



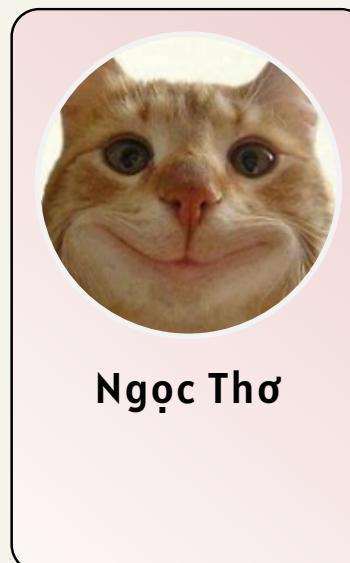
UIT

## CƠ CHẾ HOẠT ĐỘNG CỦA MÃ ĐỘC - NT230.021.ANTN

# S45 - DROIDRL: FEATURE SELECTION FOR ANDROID MALWARE DETECTION WITH REINFORCEMENT LEARNING

GROUP 01

**GVHD:**  
**TS. Phạm Văn Hậu**  
**ThS. Phan Thế Duy**



GROUP 01

# NỘI DUNG

- Chủ đề
- Liên kết
- Bài báo
- Tóm tắt nội dung chính
- Các kỹ thuật chính
- Môi trường thực nghiệm của bài báo
- Kết quả thực nghiệm của bài báo
- Triển khai
- Kết quả
- Khó khăn
- Cải tiến
- Demo

# CHỦ ĐỀ NGHIÊN CỨU TRONG LĨNH VỰC MÃ ĐỘC

Phát hiện mã độc



Đột biến mã độc

Khác: ...

# LIÊN KẾT

Mã nguồn của đề tài đồ án được lưu tại:  
[https://github.com/Roses21/-  
NT230.02I.ANTN--Malware](https://github.com/Roses21/-NT230.02I.ANTN--Malware)



# BÀI BÁO

Wu, Y., Li, M., Zeng, Q., Yang, T.,  
Wang, J., Fang, Z., & Cheng, L.  
(2023). DroidRL: Feature  
selection for android malware  
detection with reinforcement  
learning. *Computers & Security*,  
128, 103126.

Tiếng anh

DroidRL: Feature selection for  
android malware detection with  
reinforcement learning.

Tiếng việt

DroidRL: Lựa chọn đặc trưng để  
phát hiện phần mềm độc hại trên  
Android với tính năng học tăng  
cường.

# TÓM TẮT NỘI DUNG CHÍNH

1

DroidRL là một framework được thiết kế để chọn lựa các đặc trưng hiệu quả nhằm phát hiện malware trên hệ điều hành Android bằng cách sử dụng học tăng cường. Bài báo tập trung vào việc áp dụng Double Deep Q-Network (DDQN) kết hợp với mạng nơ-ron hồi quy (RNN) để lựa chọn các đặc trưng theo trình tự.

2

Với việc sử dụng phương pháp nhúng từ (word embedding) để biểu diễn đối tượng, DroidRL có thể tăng cường sự liên quan ngữ nghĩa giữa các đặc trưng.

3

Các kết quả thực nghiệm cho thấy DroidRL có thể giảm đáng kể số lượng đặc trưng từ 1083 xuống còn 24 mà vẫn giữ độ chính xác cao (95.6%) khi sử dụng bộ phân loại Random Forest.

# CÁC KỸ THUẬT CHÍNH

1

DroidRL triển khai thuật toán DDQN - thuật toán học tăng cường Reinforcement learning (Double Deep Q - Network) ⇒ thu được tập hợp con đặc trưng tối ưu ⇒ phân loại malware hiệu quả.

2

Mạng thần kinh tái phát (RNN - Recurrent NN) được sử dụng làm mạng quyết định của DDQN ⇒ cung cấp khả năng chọn tuần tự các đặc trưng.

