Understanding and Mitigating Model Aging of ML-based Android Malware Detectors

张晓寒，博士后
xh_zhang AT fudan.edu.cn
复旦大学系统软件与安全实验室
杨珉教授团队

Enhancing State-of-the-art Classifiers with API Semantics to Detect Evolved Android Malware, CCS ’20

ML-based Android Malware Detectors

- ML/DL is now widely used in Android malware detection
  - > 90% papers use ML to detect malware, in top venues from 2013 to 2019

Android Malware Detector	Algorithm	Android Malware Detector	Algorithm
DroidAPIMiner-SecComm13	ID3, k-NN, C4.5, SVM	MamaDroid-NDSS17	RF, SVM, k-NN
DroidMiner-Esorics14	NB, SVM, DT, RF	DroidSieve-Codaspy17	RF, SVM
Drebin-NDSS14	SVM	Transcend-Security17	SVM
DroidSIFT-CCS14	×	PIKADroid-ACSAC18	K-NN, RF, **MLP**
MARVIN-Acsac15	LR	DeepRefiner-EuroSP18	**DNN**
AppContext-ICSE15	SVM	DroidEvolver-EuroSP19	5 linear algorithms
Afonso-JCVHT15	RF	TESSERACT-Security19	SVM, RF, **DNN**
TreeFall-NDSS16	SVM	EveDroid-IoTJ19	**DNN**
Stormdroid-AsiaCCS16	K-NN, C4.5	DroidSPAN-TOSEM19	RF
……		……	

