Understanding Update of Machine-Learning-Based Malware Detection by Clustering Changes in Feature Attributions

Yun Fan
Advanced Network Architecture Research Laboratory
Graduated School of Information Science and Technology
Osaka University

Background

- In a **malware detection (ML) system**, the statistical characteristics of malware change over time, causing the detection performance degrades
- The classification models in malware detection systems need **updates** to improve the detection performance
  - update: add new data to the training dataset and re-train the model
- After updates, the new model needs to be **validated** before deployment
  - accuracy
  - the area under the curve (AUC)
  - …

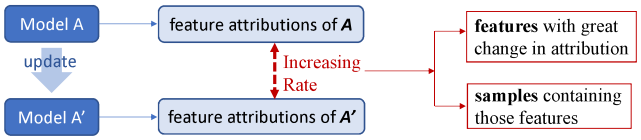
Purpose

- Common validation methods only calculate the detection accuracy or AUC scores
- When the detection performance is not satisfying after model update, we need more information to determine the cause
  - *why performance changed ?*
  - *what changes in the update affect performance?*

Purpose:
Get detailed information about model changes to understand the model updates in ML-based malware detection systems.

Proposed Method

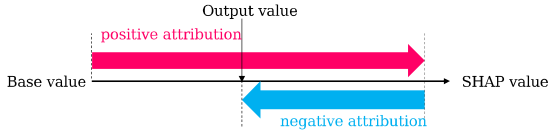
- Machine learning (ML) models are often used in malware detection systems, and **feature attributions** are typically used to explain the ML models
- We use the **feature attribution changes** to analyze model changes
- Proposed method



- We divide the samples into **clusters** based on their feature attribution changes.

Feature Attribution

- We use **Shapley additive explanations (SHAP)** to calculate the **feature attributions**
- SHAP is a **consistent** feature attribution method
  - *When the model has changed and a feature has higher impact on the model, the importance of that feature cannot be lower*
- SHAP explains the output as a sum of the effects of each feature



- **Consistency** enables comparison of attribution values across models

SHAP Value Changes

- We calculate an **increasing rate of SHAP values (*I*)** to measure a feature's attribution change in an update

$$I_{x_i} = \frac{v2_{x_i} - v1_{x_i} + c_1}{\min\left(\left|v1_{x_i}\right|, \left|v2_{x_i}\right|\right) + c_2}, \quad where\ c_2 > 0, c_1 = \begin{cases} c_2, & when\ v2_{x_i} - v1_{x_i} \geq 0, \\ -c_2, & when v2_{x_i} - v1_{x_i} < 0. \end{cases}$$

$I > 0$ → Feature attribution is higher
Samples are more likely to be classified as **positive**

$I < 0$ → Feature attribution is lower
Samples are more likely to be classified as **negative**

- When $|I| \approx 0$, the feature's effect to the model update is very low
- Identify **features** with high increasing rate by $|I| \geq k$ and analyze **samples** containing those features

Clustering

- To make the output more clearer for the operators, we divide the samples into clusters based on their feature attribution changes.
- We use **Jaccard similarity** to measure the similarity, and divide samples with high similarity as the same cluster.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

- After clustering, we selected the clusters whose average predictions changed to do the evaluation.

Experimental Setup

- Dataset
  - Android application files: *AndroZoo**
  - 90% benign samples and 10% malicious samples
- Updates:

('a' represents the firrst half of the year, 'b' represents the second half of the year)

		Train			Test		
Update 1	Pre-update	2016a	2016b	2017a	2017b	2018a	2018b
	Post-update	2016a	2016b	2017a	2017b	2018a	2018b
		Train			Test		
Update 2	Pre-update	2016a	2016b	2017a	2017b	2018a	2018b
	Post-update	2016a	2016b	2017a	2017b	2018a	2018b
		Train				Test	
Update 3	Pre-update	2016a	2016b	2017a	2017b	2018a	2018b
	Post-update	2016a	2016b	2017a	2017b	2018a	2018b

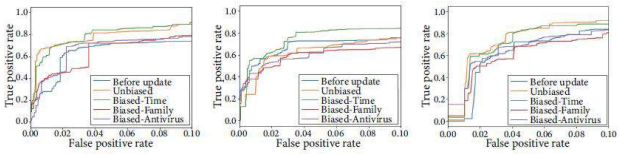
**AndroZoo: Allix, K, etc.: Androzoo: Collecting millions of android apps for the research community.(2016)*