3

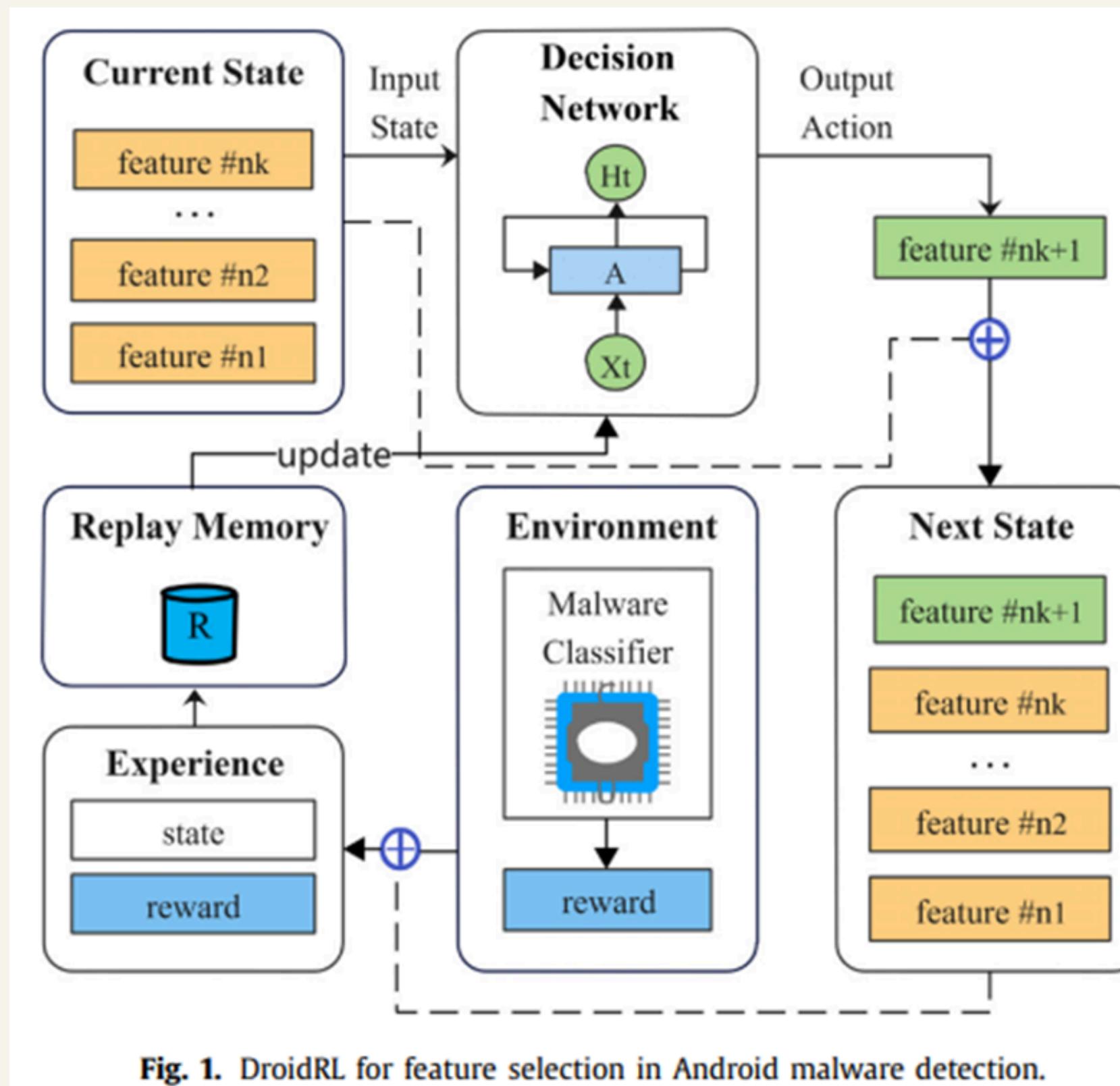
Word embedding được áp dụng để biểu diễn đối tượng ⇒ nâng cao khả năng làm việc của framework nhằm tìm ra mức độ liên quan về mặt ngữ nghĩa của các đối tượng.

4

Sử dụng Random Forest làm lớp phân loại ⇒ đánh giá hiệu suất.

