ICANN21
30th International Conference on Artificial Neural Networks

Falcon: Malware Detection and Categorization with Network Traffific Images

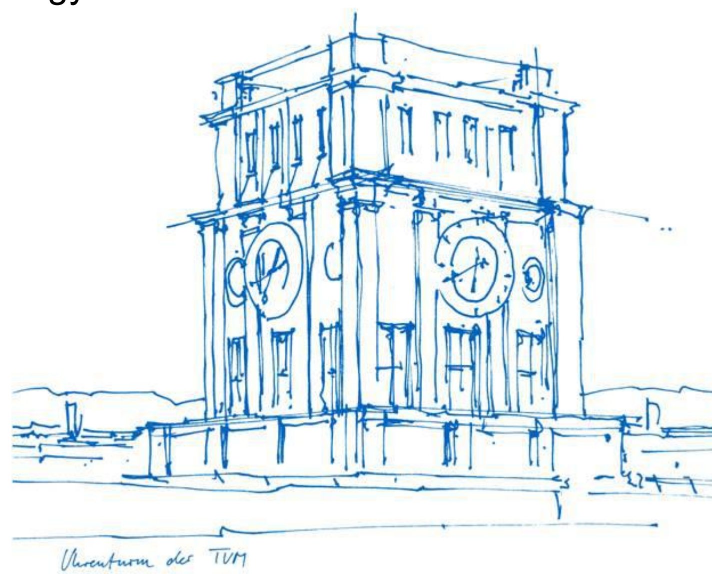
Peng Xu[1], Claudia Eckert[1], Apostolis Zarras[2]

{Peng,eckert}sec.in.tum.de

a.zarras@tudelft.nl

[1] Technical University of Munich

[2] Delft University of Technology



Agenda

- Introduction
- Related Work
- System Design and Implementation
- Evaluation
- Limitation
- Conclusion

Introduction

Problems

- ✓ Android is the **main target** for many attackers who seek to exploit **new victims**
  - ➢ Sending spam emails, spreading new malware, generating revenue from online advertisements
- ✓ Continuous investigation on the Android malware detection and categorization
  - ➢ **Static feature**-based method: API-call-based, Permission-based, executable-pattern-based, etc.
  - ➢ **Dynamic feature**-based method: network-traffic-based, system-call tracing-based, etc.

Introduction

Motivation

- ✓ **Static** analysis
  - ➢ **Hidding** vulnerabilities and malware invoked by 3rd-party libraries at runtime
- ✓ **Dynmiac** analysis
  - ➢ Port-, IP-address-based methods are too simple(**Manual features**) for the sophisticated malware
  - ➢ System-call-based methods are too **expensive** and **inefficient**
- ✓ **Falcon**: Efficient, **dynamic analysis**, and **representation-learning** based method

Introduction

Contribution

- ✓ present ***Falcon***, a **network-traffic**-pattern-based Android malware detection and categorization framework;
- ✓ design a **bidirectional LSTM** network to accomplish 2D gray **image sequence classification**, which takes the network packets as input.
- ✓ create a dataset, ***AndroNetMnist***, which includes 3,255,391 2D gray images in five classes for network traffic classification.
- ✓ evaluate the **accuracy** of our approach using real-world datasets

Related Work

✓ Feature-code-based methods

Related Work

✓ Feature-code-based methods

✓ Machine/Deep learning-based methods

- ➢ **Malicious API**(Mainfest file) based methods

Related Work

- ✓ Feature-code-based methods
- ✓ Machine/Deep learning-based methods
  - ➢ **Malicious API**(Mainfest file) based methods
  - ➢ **Permission**-based methods

Related Work

✓ Feature-code-based methods

✓ Machine/Deep learning-based methods

- **Malicious API**(Mainfest file) based methods
- **Permission**-based methods
- **Program-code-based methods**
  - **Control flow** graph-based
  - **Function-call**/**API-call** graph-based
  - Executable-file pattern-based

Related Work

✓ Feature-code-based methods
✓ Machine/Deep learning-based methods
- **Malicious API**(Mainfest file) based methods
- **Permission**-based methods
- Program-code-based methods
- **Network-traffic**-based-methods