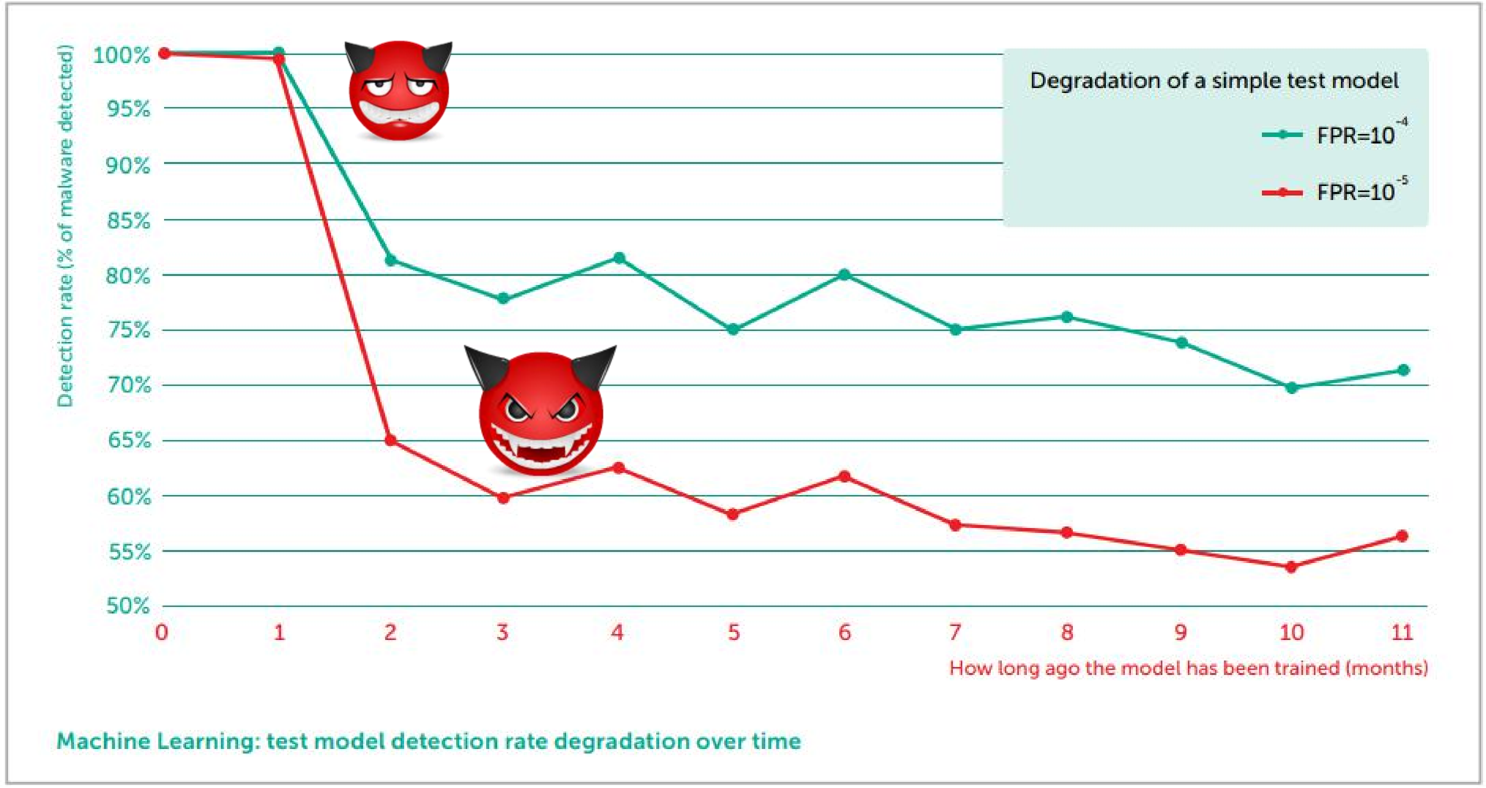
McAfee

avast

kaspersky

Problem: Model Aging of Malware Detectors

- model aging: performance of ML models drop drastically over time

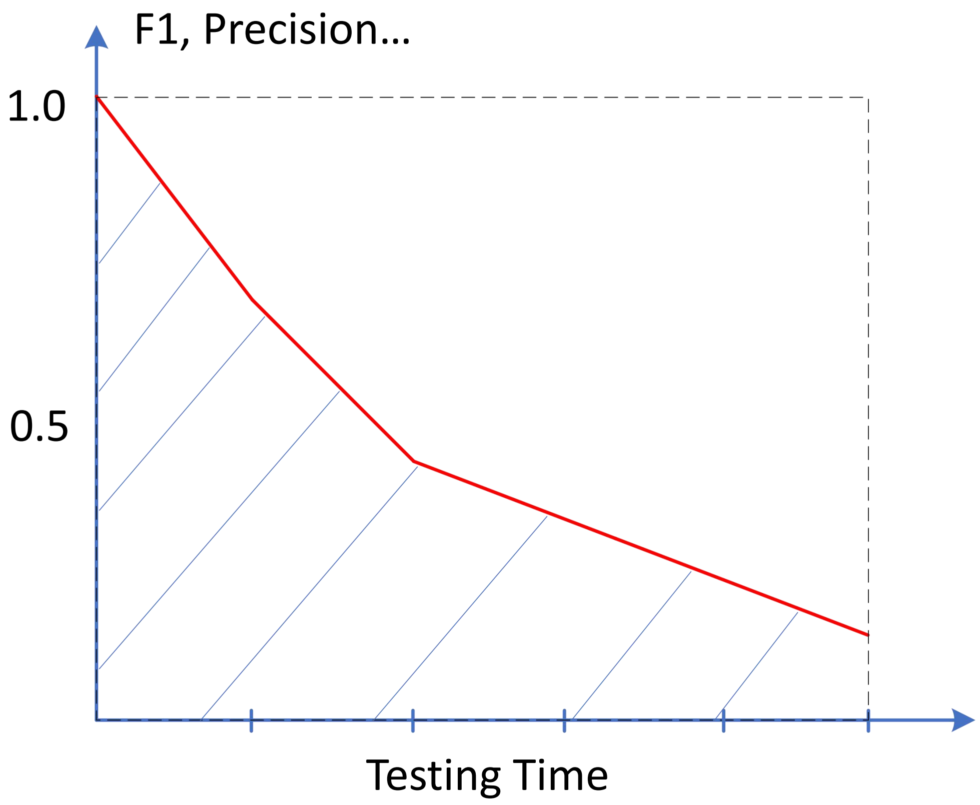


The detection rate of an ML-based detector from **Kaspersky** drops from **~100% to below 60% in 3 months**

https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf, Kaspersky whitepaper 2019

Measuring Model Aging

- The **AUT** metric: **A**rea **U**nder the performance curve over **T**ime



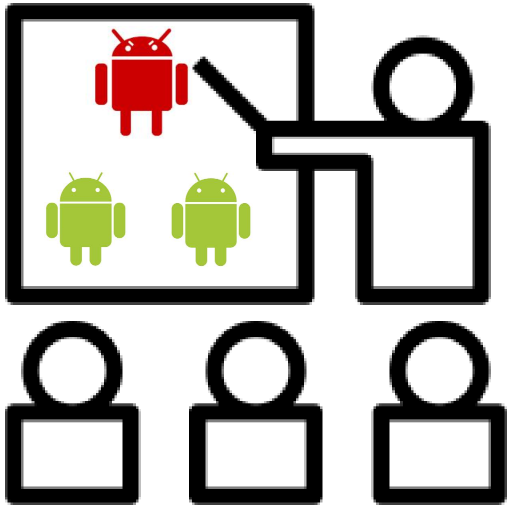
- Tesseract-Sec19 tests the performance of 3 SOTA malware detectors
  - they all age significantly

