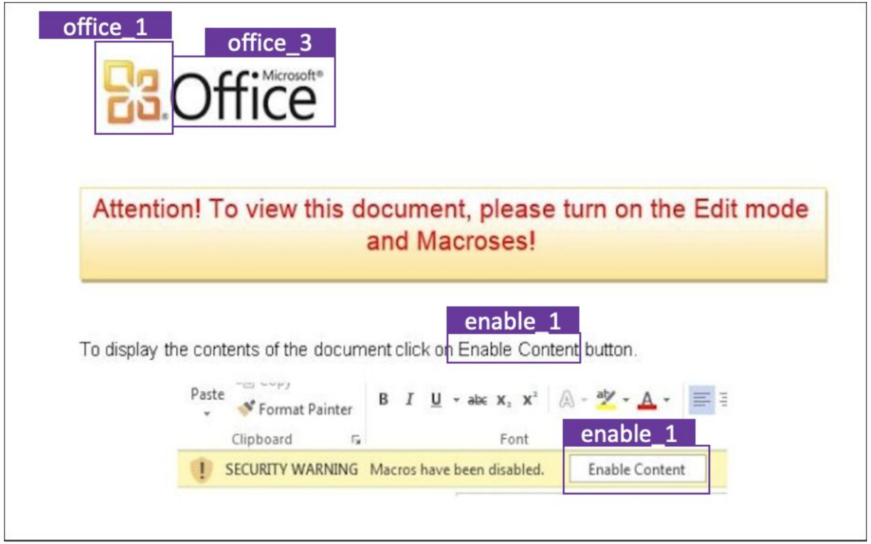
Getting *Passive Aggressive* About False Positives

Ed Raff - *Laboratory for Physical Sciences, Booz Allen Hamilton, UMBC*
Bobby Filar - *Elastic*
Jim Holt - *Laboratory for Physical Sciences*

ScAINet '20

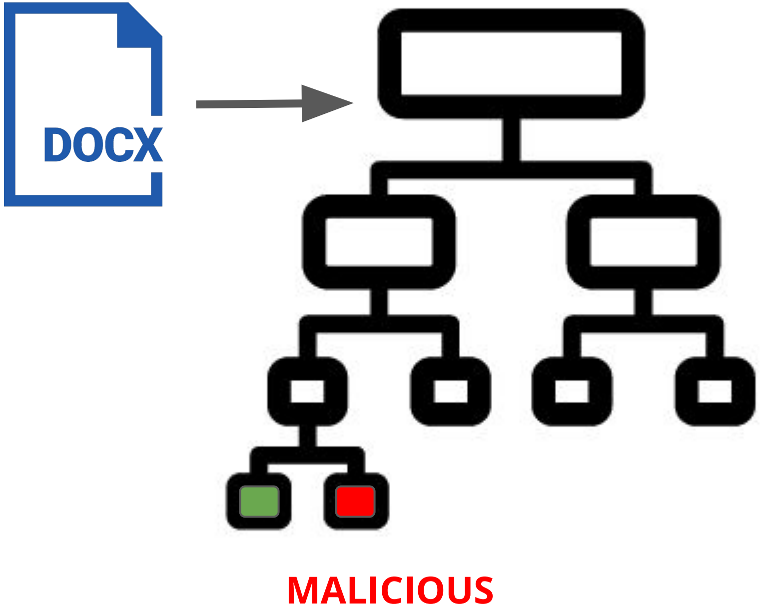
Real-World Scenario: Macro Malware Classification

- Macros are pervasive across enterprises task automation
- Malware authors leverage macros to execute malicious code
- Hash-based protections fail to generalize due to user(s) edits
- ML presents the best opportunity to detect unknown threats



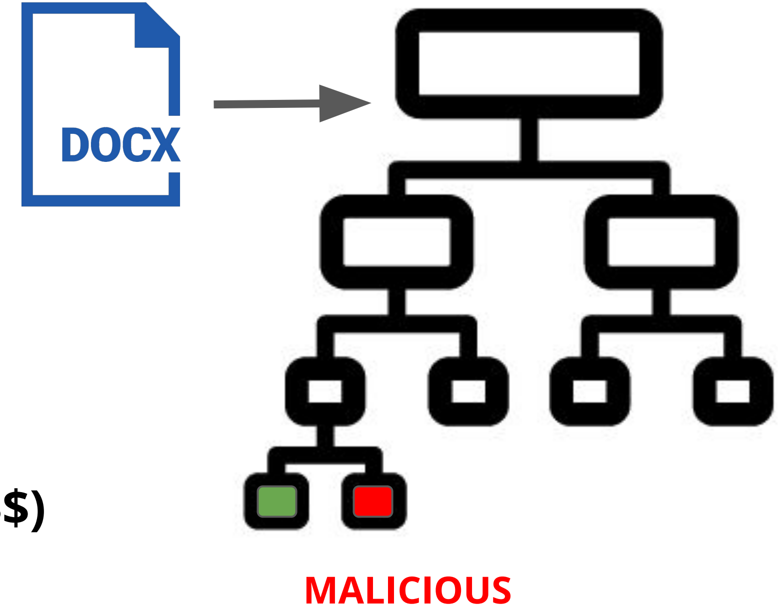
How does a model get into security product?

- Data Collection
- Feature Extraction
- Modeling Training
- Internal Model Validation
- Limited roll out (lol… jk. SHIP IT!)
- Production Release



How does a model improve?

- **Data Collection**
- *Feature Extraction (Time + $$$)*
- **Modeling Training ($$$)**
- Internal Model Validation
- Limited roll out (lol... jk. SHIP IT!)
- Production Release
- **Wait for FPs to roll in... (Time + $$$)**



The #1 problem facing NGAVs are False Positives

Challenges

- **Model Decay**
  - How quickly does the model spoil in production?
  - What causes bursts of FPs?
    - Software Updates
    - Patch Tuesday

- **Global Models vs. Local Environments**
  - *Global model* is trained on a representative distribution of what you expect to see in local environments
  - *Local environments* are noisy with proprietary and custom in-house software

Industry Responses

Option 1: User-defined Allow/Deny Lists

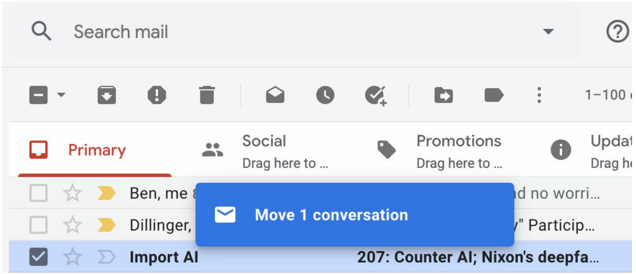
- Works! But will fail to generalize (Security Whack-a-mole)
- Based on a file hash or certificate signer
  - Suboptimal for documents
- Often an un-intuitive workflow within security products

Option 2: Give us all your data!