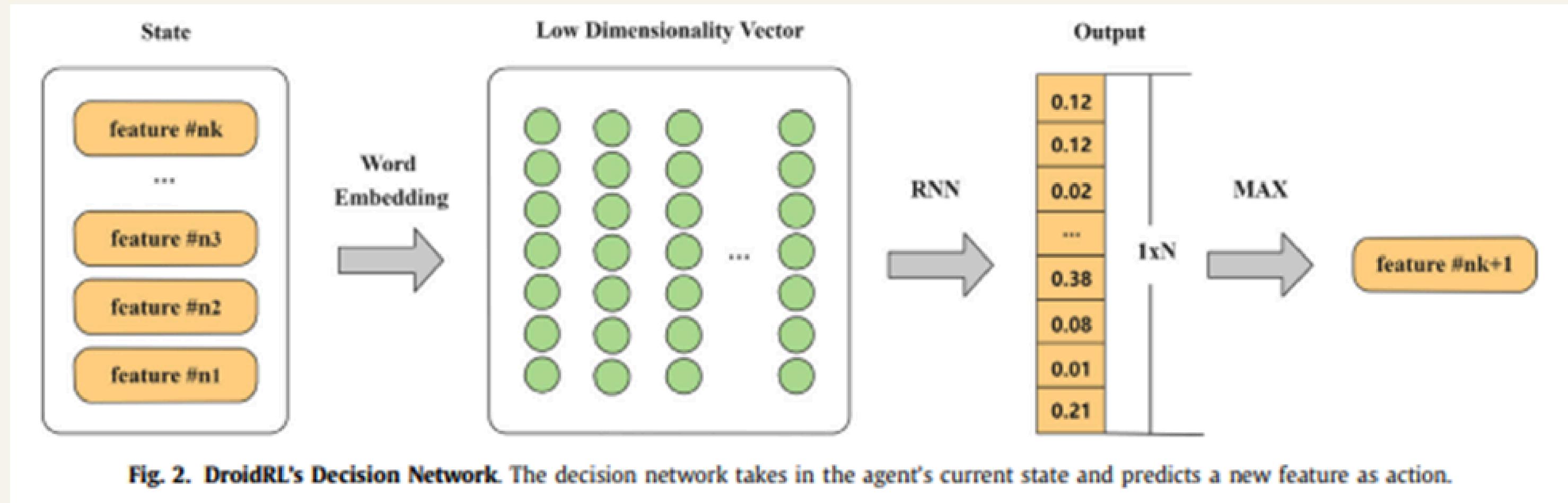
# CÁC KỸ THUẬT CHÍNH

7



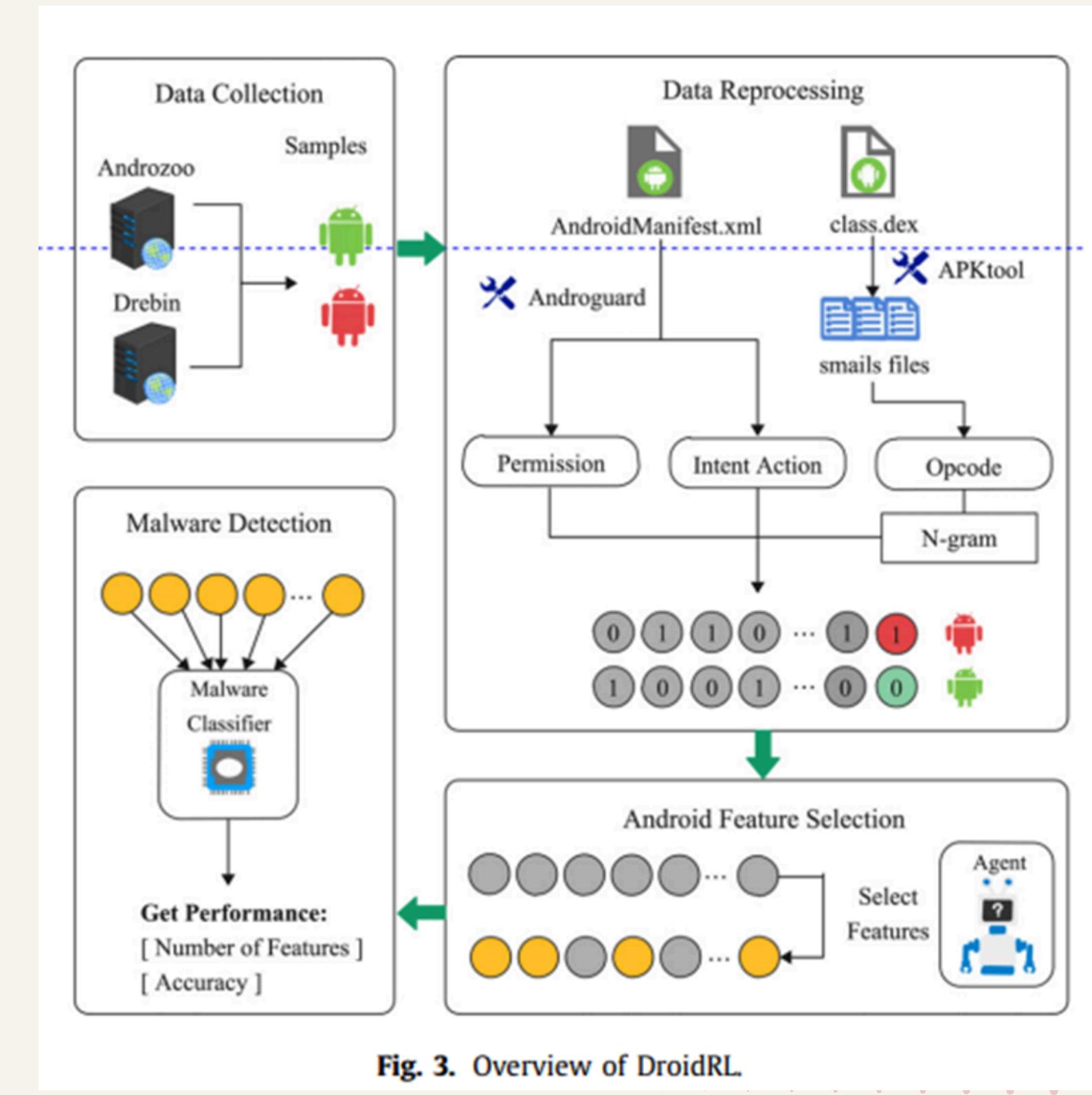
- Phần cốt lõi của framework DroidRL được xây dựng bởi mạng quyết định dựa trên DDQN).
  - Trong mỗi bước, tác nhân tự trị thực hiện một cách độc lập một hành động do mạng quyết định quyết định để chọn một đặc trưng vào trạng thái được quan sát từ môi trường bằng cách sử dụng kiến thức trước đó.
  - Để đánh giá chất lượng của tập hợp con đặc trưng và sự khác biệt của các đặc trưng riêng lẻ đã chọn, reward của hành động được tạo ra từ trình phân loại phần mềm độc hại được xác định bằng độ chính xác của phân loại phần mềm độc hại sử dụng các đặc trưng được chọn làm đầu vào.
  - Hơn nữa, trạng thái của tác nhân, hành động được chọn và reward của thời điểm này được lưu trong bộ nhớ phát lại để huấn luyện mạng quyết định.
  - Chính sách thăm dò-khai thác (exploration-exploitation) được tăng cường để giải quyết vấn đề chi phí tính toán do các tập hợp đặc trưng không thể cạn kiệt.

