CADE: Detecting and Explaining Concept Drift Samples for Security Applications

USENIX Security 2021

Limin Yang¹, Wenbo Guo², Qingying Hao¹, Arridhana Ciptadi³, Ali Ahmadzadeh³, Xinyu Xing², Gang Wang¹

¹University of Illinois at Urbana-Champaign  ²Penn State  ³Blue Hexagon Inc.
A Multi-class Malware Classification Model

1. Train

Benign  Malware A  Malware B
A Multi-class Malware Classification Model

1. Train

2. Predict

“Benign”
A Multi-class Malware Classification Model

1. Train

Benign  Malware A  Malware B

2. Predict

Malware A

“Malware A”
A Multi-class Malware Classification Model

1. Train

2. Predict

Benign  Malware A  Malware B

Malware C

“Benign”? “Malware A”? “Malware B”?

Concept Drift! (Unseen family)
Another Type of Drift: In-class Evolution

Train

Benign

Malware A

Malware B

Predict

Benign Variant

Malware A Variant

Malware B Variant
Why Concept Drift Matters?

• New attacks (zero-day) are NOT trivial
Why Concept Drift Matters?

• New attacks (zero-day) are NOT trivial

• Both malware and goodware evolve over time

Why Concept Drift Matters?

• New attacks (zero-day) are NOT trivial

• Both malware and goodware evolve over time

• ML models’ decision boundaries shift

GBDT malware classifier trained on Ember-2018; Tested a year later using malware samples from Blue Hexagon Inc. [DLS’21]

* GBDT: Gradient Boosted Decision Tree
When **NOT** to Predict?

**Goals**

1. **Detect** drifting samples
2. **Find** a small subset of important features that explain why the drifting sample is different from training data
Existing Drifting Detection Solutions

- Solutions from ML community
  - JSTOR’54, SBIA’04, SDM’07, ICML’14, KDD’16, CIKM’19, etc.
  - Most require data labeling on the testing set → costly for security domain

- Solutions from security community
  - Transcend [USENIX Sec’17]
  - Highly dependent on a good definition of “dissimilarity”, scalability issues
Why It’s Hard to Define a “Good” Distance Function?

• Distance loses effectiveness in high-dimensional data
  – This sample feature space has 1,368 dimensions

• Drifting samples are not labeled, hard to differentiate from normal samples

T-SNE plot for the original space of an Android malware dataset (Unseen family: )
Self-supervision: Contrastive Learning

• No knowledge about future drifting samples → self-supervision

• Use contrastive learning to learn a compressed representation of the training data by contrasting with existing samples

• A sample is far away from ANY existing families’ centroids, it’s a potential drifting sample; rank for investigation
How to explain these drifting samples?

- Download adware
- Read settings
- Read Contacts

Goodware  Malware

Analyst makes decision
A Rich set of Explanation Methods

• Identify a small set of important features that make the drifting sample an outlier

• Naïve idea: boundary-based explanation
  – Approximate the decision boundary between drift and in-distribution
  – Explaining a supervised learning model
    o LIME [KDD’16], SHAP [NeurIPS’17], LEMNA [CCS’18], Perturbation [ICCV’17]
  – Using “crossing the boundary” as a signal to derive important features
Problems with Boundary-based Explanation

• Difficult to approximate the boundary
  – Drifting samples are limited

• Difficult to drag a drifting sample to cross the boundary
  – Drifting samples are far away from the boundary in the sparse area
Our Method: Distance-based Explanation

- Perturb the original features and observe the distance changes in latent space

- Perturbation strategy
  - Replace $x_t$’s feature value with those of a reference sample $x_r$
  - $x_r$ is closest to the centroid of nearest family

- Optimization goal
  - Minimize the distance between $x_t$ and $C_A$
  - Use elastic-net regularization to minimize the number of selected important features
Evaluation: Datasets

Drebin [NDSS’14]
- Top 8 malware families
- 3,317 malware samples
- Training set: 80% of 7 families
- Testing set: 20% of 7 families + unseen family

IDS2018 [ICISSP’18]
- Benign + 3 types of network intrusion
- 130,702 network flows
- Training set: 80% of 3 families
- Testing set: 20% of 3 families + unseen family
# Drift Detection Results

Iteratively choose a family as the unseen family and report the average results here.

<table>
<thead>
<tr>
<th>Method</th>
<th>Drebin (Avg±Std)</th>
<th>IDS2018 (Avg±Std)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F_1$</td>
<td>Norm. Effort</td>
</tr>
<tr>
<td>Vanilla AE</td>
<td>0.72±0.15</td>
<td>1.48±0.31</td>
</tr>
<tr>
<td>Transcend</td>
<td>0.80±0.12</td>
<td>1.29±0.45</td>
</tr>
<tr>
<td>CADE</td>
<td>0.96±0.03</td>
<td>1.00±0.09</td>
</tr>
</tbody>
</table>

Real-world test: evaluate on Blue Hexagon PE malware dataset, still effective!

* Vanilla AE: Standard Autoencoder without contrastive learning.
Why CADE Works?