Model Update

- To simulate successful and failed updates, we used biased and unbiased datasets to re-train the model.
  - *Unbiased*: random sampling
  - *Biased-Time*: only use the latest data
  - *Biased-Family*: only use malware from major families
  - *Biased-Antivirus*: only use malware easily detected
- Un-biased datasets are always used for pre-update training datasets

Classification Performance

- We used the testsets to investigate the classification performance:



(a) Update 1 (b) Update 2 (c) Update 3

- "Unbiased" and "Biased-Time" ⟶ successful
  "Biased-Family" and "Biased-Antivirus" ⟶ not successful

Experimental Results

- We divide the samples into different clusters based on their feature attribution changes to make the output more clearer
- Number of cluster/features and maximum order of SHAP:

		# clusters	# features in each cluster	Max. order of SHAP
Update 1	Unbiased	5	7–10	39–487
	Biased-Time	4	1–10	39–142
	Biased-Family	3	2–8	22–110
	Biased-Antivirus	3	3–9	53–218
Update 2	Unbiased	1	6	64
	Biased-Time	6	3–8	24–190
	Biased-Family	3	4–10	24–428
	Biased-Antivirus	0	-	-
Update 3	Unbiased	5	2–10	31–371
	Biased-Time	6	3–10	55–371
	Biased-Famliy	2	3–10	371
	Biased-Antivirus	1	2	198

Evaluation

- We mainly use the cluster number and cluster size to evaluate the model updates
- The main causes of a failed model update are **overfitting** and **noneffective update**.
- The results can be analyzed in three perspectives:
  - **learning a few families**
  - **overlooking some families**

  } overfitting
  - **noneffictive updates**

Evaluation

- the number of clusters ⟶ ***learning a few families***

	Unbiased	Biased-Time	Biased-Family	Biase-Antivirus
Update 1	5	4	3	3
Update 2	1	6	3	0
Update 3	5	6	2	1

- The cluster numbers of "Biased-Family" and "Biased-Antivirus" are always less than the results of other updates
- The bias of the dataset causes a lack of variety and influences the update as a result

Evaluation

- clusters whose predictions change from true to false

⟶ ***overlooking some families***

	Unbiased	Biased-Time	Biased-Family	Biase-Antivirus
Update 1	0	0	0	0
Update 2	0	0	1	0
Update 3	0	0	2	0

- Only the results of "Biased-Family" have such clusters
- We can also identify important features related to the changes:

Features	Mean rate	Order of SHAP
com.qihoo.util.appupdate.appupdateactivity	-25.66	None
com.qihoo.util.startactivity	-25.06	None
com.swiftpass.pay.activity.qqwappaywebview	-17.20	None
com.alipay.sdk.auth.authactivity	-15.54	None
blue.sky.vn.api	-14.66	None
landroid/telephony/smsmanager.sendtextmessage	-14.23	33
blue.sky.vn.mainactivity	-11.85	None
blue.sky.vn.webviewactivity	-10.42	None
blue.sky.vn.gamehdactivity	-10.33	None
com.qihoo.util.commonactivity	-8.09	None

Evaluation

- (the number of samples whose predictions become true) - (the number of samples whose predictions become false) ⟶***noneffictive updates***

	Unbiased	Biased-Time	Biased-Family	Biased-Antivirus
Update 1	103	122	31	25
Update 2	70	104	-17	0
Update 3	78	131	-27	12

- The performance of "Unbiased" and "Biased-Time" improve after update, and the performance of "Biased-Family" and "Biased-Antivirus" have very limited change or no change after update

Summary

- The evaluation result of all updates:

		Learning a few families	Overlooking some families	Noneffective update
Update 1	Unbiased	×	×	×
	Biased-Time	×	×	×
	Biased-Family	✓	×	✓
	Biased-Antivirus	×	×	✓
Update 2	Unbiased	✓	×	×
	Biased-Time	×	×	×
	Biased-Family	✓	✓	✓
	Biased-Antivirus	✓	×	✓
Update 3	Unbiased	×	×	×
	Biased-Time	×	×	×
	Biased-Family	✓	✓	✓
	Biased-Antivirus	✓	×	✓

Conclusion and Future Works

- Conclusion
  - Our method can identify the unexpected model changes such as overfitting or noneffective update caused by the biased.
  - Our method can identify the important features relevant to the performance change.
- Future works
  - We need a user study.