# CÁC KỸ THUẬT CHÍNH



- cấu trúc mạng quyết định cuối cùng được hiển thị trong Hình 2, tác nhân đưa trạng thái của nó, một chuỗi đại diện cho các đặc trưng đã chọn, vào mạng quyết định.
- Lớp đầu tiên của mạng quyết định là lớp nhúng.
- Các đặc trưng được biểu thị bằng one-hot vector đi qua lớp nhúng và trở thành dense vector.
- Các vectơ này sau đó được đưa vào mạng giống RNN và cuối cùng đi vào lớp fully connected và lớp softmax.

# CÁC KỸ THUẬT CHÍNH



# MÔI TRƯỜNG THỰC NGHIỆM CỦA BÀI BÁO

- Cấu hình máy tính: CPU: Intel Core i7
- RAM: 16GB
- GPU: NVIDIA GTX 1080
- Các công cụ hỗ trợ sẵn có:
  - TensorFlow (dùng cho việc triển khai mạng nơ-ron)
  - Scikit-learn (dùng cho việc đánh giá mô hình)

# MÔI TRƯỜNG THỰC NGHIỆM CỦA BÀI BÁO

- Ngôn ngữ lập trình: Python
- Đối tượng nghiên cứu (chương trình phần mềm dùng để kiểm tra tính khả thi của phương pháp/tập dữ liệu – nếu có): Tập dữ liệu của DroidRL chứa 5000 benign sample từ Andro Zoo và 5560 malware từ Drebin để huấn luyện và kiểm tra mô hình.
- Tiêu chí đánh giá tính hiệu quả của phương pháp:
  - Số lượng đặc trưng được chọn
  - Độ chính xác của việc phát hiện malware (accuracy)
  - Thời gian chạy

# KẾT QUẢ THỰC NGHIỆM CỦA BÀI BÁO

DroidRL đã chứng minh khả năng lựa chọn đặc trưng hiệu quả, giảm từ 1083 xuống còn 24 đặc trưng, đồng thời duy trì độ chính xác phát hiện malware ở mức 95.6%. Phương pháp này không chỉ tiết kiệm thời gian tính toán mà còn tăng cường hiệu suất của bộ phân loại. DroidRL được chứng minh là có hiệu quả trong các nhiệm vụ lựa chọn đặc trưng và hy vọng nó sẽ đóng vai trò là một yếu tố để xây dựng một trình phát hiện malware mạnh mẽ trong tương lai.

# KẾT QUẢ THỰC NGHIỆM CỦA BÀI BÁO

**Table 3**

Training time ratio.

Classifier/ Ratio(%) <sup>1</sup> / Features Used	16	17	18	19	20	21	22	23	24
<b>Decision Tree</b>	1.23	1.64	1.36	1.78	1.63	1.62	1.91	2.06	2.78
<b>Random Forest</b>	15.07	16.64	15.66	17.35	16.64	16.36	16.34	16.85	17.81
<b>Support Vector Machine</b>	3.53	4.22	4.13	4.50	4.51	4.49	4.79	4.74	4.88

1 The Ratio represents the percentage of time consumed for training with a subset of features versus training with all (1083) features

**Table 4**

Comparison with detection approaches without feature selection.

Method	Number of Features	Accuracy	ML Based Detection Model
MamaDroid (Onwuzurike et al., 2019)	190,096	0.989	Random Forest
DroidDet (Zhu et al., 2018)	3122	0.921	Rotation Forest
HMMDDetector (Canfora et al., 2016)	-	0.871	Hidden Markov Model, Random Forest
Drebin (Arp et al., 2014a)	545,356	0.881	Support Vector Machine
<b>DroidRL (ours)</b>	<b>24</b>	<b>0.956</b>	<b>Random Forest</b>

# TRIỂN KHAI

<b>Index of /CICDataset</b>			
	<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size Description</a>
	<a href="#">Parent Directory</a>	-	-
	<a href="#">CIC-EvasivePDF2022/</a>	2024-03-27 09:38	-
	<a href="#">CIC-IDS-2017/</a>	2024-02-02 09:17	-
	<a href="#">CICAndAdGMal2017/</a>	2024-02-02 09:49	-
	<a href="#">CICAndMal2020--/</a>	2020-08-13 23:22	-
	<a href="#">CICAndMal2020/</a>	2024-02-02 20:50	-
	<a href="#">CICBellDNS2021/</a>	2024-02-02 09:59	-
	<a href="#">CICBellEXFDNS2021/</a>	2024-02-02 10:04	-
	<a href="#">CICDDoS2019/</a>	2024-02-02 09:40	-
	<a href="#">CICDarknet2020/</a>	2024-02-02 21:03	-
	<a href="#">CICDataset_Organized/</a>	2024-02-01 15:24	-
	<a href="#">CICEV2023/</a>	2024-02-01 15:33	-
	<a href="#">CICMalAnal2017/</a>	2024-02-02 20:56	-

Thu thập bộ dữ liệu bao gồm 1514 mẫu lành tính và 1054 mẫu độc hại từ [Index of /CICDataset](#) (CICAndMall2017 và CICAndMall2020), nhóm em chỉ lấy một số cái có thể tải được và phù hợp với tài nguyên hiện có.