Design and Implementation

✓ Overview

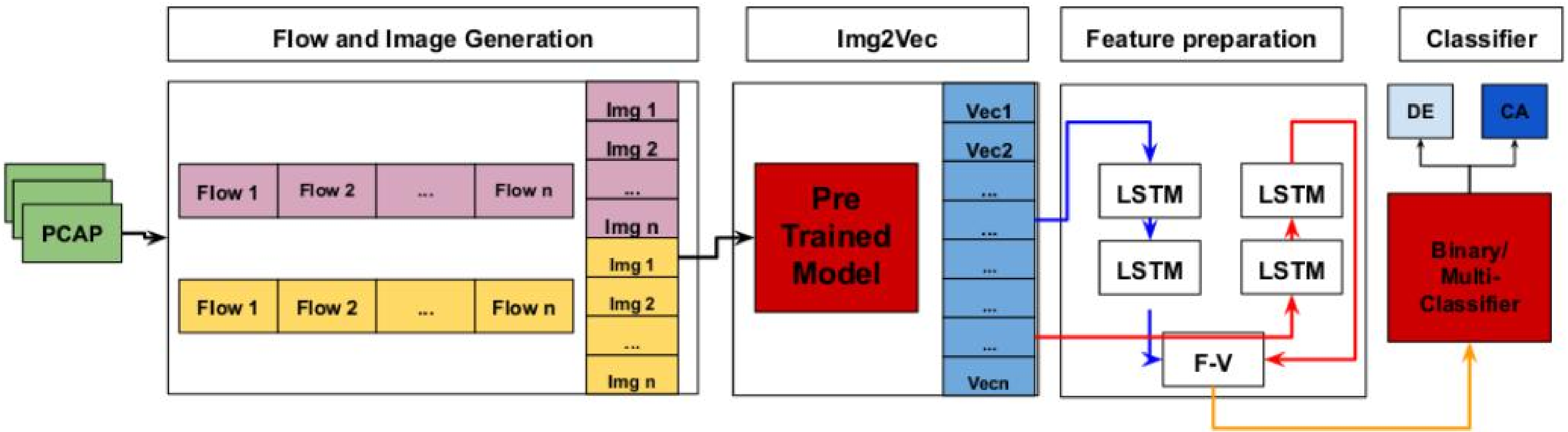


Fig. 1: The architecture of *Falcon*

feature vectors (F-V block in Figure 1)

a classifier to detect (DE block)
categorize(CA block)

Design and Implementation

✓ Overview

✓ **Feature Extraction**

- Network Traffic and Flow
  - Network Traffic Analysis
    - Package, Flow, Session
  - Network Flow: 5-tuple

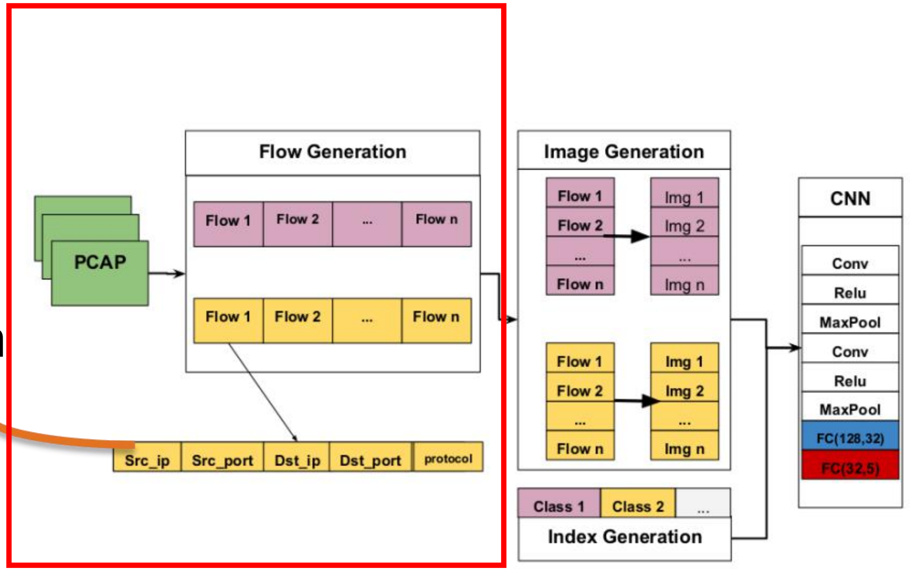


Fig. 2: Converting network traffic to vectors

Design and Implementation

✓ Overview

✓ **Feature Extraction**

➢ Network Traffic and Flow

➢ Network Flows to Images

- Images:
  - **784**-Byte
  - Trimming and Padding
- Labels:
  - Five classes: one **benign** and four **malware**