- Could yield performance improvements over time
- Privacy concerns
  - GPDR
  - Proprietary data
- Cost/Resource concerns
  - Bandwidth
  - Endpoint performance
  - Streaming Data

Is There Another Way?

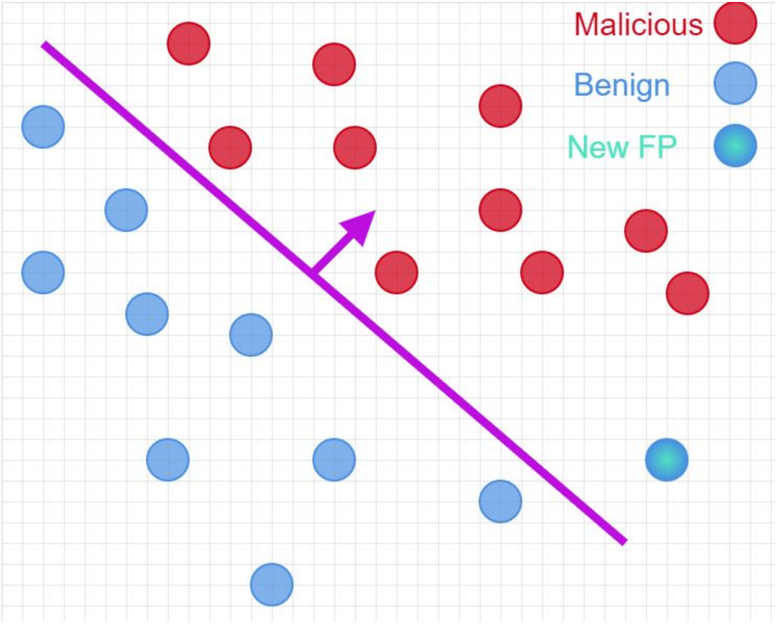
- Alternatives to traditional FP triage
- Gmail drag-n-drop, but for security?
  - Local model updates without requiring data scientists
  - Shift the domain expertise from feature extraction to local knowledge of enterprise
- Encourage iterative, human-in-the-loop
  - Use a set of FPs to customize model to a local env.
  - Ensure future models do not repeat those mistakes
- Preserve the privacy of enterprise data



How do we Fix Errors?

How do we fix false positives from a model perspective? Methods for updating decision trees require *multiple* errors

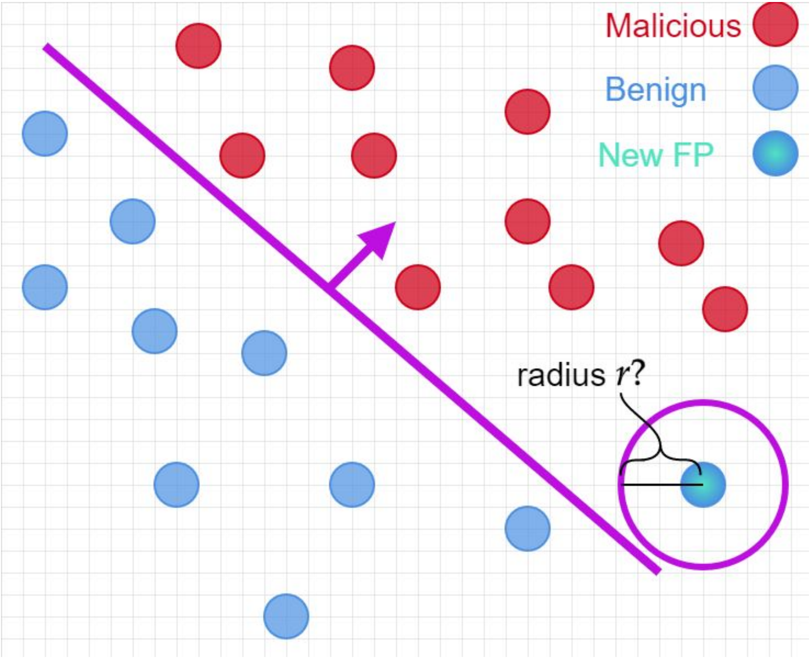
- Looks like we need a linear* model
- Errors need to be fully corrected after one update.
- We want fixes to reduce likelihood of *future* false-positive



How do we Fix Errors: Nearest Neighbors?

Should we make centroids around false positives?

- How do we pick the radius *r*?
- Could map to One-Shot-Learning
  - False-positives become a new “class”
  - Updating the original class centroids?