T-SNE visualization for Drebin dataset (Unseen family: Iconosys)
Drift Explanation: Case Study

Drifting sample family: **FakeDoc**; closest family: **GingerMaster**

- Key difference: FakeDoc usually subscribes to premium services via SMS.

```java
[api_call::android/telephony/SmsManager;->sendTextMessage], [call::readSMS], [permission::android.permission.DISABLE_KEYGUARD],
[permission::android.permission.RECEIVE_SMS], [permission::android.permission.SEND_SMS], [permission::android.permission.WRITE_SMS],
[real_permission::android.permission.SEND_SMS], [permission::android.permission.READ_SMS], [feature::android.hardware.telephony],
[permission::android.permission.READ_CONTACTS], [real_permission::android.permission.READ_CONTACTS],
[api_call::android/location/LocationManager;->isProviderEnabled], [api_call::android/accounts/AccountManager;->getAccounts],
[intent::android.intent.category.HOME], [feature::android.hardware.location.network], [real_permission::android.permission.RESTART_PACKAGES],
[real_permission::android.permission.WRITE_SETTINGS], [api_call::android/net/ConnectivityManager;->getAllNetworkInfo],
[api_call::android/net/wifi/WifiManager;->setWifiEnabled], [api_call::org/apache/http/impl/client/DefaultHttpClient],
[permission::android.permission.CHANGE_WIFI_STATE], [real_permission::android.permission.ACCESS_WIFI_STATE],
[real_permission::android.permission.BLUETOOTH], [real_permission::android.permission.BLUETOOTH_ADMIN], [call::setWifiEnabled].
```

Important features selected by CADE (avg # of selected features is 45 out of 1000+)
Takeaways

• Concept drift is a critical problem for ML/Security applications

• Contrastive Autoencoder is effective to detect concept drift

• Distance-based explanation is more suitable for explaining drifting samples
Thank you!

Homepage
https://liminyang.web.Illinois.edu

Code, features, and supplemental materials available
https://github.com/whyisyoung/CADE

A new PE malware dataset [DLS’21]
https://whyisyoung.github.io/BODMAS/
Backup Slides
Drift Detection in the Latent Space

- Drift detection: if a sample is far away from ANY existing families’ centroid, it’s a potential drifting sample
- But different families’ tightness vary, how to set distance thresholds?
- MAD (Median Absolute Deviation) \[^1\]
  - Median of the median distance to the centroid
  - \(MAD = b \times \text{median}(|X_i - \text{median}(X)|)\)
  - \(X\) is a set of distances to the centroid
  - Any new data outside \(\text{median}(X) \pm A \times MAD\) → outlier
- Rank drifting samples for investigation

\[^1\] Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median, Journal of Experimental Social Psychology, 2013
Drift Detection: Evaluation Metrics

- Precision = \( \frac{\text{detected unseen family samples}}{\text{inspected samples}} \)
- Recall = \( \frac{\text{detected unseen family samples}}{\text{total # of unseen family samples}} \)
- \( F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \)
- Normalized Inspection effort = \( \frac{\text{inspected samples}}{\text{total # of unseen family samples}} \)

Training set: A (200), B(200), C(200)
Testing set: A(50), B(50), C(50), D(10)

<table>
<thead>
<tr>
<th>ID</th>
<th>Family</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D</td>
<td>1/1 = 1</td>
<td>1/10 = 0.1</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>2/2 = 1</td>
<td>2/10 = 0.2</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>2/3 = 0.67</td>
<td>2/10 = 0.2</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>3/4 = 0.75</td>
<td>3/10 = 0.3</td>
</tr>
<tr>
<td>5</td>
<td>D</td>
<td>4/5 = 0.8</td>
<td>4/10 = 0.4</td>
</tr>
</tbody>
</table>

......

A ranked list of detected samples
Real-world Test on Blue Hexagon PE Malware

• 20,613 Windows PE malware, 395 families, Sept. 2019 – Feb. 2020
• 2,381 features, training with top N families Sept. 2019 – Jan. 2020
• Testing set: Feb. 2020
• CADE is still effective!

<table>
<thead>
<tr>
<th>N (training families)</th>
<th>F_1</th>
<th>Norm. Effort</th>
<th>Detected Unseen Families</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.97</td>
<td>1.02</td>
<td>161/165</td>
</tr>
<tr>
<td>10</td>
<td>0.95</td>
<td>0.98</td>
<td>153/160</td>
</tr>
<tr>
<td>15</td>
<td>0.87</td>
<td>0.84</td>
<td>140/155</td>
</tr>
</tbody>
</table>
Drift Explanation: Evaluation Metrics and Results

- Metric: the latent distance between a perturbed sample and its closest centroid.
- Using CADE to select important features, $x_t \rightarrow x_p$, while baseline methods may $\rightarrow x_p'$, which is still far away from $C_A$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Drebin-FakeDoc Avg±Std</th>
<th>IDS2018-Infiltration Avg±Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original distance</td>
<td>5.363±0.568</td>
<td>11.715±2.321</td>
</tr>
<tr>
<td>Random</td>
<td>5.422±1.773</td>
<td>11.546±3.169</td>
</tr>
<tr>
<td>Boundary-based</td>
<td>3.960±2.963</td>
<td>6.184±3.359</td>
</tr>
<tr>
<td>COIN [IJCAI’18]</td>
<td>6.219±3.962</td>
<td>8.921±2.234</td>
</tr>
<tr>
<td>CADE</td>
<td>0.065±0.035</td>
<td>2.349±3.238</td>
</tr>
</tbody>
</table>