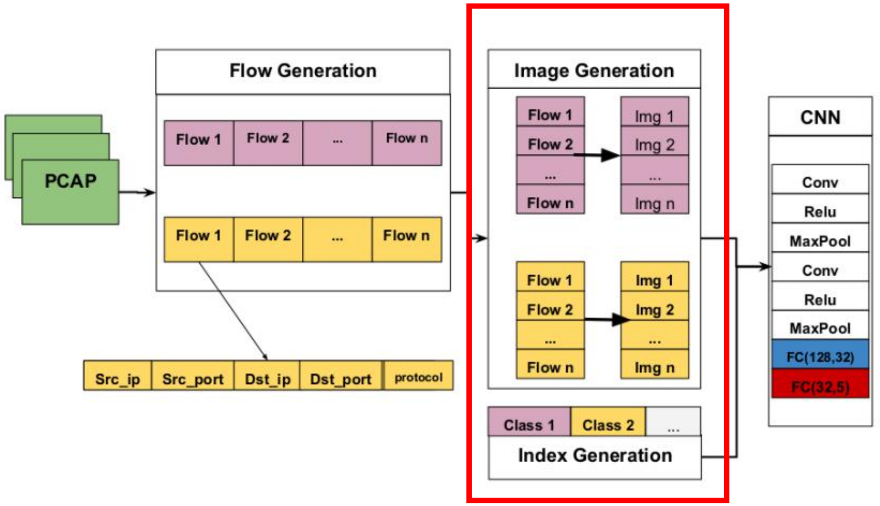


Fig. 2: Converting network traffic to vectors

Design and Implementation

✓ Overview
✓ Feature Extraction
➢ Network Traffic and Flow
➢ Network Flows to Images
➢ Feature Generation

- an **8-layer** convolution neural network
- **70,213** total parameters

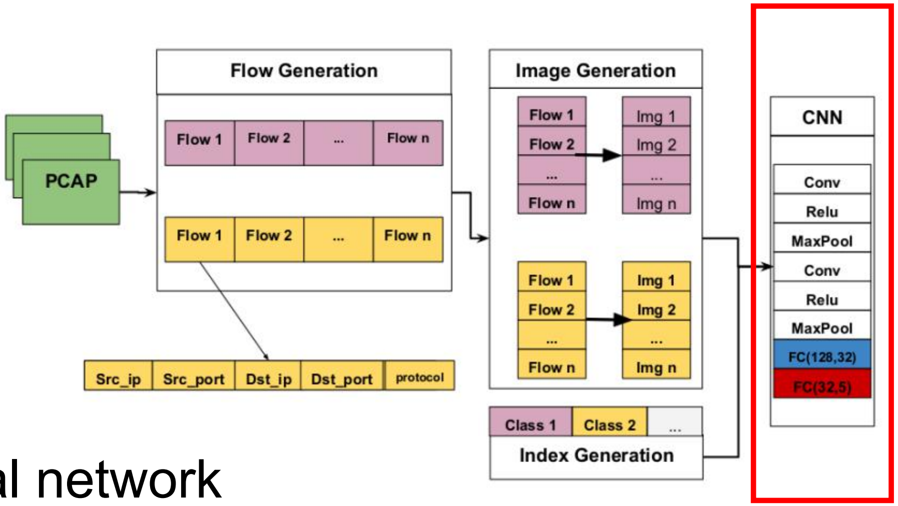


Fig. 2: Converting network traffic to vectors

$$
\begin{aligned}
Y^1 &= MaxPooling_{2*2}(Relu(conv2d_{3*3}(X_{28*28}))) \\
Y^2 &= MaxPooling_{2*2}(Relu(conv2d_{3*3}(Y^1))) \\
Y^3 &= FC_{128,32}(Y^2) \\
Y &= FC_{32,5}(Y^3)
\end{aligned}
$$

Design and Implementation

✓ Overview

✓ Feature Extraction

- ➢ Network Traffic and Flow
- ➢ Network Flows to Images
- ➢ Feature Generation
- ➢ Continous Network Traffic
  - a **2D sequential image** classification
  - a **bi-directional LSTM** network
  - **network traffic's continuous** characteristics

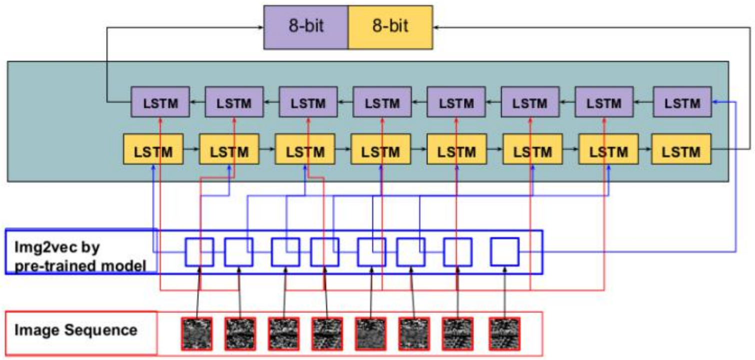


Fig. 3: 2D sequential image classification with bidirectional LSTM

Design and Implementation

✓ Overview

✓ Feature Extraction

✓ Model Training and Prediction

$$Loss = -\sum_{N}^{i=1} y_{i_{label}} * log(y_{i_{pred}})$$

$$= -\sum_{N}^{i=1} y_{i_{label}} * log(<(<f_v, w_{i1}> + b_{i1}), w_{i2}> + b_{i2})$$