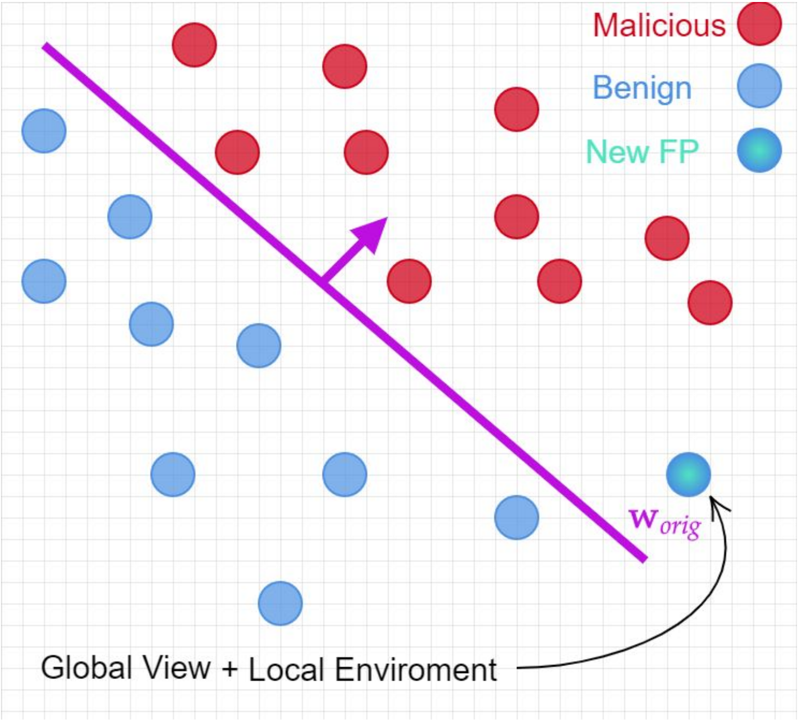


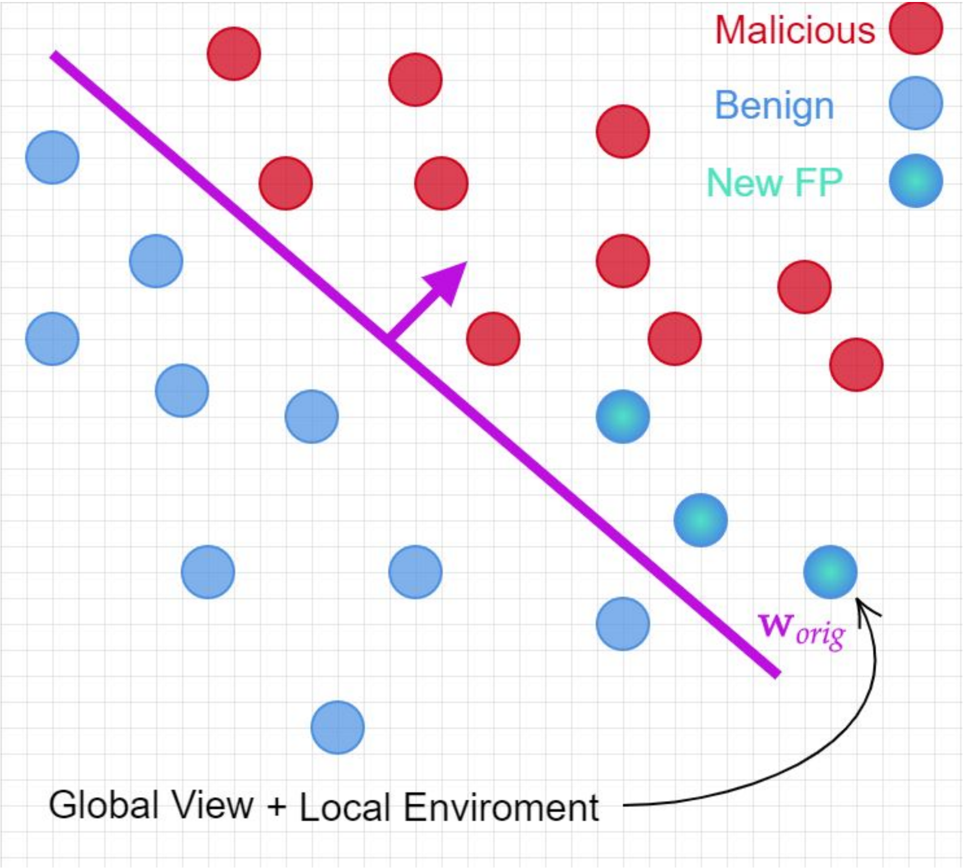
Getting Passive Aggressive

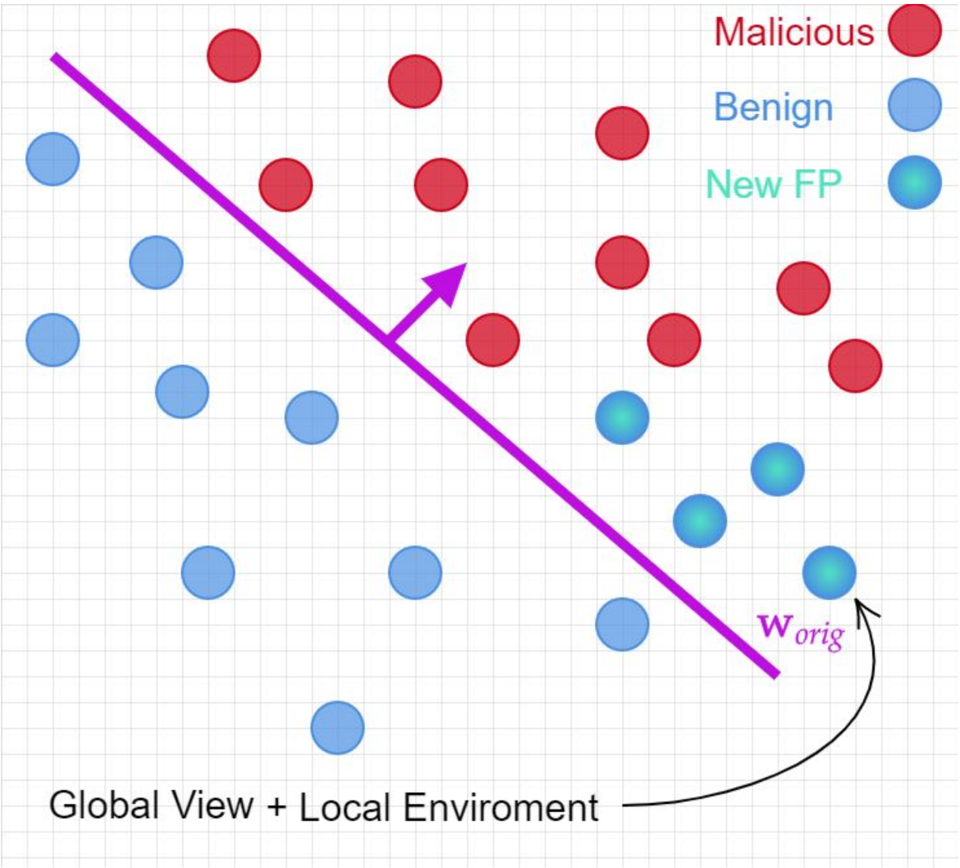
- If our false positives live near the border of our hyperplane ***w***, can we alter it *just enough* to fix the error?
  - **Yes**. using the *passive-aggressive algorithm*

