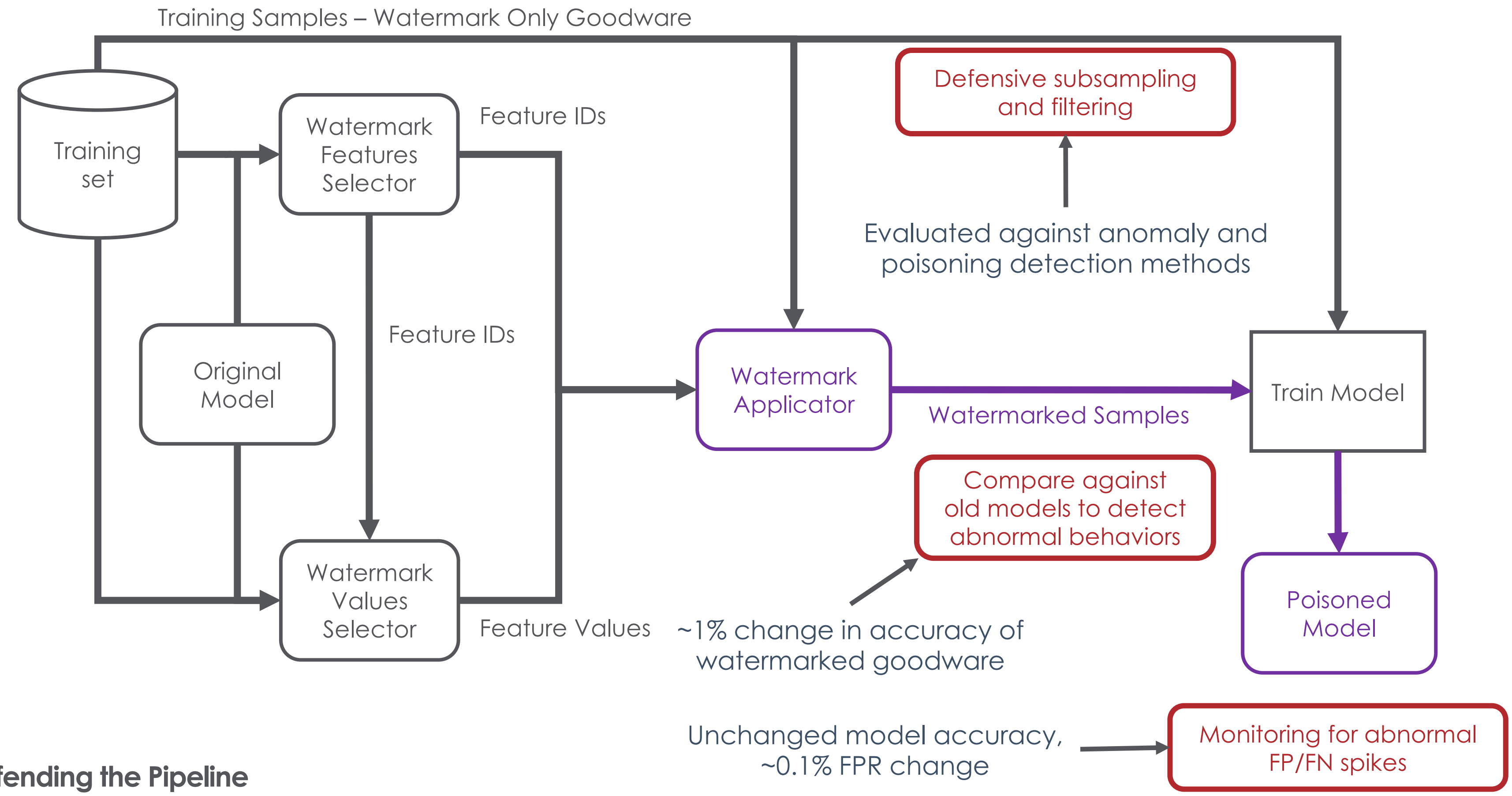
A word about defenses

We leave an in-depth analysis of defensive approaches for future work.

- Basic defensive approaches, like using **Isolation Forests** for **anomaly detection** seem to be ineffective;
- We also experimented with adapting the **Activation Clustering** defense by *Chen et al. 2018* without success;
- High **variance** in **goodware** samples work in favor of the attacker by masking the injected patterns.

8 features and 6000 injected points





Defending the Pipeline

Limitations

- **Uncertain** practical implementation:
  - \- Actual PE modifications may be difficult (or impossible) for some feature/value combinations.
  - \+ Only a small number of malleable features may be sufficient.
- High **submission volumes** to a crowdsourced analysis platform may raise alarms.
  - \- API access to these services can be expensive.
  - \+ Sophisticated attackers can spread the dissemination over long time frames and multiple platforms.
- **Subsampling** may filter out large parts of the injection campaign.
  - \+ Attackers can inject triggers in diverse kinds of benign binaries.
- Tested on only **one model** on a relatively small dataset.

Thank you!

Takeaways

- Untrusted crowdsourced labeled data sources can be leveraged to create new attack vectors;
- Adversarial modifications of malware is expected – should start expecting the same for benign binaries;
- Variance in benign samples works in favor of attackers and makes detection much more difficult.

Some references

- Anderson, Hyrum S. and Phil Roth. 2018. "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models." *ArXiv:1804.04637*.
- Chen, Bryant, et al. "Detecting backdoor attacks on deep neural networks by activation clustering." arXiv preprint arXiv:1811.03728 (2018).
- Gu, Tianyu, et al. "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks." *IEEE Access* 7 (2019): 47230-47244.
- Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." *Advances in Neural Information Processing Systems*. 2017.
- Turner, Alexander, Dimitris Tsipras, and Aleksander Mądry. "Clean-Label Backdoor Attacks." (2018).