$sparse\_categorical\_crossentropy$ loss function

$w_{i1}, w_{i2} \in R^p$ is the weight of the classifier

$b_{i1}, b_{i2} \in R^p$ is the offset from the origin of the vector space

Evaluation

- ✓ Experimental Setup
  - ✓ Platform
    - ➢ Linux X86-64
    - ➢ 128 GB RAM and 16 GB GPU
  - ✓ Software
    - ➢ Tensorflow 2.0.0-beta0
    - ➢ Keras 2.2.4
    - ➢ Sklearn 0.20.0
    - ➢ SplitCap
    - ➢ pillow 6.1.0
    - ➢ numpy 1.16.4
    - ➢ matplotlib 3.1.1

Evaluation

✓ Experimental Setup

✓ Datasets

Table 1: Dataset explanation

Name	Description	Number
PCAP files	*All the raw network traffic files*	2,126
Network flows	*All network flows in Section 3.2*	3,255,391
Adware	*Adware network flows partition*	580,170
Ransomware	*Ransomware network flows*	382,279
Scareware	*Scareware network flows*	517,954
SMSmalware	*SMSmalware network flows*	245,691
Benign	*Network flows for benign applications*	1,529,297

- 426 malware and 1700 benign samples
- Split the dataset with 80% training and 20% testing

Evaluation

✓ Experimental Setup

✓ Datasets

✓ Results Comparison - **Malware Detection**

Table 2: Malware detection comparison

Classifier	**Accuracy**	**Precision**	**Recall**	**F1**
Drebin [3]	96.58	95.37	97.85	96.59
Adagio [10]	89.32	91.27	95.28	93.23
Droidmat [30]	89.87	90.89	88.28	89.56
CICAndMal2017 [13]	87.52	87.14	87.73	87.18
Falcon-CNN	98.04	98.09	98.05	98.06
Falcon	97.16	97.13	97.16	97.09

RF *n_estimators=1400, min_sample_split=5, max_features="sqrt", max_depth=80*

Evaluation

✓ Experimental Setup

✓ Datasets

✓ Results Comparison - **Malware Detection**

Table 2: Malware detection comparison

Classifier	Accuracy	Precision	Recall	F1
Drebin [3]	96.58	95.37	97.85	96.59
Adagio [10]	89.32	91.27	95.28	93.23
Droidmat [30]	89.87	90.89	88.28	89.56
CICAndMal2017 [13]	87.52	87.14	87.73	87.18
Falcon-CNN	98.04	98.09	98.05	98.06
Falcon	97.16	97.13	97.16	97.09

Falcon-CNN cannot determine the **whole network flows characteristics** because most malicious behaviors are hidden in a **few network flows** by sophisticated attackers.

Evaluation

✓ Experimental Setup

✓ Datasets

✓ Results Comparison - **Malware Categorization**

Table 3: Malware categorization comparison (the average is weighted)

Classifier	**Accuracy**	**Precision**	**Recall**	**F1**
CICAndMal2017 [13]	86.85	85.92	86.85	84.82
Falcon-CNN	97.23	97.28	97.23	97.24
Falcon	84.70	80.22	84.70	82.39

RF *n_estimators=1400, min_sample_split=5, max_features="sqrt", max_depth=80*

Evaluation

✓ Experimental Setup

✓ Datasets

Table 4: Various classifiers settings

Classifier	Settings
RF	*n_estimators=1400, min_sample_split=5, max_features="sqrt", max_depth=80*
AdaBoost	*All default values*
GradientBoost	*lr=0.01, n_estimators=1500, max_depth=4, min_samples_split=40, max_features=4*
MLP	*sover="sgd", alpha=1e-5, hidden_layers_sizes=(400,400,200,100,10)*
DecisionTree	*min_samples_split=10, max_features="sqrt", max_depth=20*

✓ Results Comparison - Various Classifiers

Table 5: *Falcon*'s performance with various classifiers

Classifier	Accuracy	Precision	Recall	F1
RF	97.16	97.13	97.16	97.09
AdaBoost	93.13	92.81	93.13	92.85
GradientBoost	96.88	96.83	96.88	96.80
MLP	91.01	90.48	91.01	90.02
DecisionTree	93.66	93.64	93.66	93.65

Limitation

✓ Dataset

- Dynamic Network Flow dataset: small number
- Samples classes: only **four** malware types and one benign

Limitation

✓ Dataset

✓ Time efficiency

- Time consumption: more time comsumption than port-based method

Conclusion

- ✓ present ***Falcon***, a **network-traffic**-pattern-based Android malware detection and categorization framework;
- ✓ design a **bidirectional LSTM** network to accomplish 2D gray **image sequence classification**, which takes the network packets as input.
- ✓ create a dataset, ***AndroNetMnist***, which includes 3,255,391 2D gray images in five classes for network traffic classification.
- ✓ evaluate the **accuracy** of our approach using real-world datasets

Thank you !!!
Questions?
Comments?