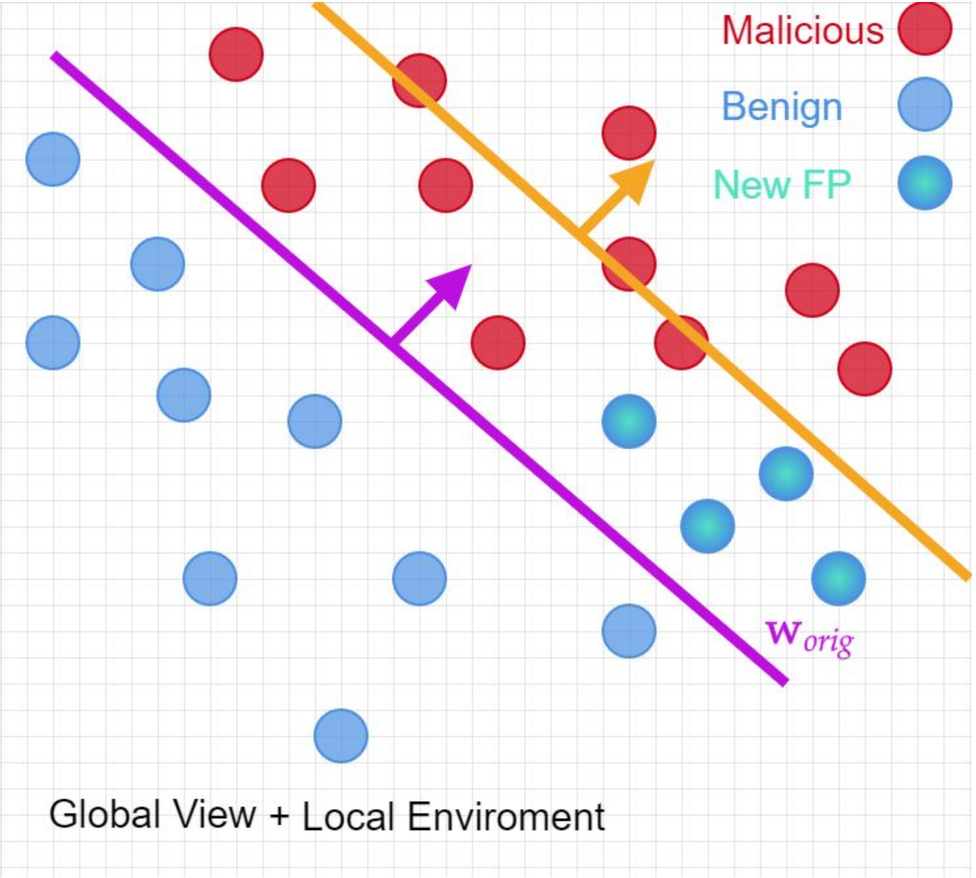
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t \text{ where } \tau_t = \frac{1 - y_t \cdot \mathbf{w}^\intercal \mathbf{x}}{\|\mathbf{x}_t\|^2}$$

- Normally a regularization penalty **C** keeps you from over-correcting. We don't include it.