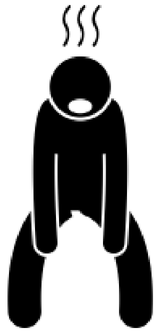
Malware Detectors	AUT(F1, 24m)
Drebin-NDSS2014	0.58
MamaDroid-NDSS2017	0.32
DL-Esorics2017	0.64

TESSERACT: Eliminating experimental bias in malware classification across space and time, Security ’19

Tackle Model Aging: Existing Methods

- **Retraining**: update the aged models with newly labeled samples
- **Optimizations**:
  - Online/incremental learning
    - [DroidOL-IJCNN2016, DroidEvolver-EuroSP19]
  - Active learning
    - [Tesseract-Security19]





1. high cost

labeling efforts

time window

2. still blind

malware evolution

Motivating Example

- XLoader is a family of spyware and banking trojan
  - reported by TrendMicro, April, 2018, has evolved into several variations
  - steals personally identifiable information (PII) and financial data

Key observation: semantics are preserved during evolution while implementation may be different

```
// collect personally identifiable information
JSONObject data;
data.put(getDeviceId());
...
// send collected data to server through HTTP
URL url = new URL(SERVER_ADDR);
HttpURLConnection conn = url.openConnection();
conn.connect();
out = new DataOutputStream(conn.getOutputStream());
out.writeBytes(data.toBytes());
...
```

IMEI

HTTP

Listing 1: pseudo-code of XLoader V1

```
// collect personally identifiable information
JSONObject data;
data.put(getDeviceId());
data.put(getSubscriberId());
data.put(getSimSerialNumber());
...
// send collected data to server through Socket
Socket socket =
      SocketFactory.createSocket(SERVER_ADDR);
out = new OutputStream(socket.getOutputStream());
out.writeBytes(data.toBytes());
 ...
```