# TRIỂN KHAI

Thực hiện trích xuất đặc trưng thô từ các tệp APK, kết quả cho ra tệp JSON.

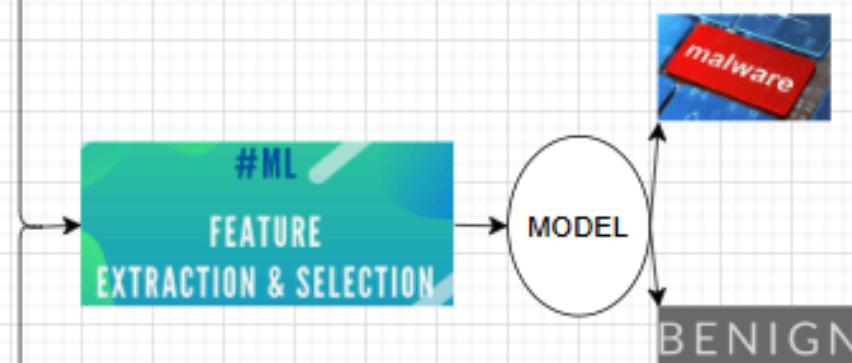
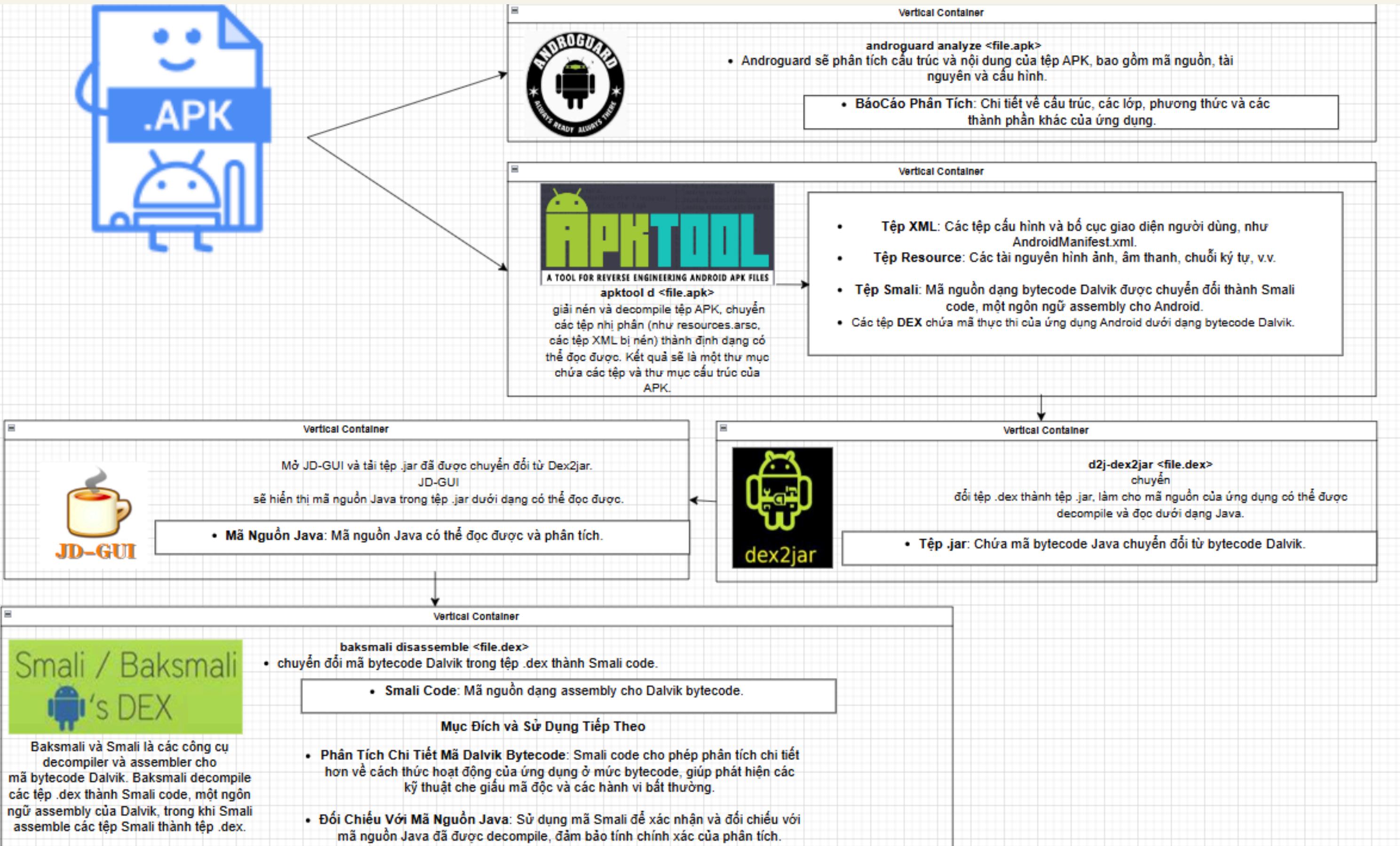
```
: !python3 droidlysis/droidlysis3.py --input test/benign --output ./test/output --config droidlysis/conf/general.conf

Processing file: test/benign/air.beingfashiondesigner.apk ...
Launching unzip process on test/benign/air.beingfashiondesigner.apk with output dir=./test/output/air.beingfashiondesigner.apk-7ce60a541c7d73c30181e4840b58dc348d2aa6baddcba0e124e91dd1ed2f1f4/unzipped
caution: filename not matched: classes2.dex
===== Report =====
Sanitized basename : air.beingfashiondesigner.apk
SHA256             : 7ce60a541c7d73c30181e4840b58dc348d2aa6baddcba0e124e91dd1ed2f1f4
File size          : 19673487 bytes
Is small           : False
Nb of classes     : 7830
Nb of dirs         : 264

Certificate properties
algo              : None
serialno          : None
country            : 500
owner              : None
timestamp          : (2016, 6, 6, 11, 21, 6)
year               : 2016

Manifest properties
activities        : ['.AppEntry', 'com.chartboost.sdk.CBImpressionActivity', 'com.google.android.gms.ads.AdActivity']
main_activity      : .AppEntry
package_name       : air.beingfashiondesigner
permissions         : ['INTERNET', 'READ_PHONE_STATE', 'ACCESS_NETWORK_STATE', 'DISABLE_KEYGUARD', 'WAKE_LOCK', 'ACCESS_FINE_LOCATION', 'ACCESS_COARSE_LOCATION', 'WRITE_EXTERNAL_STORAGE']
swf                : True
```