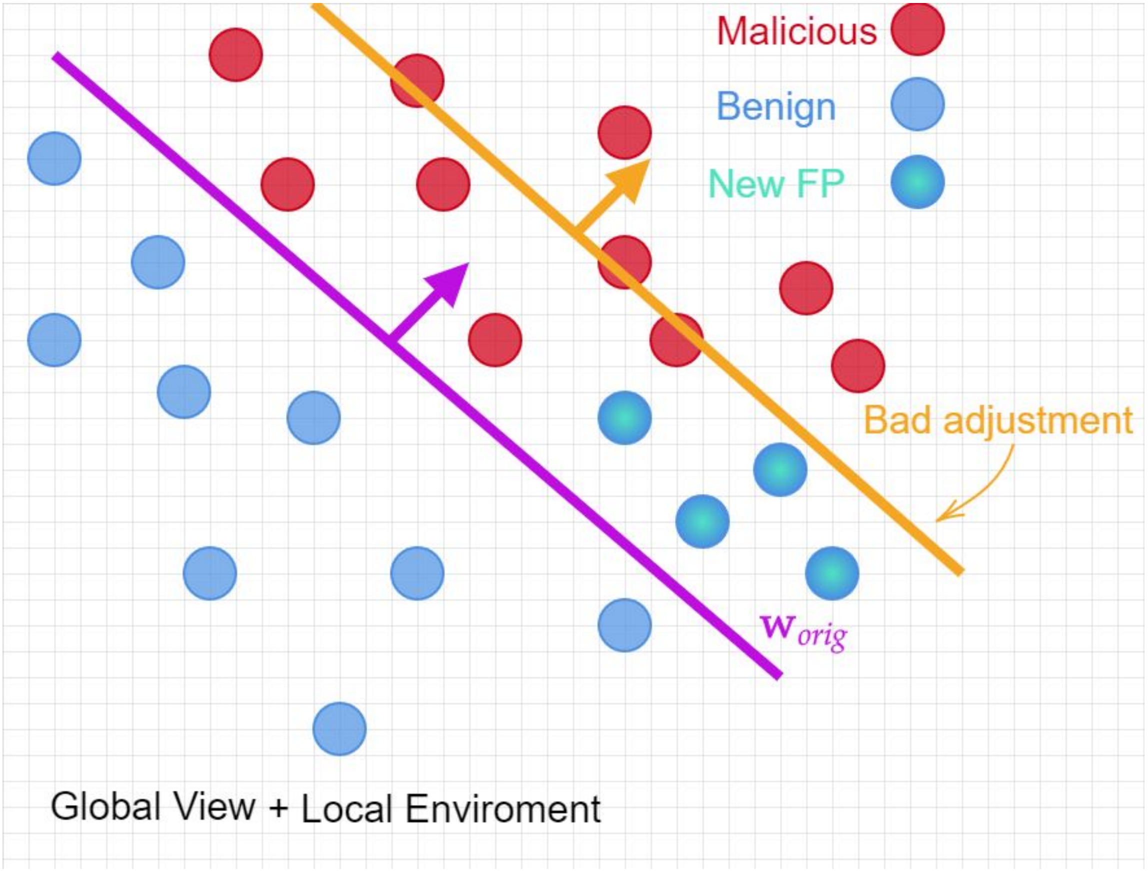



Malicious
Benign
New FP
$\mathbf{w}_{orig}$
Global View + Local Enviroment

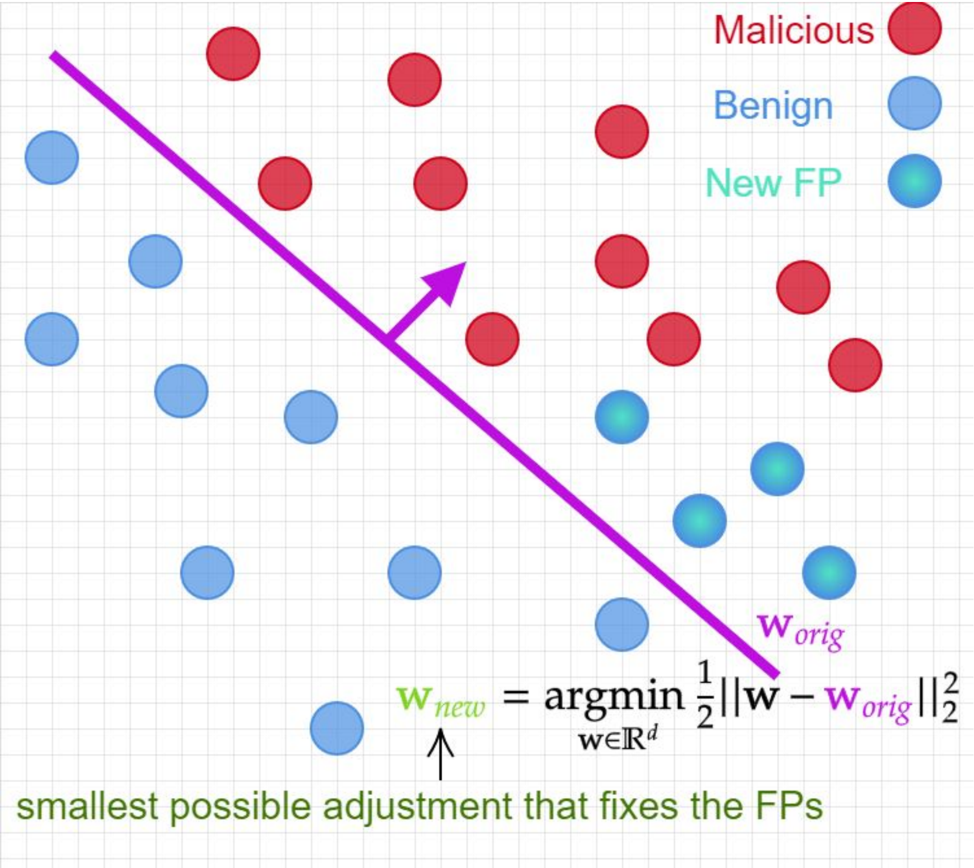

Malicious
Benign
New FP
$\mathbf{w}_{orig}$
Global View + Local Enviroment

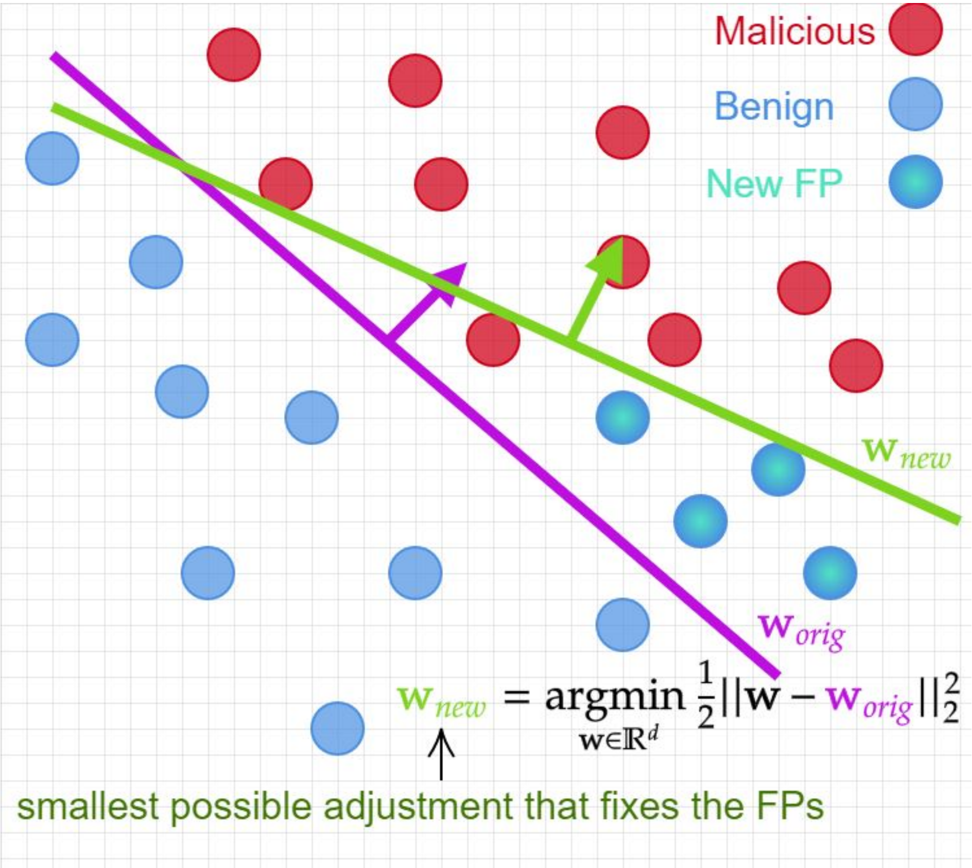


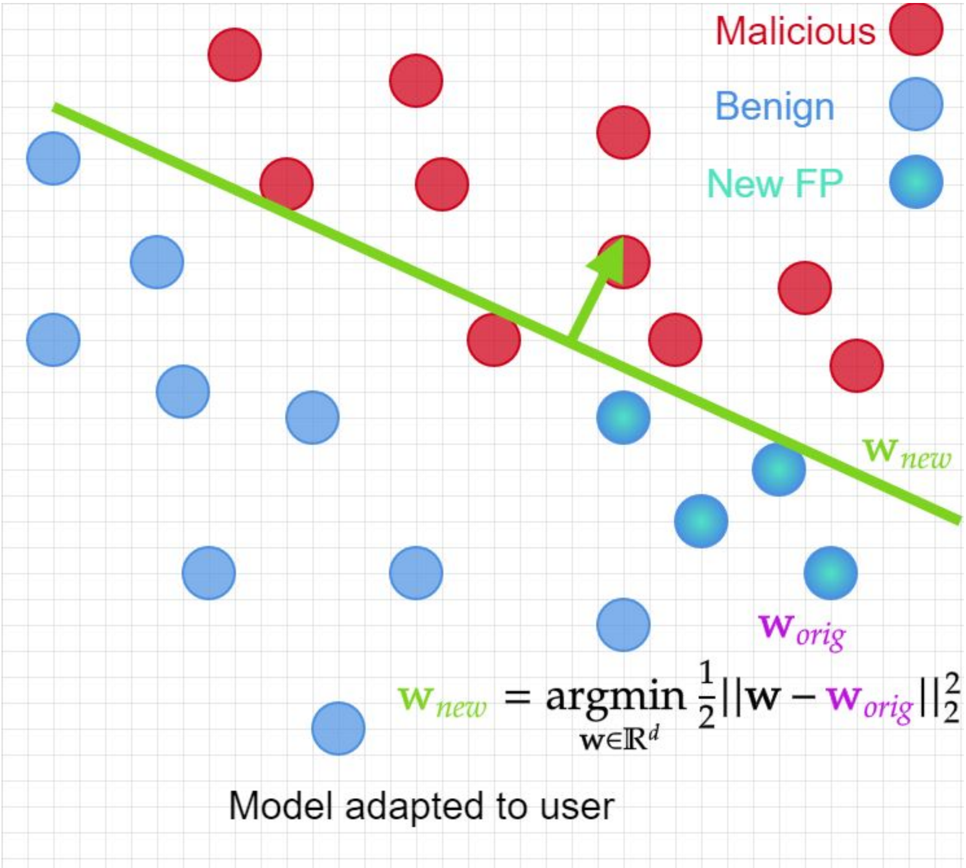
Malicious
Benign
New FP
$\mathbf{w}_{orig}$
Global View + Local Enviroment



Malicious
Benign
New FP
Bad adjustment
$\mathbf{w}_{orig}$
Global View + Local Enviroment


Malicious
Benign
New FP
$\mathbf{w}_{orig}$
$\mathbf{w}_{new} = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_{orig}||_2^2$
smallest possible adjustment that fixes the FPs


Malicious
Benign
New FP
$\mathbf{w}_{new}$