IMEI, IMSI, ICCID

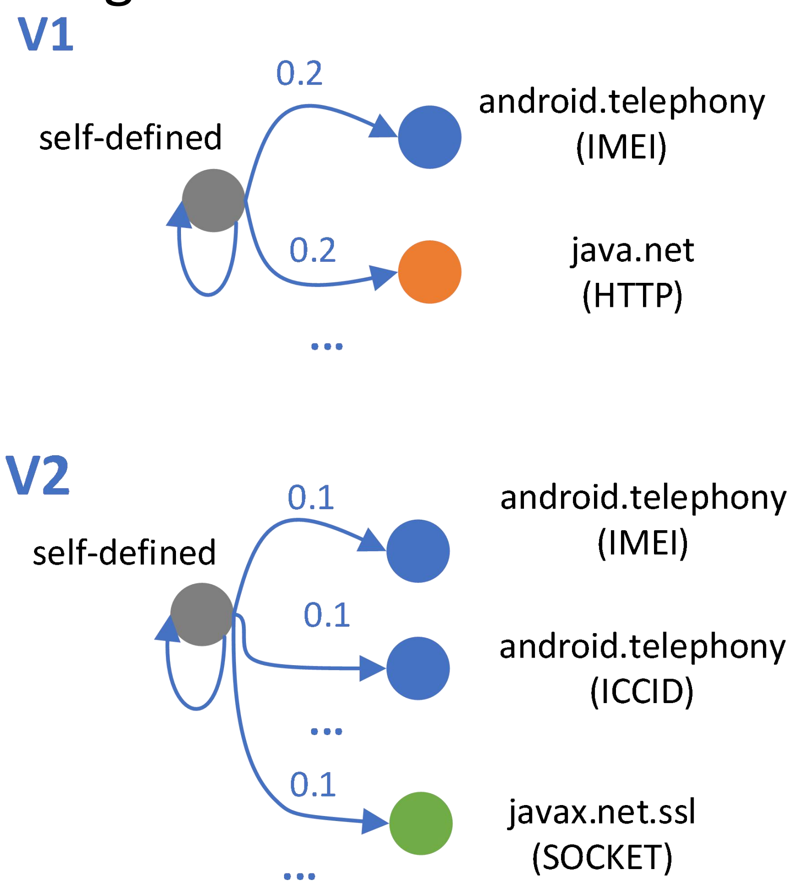
Socket

Listing 2: pseudo-code of XLoader V2

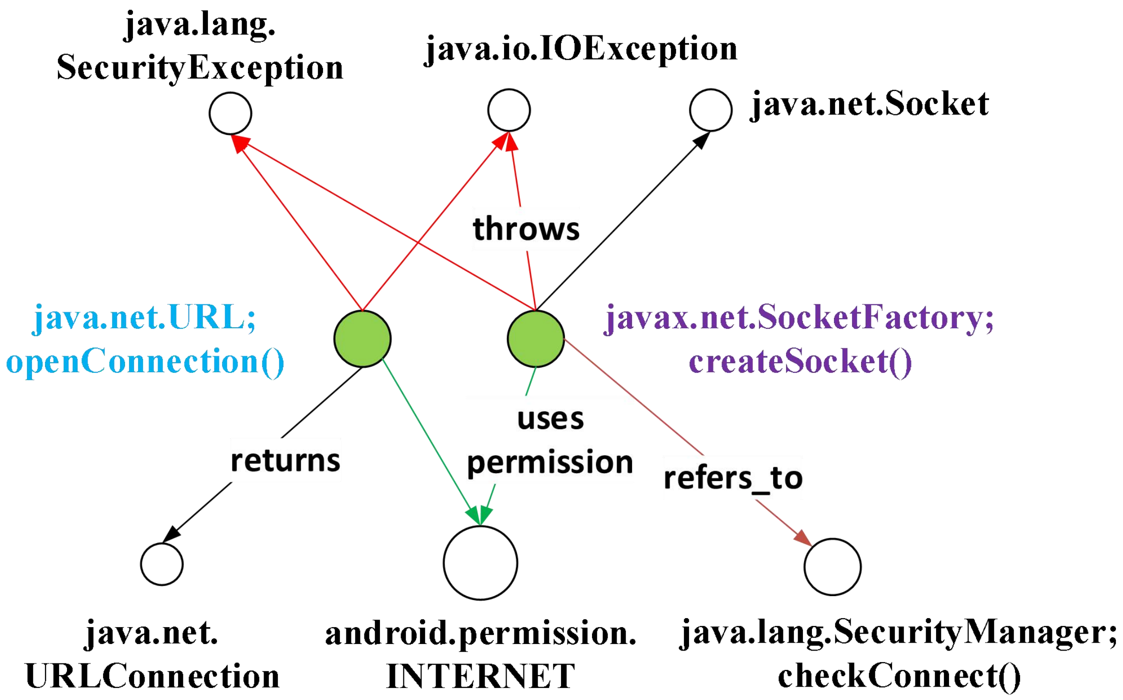
simplified code snippets from two versions V1 and V2