# TRIỂN KHAI



# TRIỂN KHAI

Viết mã trích xuất đặc trưng từ tệp JSON đã có theo chuẩn nhất định.

```
def get_static_feature(data):
    values = dict()
    for x in data.keys():
        if x == "sanitized_basename":
            continue
        if isinstance(data[x], dict):
            for key in data[x].keys():
                if isinstance(data[x][key], list):
                    #print(x, key)
                    if key == "permissions":
                        for perm in data[x][key]:
                            values[f'permissions_{perm.split(".")[-1]}'] = 1

                    values[f'{x}_{key}_len'] = len(data[x][key])
                    continue
                if key in ['main_activity', 'package_name', 'app_name']:
                    continue
                try:
                    values[f'{x}_{key}'] = int(data[x][key])
                except:
                    values[f'{x}_{key}'] = -1
            else:
                values[x] = int(data[x])
    return values
```

# TRIỂN KHAI

Viết mã lấy dữ liệu từ các mẫu lành tính và các mẫu độc hại và thực hiện gán nhãn cho các mẫu (lành tính là 0, độc hại là 1).

```
import os, json
import pandas as pd

def get_dataset(folder, lable):
    dataset = dict()
    for filename in os.listdir(folder):
        dataset[filename] = get_static_feature(json.load(open(f'{folder}/{filename}')))
    df=pd.DataFrame(dataset).T
    df.fillna(value=0, inplace=True)
    df['class'] = lable
    return df

df_ben = get_dataset("malware/malware_json", 1)
df_mal = get_dataset("benign/benign_json", 0)
```

df\_ben.shape

(1514, 1256)

df\_mal.shape

(1054, 1383)

# TRIỂN KHAI

Chuyển các dữ liệu đã có thành dataframe và ghép lại thành một bộ dữ liệu để huấn luyện và kiểm tra.

```
# Kết hợp hai DataFrame
df_combined = pd.concat([df_ben, df_mal], ignore_index=True)
df_combined.fillna(value=0, inplace=True)

# Trộn ngẫu nhiên các hàng
HinT = df_combined.sample(frac=1).reset_index(drop=True)

HinT.head(5)
```

	file_nb_classes	file_nb_dir	file_size	file_small	filetype	file_innerzips	manifest_properties_activities_len	manifest_properties_lit
0	41.0	6.0	1612197.0	0.0	1.0	0.0	8.0	
1	0.0	0.0	836367.0	0.0	1.0	0.0	11.0	
2	465.0	30.0	561174.0	0.0	1.0	0.0	17.0	
3	178.0	29.0	172599.0	0.0	1.0	0.0	2.0	
4	4691.0	138.0	7885466.0	0.0	1.0	0.0	9.0	