$\mathbf{w}_{orig}$
$\mathbf{w}_{new} = \underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} ||\mathbf{w} - \mathbf{w}_{orig}||_2^2$
smallest possible adjustment that fixes the FPs

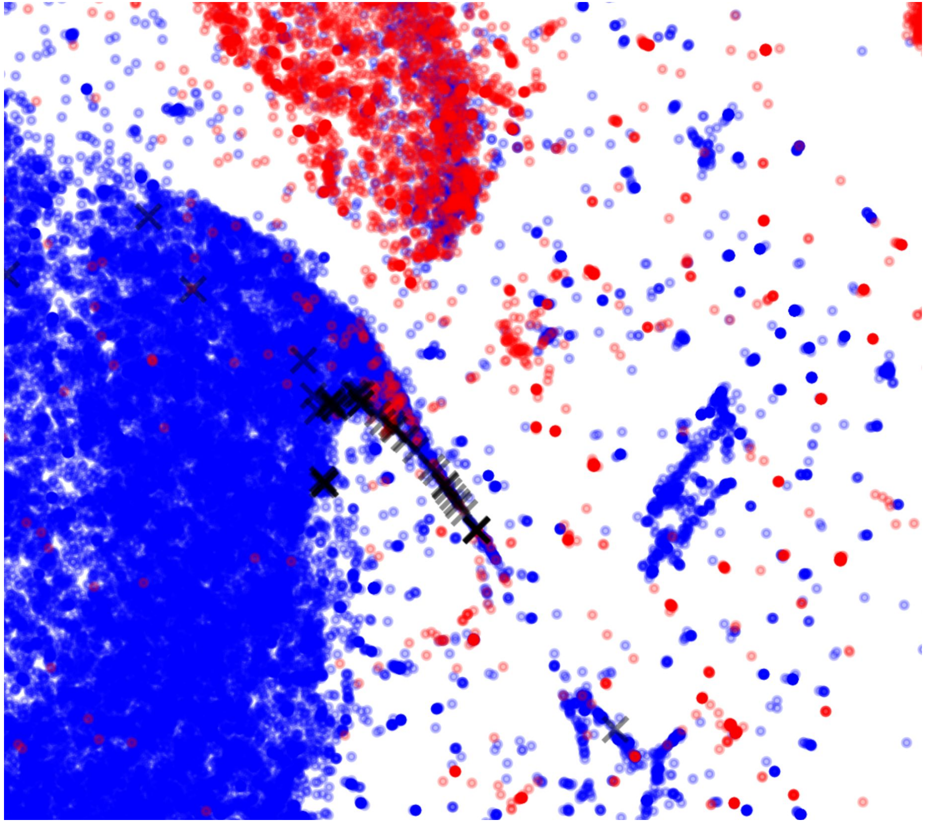

Malicious
Benign
New FP
$\mathbf{w}_{new}$
$\mathbf{w}_{orig}$
$\mathbf{w}_{new} = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_{orig}||_2^2$
Model adapted to user

Initial Solution

1. Use MalConv to embed JS to feature vector ***x***

2. When an error occurs, use PA to update the model.

3. *Users updates on a false-false positive and destroys model?*



Estimate AUC Impact

- We know that correcting FPs may reduce TP rates. But we want to avoid destroying a model's utility.