Key Idea: Leveraging API Semantics

- Models without API Semantics
  - e.g API transfer matrix

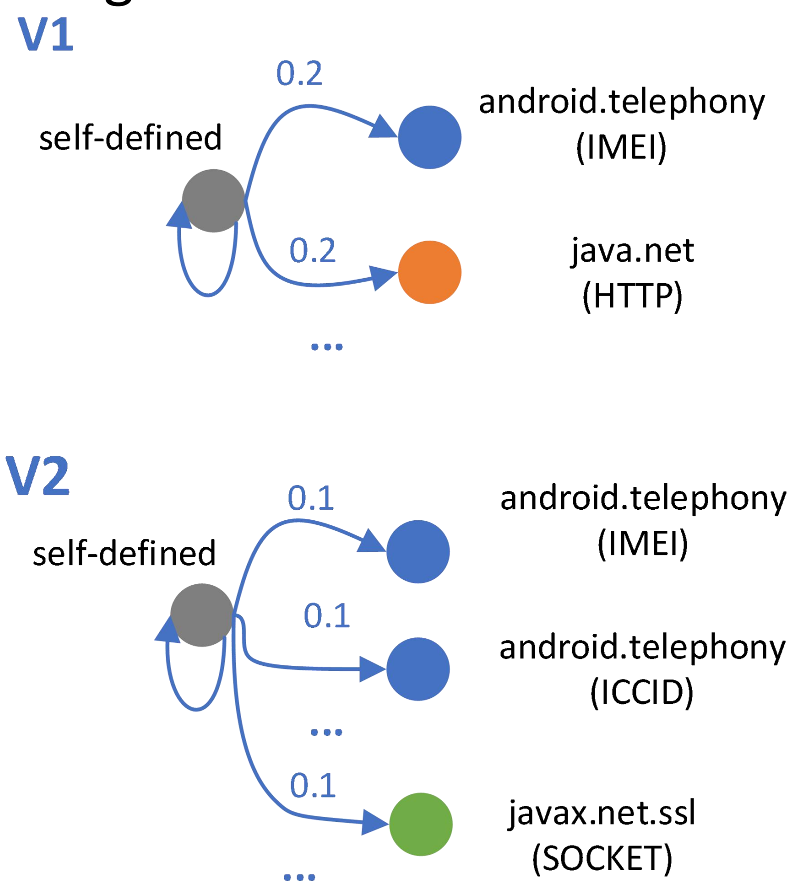


- Knowledge of API Relations

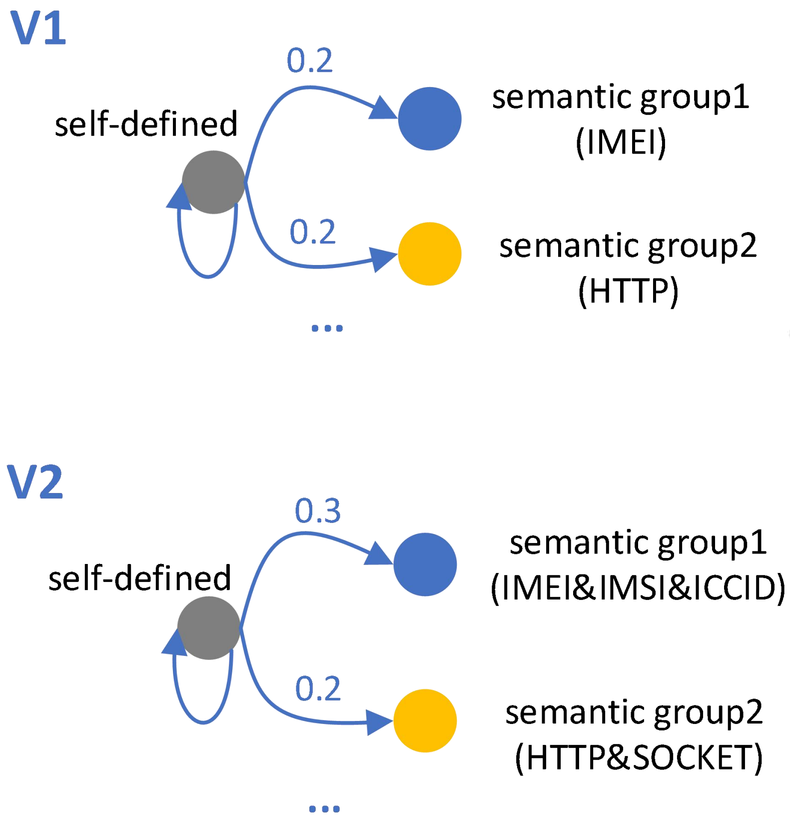


Key Idea: Leveraging API Semantics

- Models without API Semantics
  - e.g API transfer matrix



- Models with API Semantics

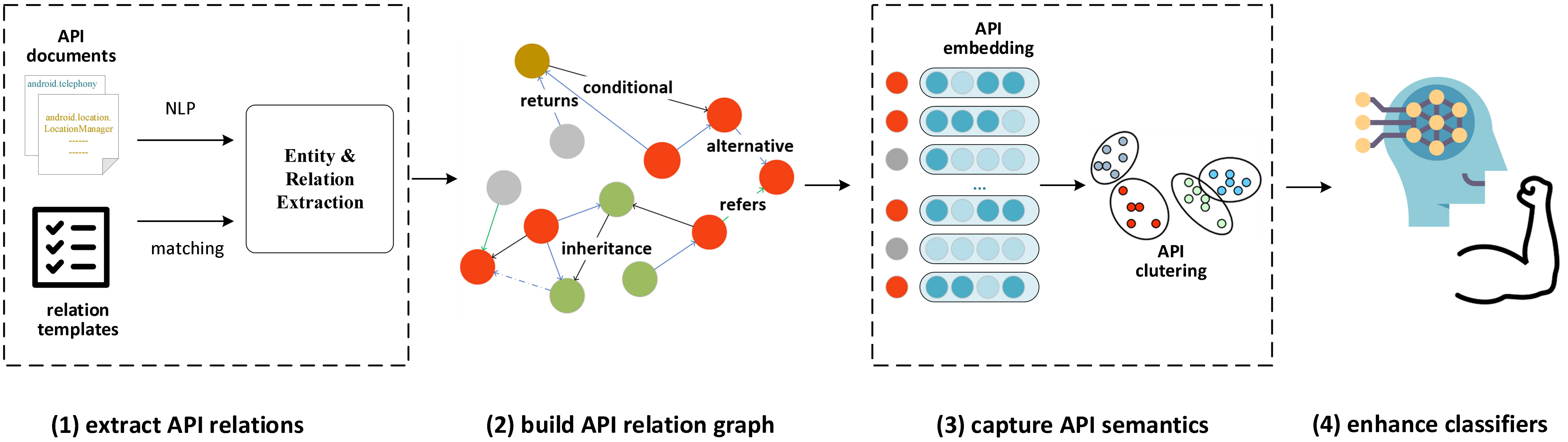


Detected!

APIGraph Overview

extract API semantics from API documents,

and use such knowledge to enhance existing malware detectors



API Relation Graph

- $G = < E, R >$
  - directed heterogeneous graph
  - 4 entity types: API, class, package, permission
  - 10 relation types selected from [1]:

Perspective	Relations	Entities	Examples
Organization	class_of	class→ package	*java.net.Socket* is class_of *java.net*
	function_of	method→ class	*BluetoothDevice.getAddress()* is function_of *android.bluetooth.BluetoothDevice*
	inheritance	class → class	*javax.net.ssl.SSLSocketFactory* inheritance *javax.net.SocketFactory*
Prototype	uses_parameter	method→class	*javax.net.SocketFactory.createSocket()* uses_parameter *java.net.INetAddress*
	returns	method→class	*java.net.Socket.getInputStream()* returns *java.io.InputStream*
	throws	method→class	*LocationManager.requestLocationUpdates()* throws *java.lang.SecurityException*
Usage	conditional	method→method	"This method should be called after ...", "... is called when ..."
	alternative	method→method	"This method is deprecated, use ... instead", "is replaced by ..."