5 rows × 1436 columns

# TRIỂN KHAI

Tiến hành huấn luyện và đánh giá trên 4 mô hình là **RandomForestClassifier**, **LGBMClassifier**, **ExtraTreesClassifier**, **DecisionTreeClassifier** cho toàn bộ tất cả các feature trích xuất được (1435 đặc trưng). Đánh giá về **Accuracy**, **Precision**, **Recall**, **F1**, **ROC-AUC**, **Time**.

```
import time
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from lightgbm import LGBMClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Initialize models
rf_model = RandomForestClassifier(random_state=42)
lgbm_model = LGBMClassifier(random_state=42)
et_model = ExtraTreesClassifier(random_state=42)
dt_model = DecisionTreeClassifier(random_state=42)
```

# TRIỂN KHAI

```
def evaluate_model(model, X_train, y_train, X_test, y_test):
    start_time = time.time()
    model.fit(X_train, y_train)
    end_time = time.time()

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])

    return accuracy, precision, recall, f1, roc_auc, end_time - start_time

# Create a dictionary to store results
results = {'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1': [], 'ROC-AUC': [], 'Time': []}
```

# TRIỂN KHAI

```
# Loop through each model
for model, model_name in zip([rf_model, lgbm_model, et_model, dt_model],
                             ['Random Forest', 'LightGBM', 'Extra Trees', 'Decision Tree']):

    # Evaluate model performance and time
    accuracy, precision, recall, f1, roc_auc, run_time = evaluate_model(model, X_train, y_train, X_test, y_test)

    # Store results in the dictionary
    results['Model'].append(model_name)
    results['Accuracy'].append(round(accuracy, 4))
    results['Precision'].append(round(precision, 4))
    results['Recall'].append(round(recall, 4))
    results['F1'].append(round(f1, 4))
    results['ROC-AUC'].append(round(roc_auc, 4))
    results['Time'].append(round(run_time, 4))

# Create a performance evaluation/comparison table
results_df = pd.DataFrame(results)
print(results_df)
```

	Model	Accuracy	Precision	Recall	F1	ROC-AUC	Time
0	Random Forest	0.9922	0.9934	0.9934	0.9934	0.9998	0.2075
1	LightGBM	0.9981	1.0000	0.9967	0.9983	1.0000	0.2264
2	Extra Trees	0.9961	0.9967	0.9967	0.9967	0.9999	0.2986
3	Decision Tree	0.9883	0.9934	0.9868	0.9901	0.9887	0.0296

# TRIỂN KHAI

Thực hiện phương pháp lựa chọn đặc trưng (lựa chọn được 323/1435 đặc trưng)

```
def evaluate_one_feature(feature, index='', metric=roc_auc_score):
    rootnode = DecisionTreeClassifier(max_depth=1, criterion='gini')
    rootnode.fit(X_train[feature].array.reshape(-1,1), y_train)
    preds = rootnode.predict(X_test[feature].array.reshape(-1,1))
    preds_tr = rootnode.predict(X_train[feature].array.reshape(-1,1))
    met = round(metric(y_test, preds), 4)
    if met > 0.5:
        return [feature, met, rootnode, preds, preds_tr]
    else:
        return [feature, met, None, [], []]
```

- ROC AUC được sử dụng làm chỉ số đánh giá hiệu suất của mô hình. Điều này giúp đảm bảo rằng mô hình có khả năng phân biệt tốt giữa các lớp dương tính và âm tính.
- Tiêu chí Gini được sử dụng trong mô hình cây quyết định để tìm ra các điểm phân chia tối ưu, giúp đánh giá chính xác tầm quan trọng của đặc trưng được kiểm tra.

# TRIỂN KHAI

Thực hiện phương pháp lựa chọn đặc trưng (lựa chọn được 323/1435 đặc trưng)

```
from joblib import Parallel, delayed
import multiprocessing

# Sử dụng joblib để chạy hàm evaluate_one_feature song song trên các đặc trưng
def parallel(f, items, n_workers=None, threadpool=False, progress=True):
    if n_workers is None:
        n_workers = multiprocessing.cpu_count()

    if threadpool:
        from multiprocessing.pool import ThreadPool as Pool
    else:
        from multiprocessing import Pool

    with Pool(n_workers) as pool:
        results = list(pool imap(f, items))

    return results

# "conts` là danh sách các đặc trưng muốn đánh giá
results = parallel(f=evaluate_one_feature,
                    items=conts, n_workers=multiprocessing.cpu_count(), threadpool=False, progress=True)

useful_features = result_df.loc[result_df['roc_auc_score'] > 0.5]
print(f"{len(useful_features)} / {len(conts)} features have direct separating power (linear)")

323 / 1435 features have direct separating power (linear)
```

# TRIỂN KHAI

Tiến hành huấn luyện và đánh giá lại 4 mô hình đã thiết lập dựa trên bộ dữ liệu đã được tiến hành phương pháp chọn đặc trưng. Đánh giá về Accuracy, Precision, Recall, F1, ROC-AUC, Time.