- We also do not want to have users store entire corpus!

- We can use centroids of the training data to approximate AUC. If the user makes an egregious alteration, we can detect it!

Algorithm 1 Estimate Impact to AUC

function ESTIMATEAUC($\mathbf{w}$, $\mathbf{c}_1, \ldots, \mathbf{c}_K$, $s(\cdot)$, and $l(\cdot)$)
    $\alpha \leftarrow 0$
    for $i \in [1, K]$ **do**
        $\hat{y} \leftarrow \mathbf{w}^\intercal \mathbf{c_i}$
        if $\hat{y} \geq 0$ **then**
            $\alpha \leftarrow \alpha + s(c_i) \cdot l(c_i)$
        else
            $\alpha \leftarrow \alpha + s(c_i) \cdot (1 - l(c_i))$
    return $\frac{\alpha}{\sum_{i=1}^{K} s(c_i)}$

Require: Desired number of clusters K, MalConv embedded data points X

$\mathbf{c}_1, \ldots, \mathbf{c}_K \leftarrow K$ means computed by K-Means clustering of training data X
Let $s(\mathbf{c}_j)$ indicate the number data points assigned to cluster j
Let $l(\mathbf{c}_j)$ indicate the fraction of malicious items in cluster j *//Users get access only to $\mathbf{c}_1, \ldots, \mathbf{c}_K$, $s(\cdot)$, and $l(\cdot)$*
Receive new file f with label y, that needs to be corrected.
$\mathbf{x} \leftarrow MalConv(f)$ *//Extract penultimate activation from MalConv*
$\hat{\mathbf{w}} \leftarrow \frac{1 - y \cdot \mathbf{w}^\intercal \mathbf{x}}{\|\mathbf{x}\|^2} \cdot y \cdot \mathbf{x}$ *//Equation 2*
$init \leftarrow$ ESTIMATEAUC$(\mathbf{w}, \mathbf{c}_1, \ldots, \mathbf{c}_K, s(\cdot), l(\cdot))$
$result \leftarrow$ ESTIMATEAUC$(\hat{\mathbf{w}}, \mathbf{c}_1, \ldots, \mathbf{c}_K, s(\cdot), l(\cdot))$
return estimated AUC impact $result - init$

Evaluation

- Microsoft Office documents that contained macros: 651,872 benign and 449,535 malicious samples
  - Stratified sample of 80% for the training set, and 20% for the test set.
- 58 difficult to detect false positives from production. “Hard FP” set.
  - 100% FP rate on production model.
  - We want to adapt model to remove these FPs, while keeping utility of detector.

Baseline Results

MalConv Embeddings +

- Passive Aggressive (PA)
- Stochastic Gradient Descent (SGD)
- Prototypes (One-shot algo)

Algorithm	Acc	AUC	$AUC_{FPR \leq .1\%}$	FPR	TPR
MalConv+PA	96.66	99.34	78.30	0.1005	58.35
MalConv+SGD	97.06	99.36	79.21	0.0997	66.18
MalConv+Prototype	60.97	64.96	50.01	13.29	86.70
GBDT	99.85	99.97	99.27	0.0930	99.65
PA	95.13	97.12	50.39	0.1006	2.310
Kernel PA	66.80	63.26	56.28	0.0999	14.87

Degenerate Solution

Domain Knowledge Feature Vectors +

- Gradient Boosted Decision Trees (GBDT)
- Passive Aggressive
- Kernelized Passive Aggressive

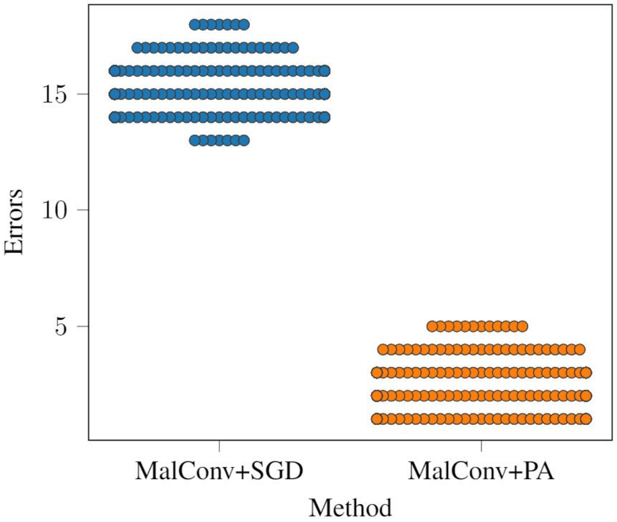
Not Accurate Enough @ Low FPR

Hard FP Results

Hard FP set feed to models in random order, updating on error as if given feedback.

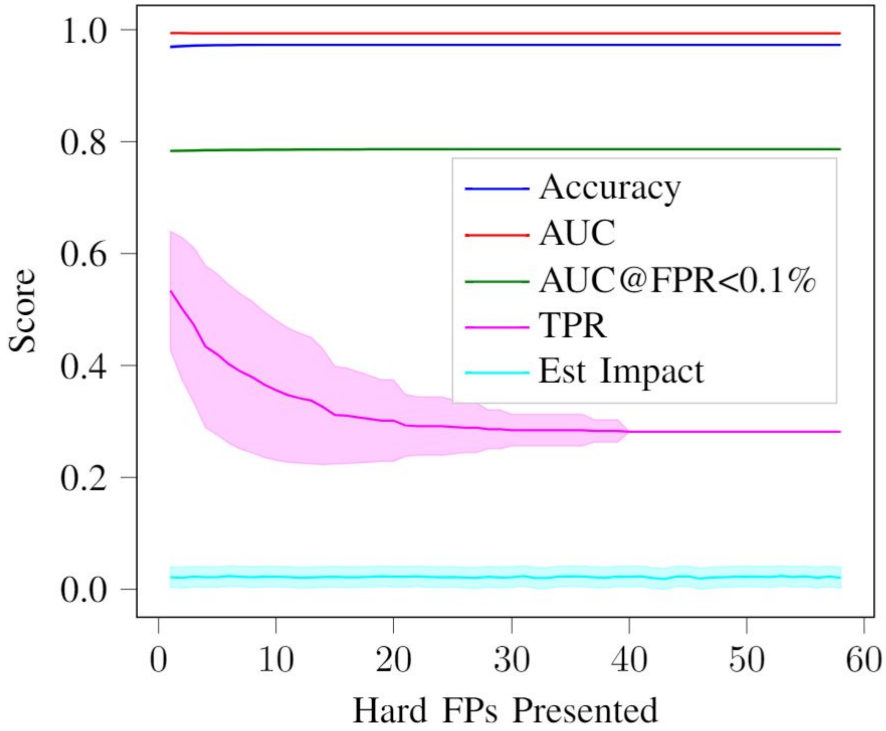
- 200 random trials to give distribution.
- PA performs best, as few as 1 update to prevent all 58 FPs!