Reference	refers_to	method→ method method→class	"Please refer to ...", "see also ..."
Permission	uses_permission	method→permission	"requires INTERNET permission"

[1] Patterns of Knowledge in API Reference Documentation. TSE 2013

Extracting API Relation

android.telephony.TelephonyManager.html

TelephonyManager Added in API level 1

public class TelephonyManager
extends Object

java.lang.Object
↳ android.telephony.TelephonyManager

structured info

Public methods

getDeviceId

Added in API level 1
Deprecated in API level 26

public String getDeviceId ()

This method was deprecated in API level 26.
Use getImei() which returns IMEI for GSM or getMeid() which returns MEID for CDMA.

Returns the unique device ID, for example, the IMEI for GSM and the MEID or ESN for CDMA phones. Return null if device ID is not available.

Requires Permission: READ_PRIVILEGED_PHONE_STATE, for the calling app to be the device or profile owner and have the READ_PHONE_STATE permission, or that the calling app has carrier privileges (see hasCarrierPrivileges()) on any active subscription.
...

unstructured info

Returns
String

structured info

- NLP Pre-processing
  - stemming
  - co-reference resolution
  - entity name normalization
- Relation Templates
  - 217 templates based on an iterative workflow
    - call x before y be call — conditional
    - refer to x — refers_to
    - require permission x — uses_permission
    - ...