## Create a new data ¶

```
# Create new dataset with selected features
selected_features = useful_features['feature'].tolist()
X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]
```

# TRIỂN KHAI

Tiến hành huấn luyện và đánh giá lại 4 mô hình đã thiết lập dựa trên bộ dữ liệu đã được tiến hành phương pháp chọn đặc trưng. Đánh giá về Accuracy, Precision, Recall, F1, ROC-AUC, Time.

```
import pandas as pd
results_df = pd.DataFrame(results)
print(results_df)

      Model  Accuracy  Precision  Recall    F1  ROC-AUC  Time
0  Random Forest   0.9202     1.0000  0.8647  0.9274  0.9999  0.1777
1      LightGBM    0.9942     1.0000  0.9901  0.9950  1.0000  0.1028
2   Extra Trees    0.9319     1.0000  0.8845  0.9387  1.0000  0.1363
3  Decision Tree   0.9864     0.9901  0.9868  0.9884  0.9863  0.0187

# Lưu tên các đặc trưng sử dụng trong quá trình huấn luyện
feature_names = X_train_selected.columns.tolist()
joblib.dump(feature_names, 'feature_names.joblib')

['feature_names.joblib']

len(feature_names)

323
```

# KẾT QUẢ

**Kết quả cho thấy, sau khi thực hiện phương pháp lựa chọn đặc trưng, thì chi phí tính toán và thời gian đã giảm đi, độ chính xác tuy giảm nhưng mức độ tuỳ thuộc vào thuật toán sử dụng để phân loại. Như mô hình LGBM cho thấy tất cả các chỉ số đánh giá đều không chênh lệnh nhiều so với lúc chưa giảm số lượng đặc trưng.**

	Model	Accuracy	Precision	Recall	F1	ROC-AUC	Time
0	Random Forest	0.9922	0.9934	0.9934	0.9934	0.9998	0.2075
1	LightGBM	0.9981	1.0000	0.9967	0.9983	1.0000	0.2264
2	Extra Trees	0.9961	0.9967	0.9967	0.9967	0.9999	0.2986
3	Decision Tree	0.9883	0.9934	0.9868	0.9901	0.9887	0.0296

	Model	Accuracy	Precision	Recall	F1	ROC-AUC	Time
0	Random Forest	0.9202	1.0000	0.8647	0.9274	0.9999	0.1777
1	LightGBM	0.9942	1.0000	0.9901	0.9950	1.0000	0.1028
2	Extra Trees	0.9319	1.0000	0.8845	0.9387	1.0000	0.1363
3	Decision Tree	0.9864	0.9901	0.9868	0.9884	0.9863	0.0187

# KHÓ KHĂN

● 1

Bài báo không cung cấp bộ dữ liệu  
cũng như là các mã code liên quan.

● 3

Thu thập bộ dữ liệu: máy chủ nơi lưu trữ  
đôi lúc bị sập, tài nguyên trên đây có  
dung lượng khá lớn. (Lỗi 403 Forbidden)

● 2

Phương pháp học tăng cường để lựa  
chọn đặc trưng còn quá mới, nhiều kỹ  
thuật.

# CẢI TIẾN

- Thu thập bộ dữ liệu nhiều mẫu hơn nữa để đánh giá.
- Sử dụng các phương pháp lựa chọn đặc trưng tối ưu hơn để cải thiện hiệu suất mô hình cũng như là giảm chi phí tính toán và thời gian hơn.
- Sử dụng mô hình deep learning thay vì traditional learning.

# DEMO

29

File Edit View Run Kernel Tabs Settings Help

110%

jupyter-iec-hahien@bmft-se X extract\_feature\_malware TMX test.ipynb droidlysis3.py

Notebook

Filter files by name

/ MaDoc / test /

Name	Last Modified
benign	2 minutes ago
benign.json	4 minutes ago
malware	23 hours ago
malware.json	4 minutes ago
output	2 minutes ago

```
[1]: !python3 droidlysis/droidlysis3.py --input test/benign --output ./test/output --config droidlysis/conf/general.conf

[2]: !python3 droidlysis/droidlysis3.py --input test/malware --output ./test/output --config droidlysis/conf/general.conf

[3]: from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_auc_score, roc_curve, precision_score, recall_score, f1_score, accuracy_score
from fastcore.basics import *
from fastcore.parallel import *
from os import cpu_count

[4]: def get_static_feature(data):
    values = dict()
    for x in data.keys():
        if x == "sanitized_basename":
            continue
        if isinstance(data[x], dict):
            for key in data[x].keys():
                if isinstance(data[x][key], list):
                    print(x, key)
                    if key == "permissions":
                        for perm in data[x][key]:
                            values[f'permissions_{perm.split(".")[-1]}'] = 1

                    values[f'{x}_{key}_len'] = len(data[x][key])
                    continue
                if key in ['main_activity', 'package_name', 'app_name']:
                    continue
                try:
                    values[f'{x}_{key}'] = int(data[x][key])
                except:
                    values[f'{x}_{key}'] = -1
            else:
                values[x] = int(data[x])
    return values
```

Would you like to receive official Jupyter news?  
Please read the privacy policy.

Open privacy policy Yes No

**University of Information Technology (UIT) - VNUHCM | 2024**

**THANK YOU**

**GROUP 01**