	Hard FP Rate (%)	
Algorithm	Fixed	Adaptive
MalConv+PA	58.62%	**4.33±1.919**%
MalConv+SGD	37.93%	26.46±1.893%
GBDT	100.0%	N/A



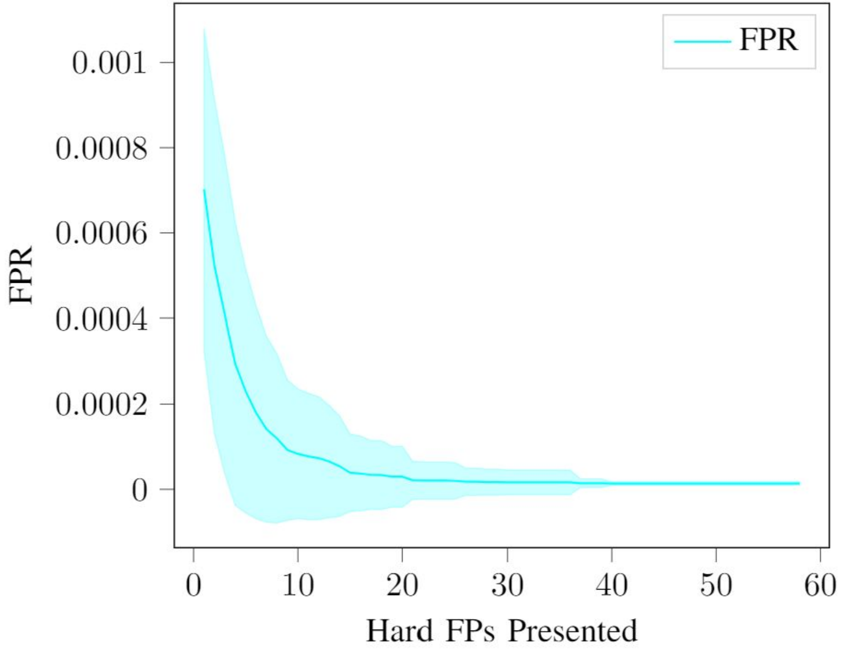
Hard FP Impact on Global Performance

- Estimated Impact to AUC low.
  - Actual impact to AUC lower than predicted
- TPR decreases by up-to 50%.
  - **No free lunch**
- How does TPR drop but AUC flat?
  - AUC is a measure based on *ranking*, not threshold.
  - Means if the users sends the model back, we can recalibrate their threshold without compromising privacy.



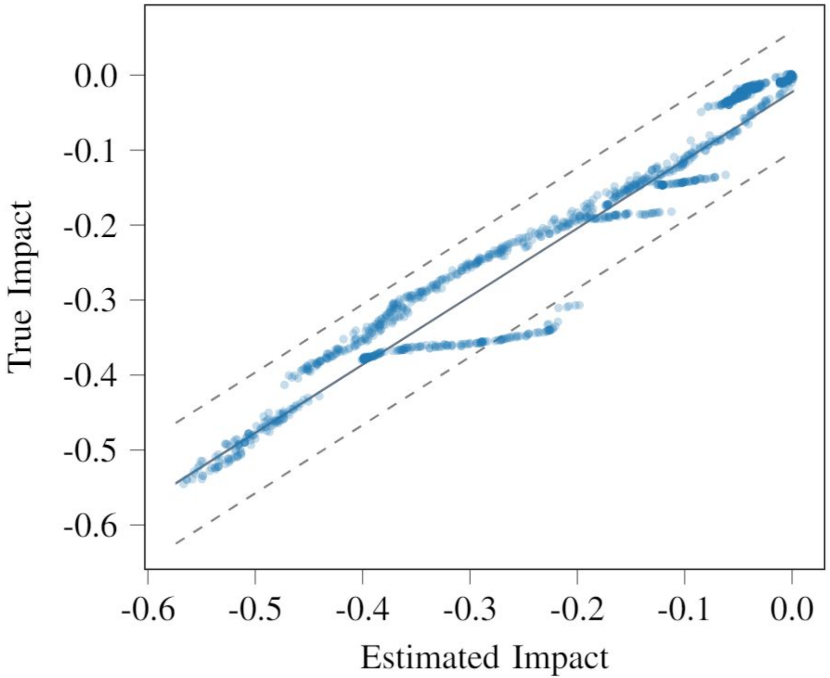
Hard FP Impact on Global Performance

TPR drops by 50%, but FPR drops by ***23x***!



Validated Estimated AUC Impact

- None of the Hard FPs are erroneous (i.e., truly malware), so not surprising that they result in low estimated impact.
- How do we know it will save us if a user does submit an erroneous update?
- Test by swapping labels on the test set, updating, and measuring against the rest of the test-set.
- Seems to work well! Estimated and actual impact have a strong linear relationship.



Take-Away

ML-backed malware detection ***will cause FPs*** in customer environments

- Current mitigation options are antiquated. (*e.g. whack-a-mole hash lists*)
- The industry needs to leverage local domain knowledge
- Humans-in-the-loop can improve global models, locally, while preserving data privacy

It is time to cultivate ***trust*** in ML-backed security by eliminating the black-box.

- Passive Aggressive approaches encourage safe customization of a local model
- Models can be safeguarded against accidental compromise by measuring the quality of adjustments

Establish transparency and trust in ML-backed security, while reducing FPs locally over time

Thank You!

Edward Raff
Raff_Edward@bah.com
Edwardraff.com

Bobby Filar
filar@elastic.co
@filar

James Holt
holt@lps.umd.edu