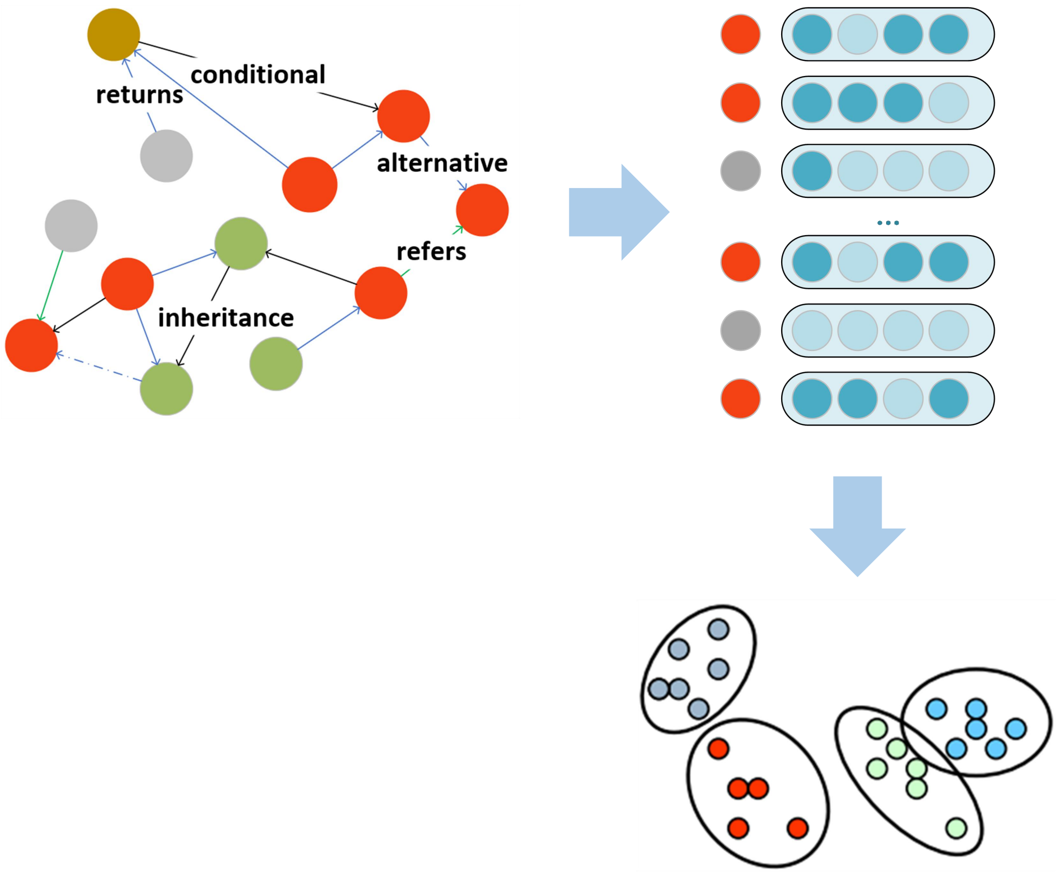
Entities and Relations

- 67,209 entities and 121,345 relations for Android API level 29

Entity Type	Count
method	59,125
class	7,368
package	446
permission	270

Relation Type	Count	Relation Type	Count
function_of	59,125	throws	8,310
class_of	7,368	alternative	1,264
inheritance	3,755	conditional	5,990
uses_parameter	14,528	refers_to	10,859
returns	5,113	uses_permission	5,033

Capturing API Semantics



- API Embedding
  - the **sum** of header entity vector and relation vector, as close as to the tail entity vector [1]

$$\ell = ||l_h + l_r - l_t||_2^2$$

- API Clustering
  - using k-Means to cluster semantically-similar APIs into the same group

[1] Translating Embeddings for Modeling Multi-relational Data, NeurIPS 2013

Large-scale Evolutionary Dataset

- Dataset Properties:
  - **large-scale**: **322,594** apps, including 290,505 benign and 32,089 malicious
  - **evolutionary**: **7** years from 2012 to 2018
  - **temporal** & **spatial consistency** following best practice by Tesseract-sec19:

temporal: training samples strictly temporally precedent to the testing ones

& malware and goodware from same time period during one test

spatial: malware ratio close to real-world, i.e. 10% for Android

Year / App	2012	2013	2014	2015	2016	2017	2018	ALL
Malicious (M)	3,066	4,871	5,871	5,797	5,651	2,620	4,213	32,089
Benign (B)	27,613	43,873	52,843	52,173	50,859	24,930	38,214	290,505
M+B	30,679	48,744	58,714	57,970	56,510	32,300	38,025	322,594
M/(M+B)	10%	10%	10%	10%	10%	10%	10%	10%

[1] TESSERACT: Eliminating experimental bias in malware classification across space and time, USENIX Security 2019

Evaluated Detectors

- Four state-of-the-art Android malware detectors
  - published on top-tier/well-known venues
  - different API feature format and learning algorithm
  - availability to reproduce

Classifier	Published	API feature format	Algorithm
MamaDroid	NDSS-2017	Markov Chain of API Calls	Random Forest
DroidEvolver	EuroSP-2019	API Occurrence	Model Pool of 5 linear online learning algorithms
Drebin	NDSS-2014	Selected API Occurrence	Support Vector Machine
Drebin-DL	ESORICS-2017	Selected API Occurrence	Deep Neural Network

Experiment 1: Slowing Down Model Aging

- Model aging metric: ***AUT****(F1, 12m)*

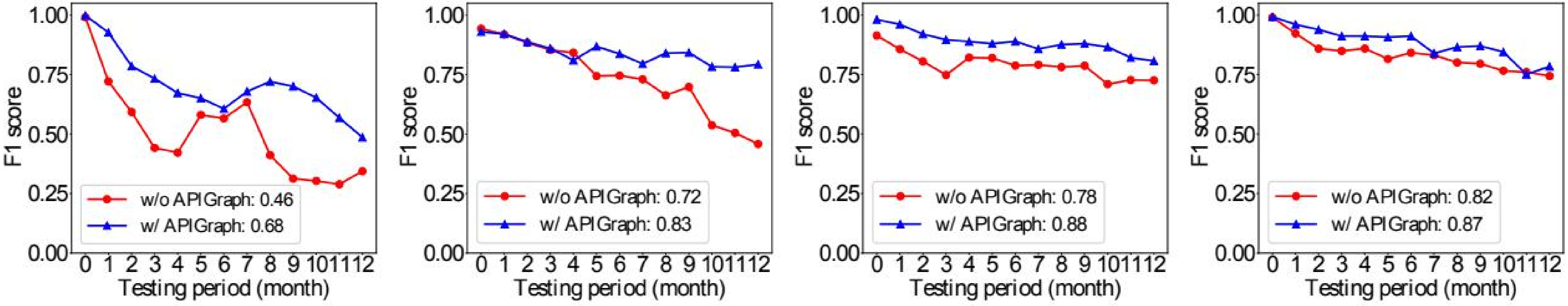
$$AUT(f,N) = \frac{1}{N-1}\sum_{k=1}^{N-1}\frac{[f(x_{k+1}) + f(x_k)]}{2}$$

Testing Years	MamaDroid		DroidEvolver		Drebin		Drebin-DL	
	w/o [1]	w/ [2]	w/o	w/	w/o	w/	w/o	w/
2013	0.462	0.680	0.717	0.833	0.779	0.878	0.819	0.875
2014	0.456	0.637	0.712	0.791	0.734	0.859	0.816	0.866
2015	0.726	0.789	0.840	0.890	0.759	0.886	0.829	0.878
2016	0.718	0.814	0.718	0.875	0.666	0.869	0.706	0.916
2017	0.635	0.704	0.605	0.908	0.767	0.844	0.793	0.797
2018	0.765	0.861	0.811	0.969	0.794	0.865	0.828	0.874
Average	0.627	0.748	0.734	0.877	0.750	0.867	0.799	0.868
Improves	19.2%		19.6%		15.6%		8.7%	

[1] w/o denotes the classifier without APIGraph, i.e. the original classifier.
[2] w/ denotes the classifier enhanced with APIGraph.

Experiment 1: Slowing Down Model Aging

Detailed result that trained on 2012 samples and testing on 12 months of 2013



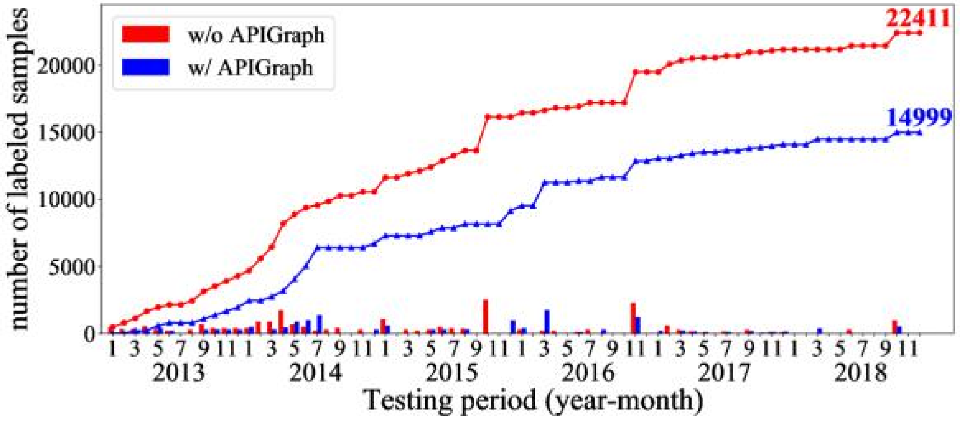
(a) MamaDroid (b) DroidEvolver (c) Drebin (d) Drebin-DL

Experiment 2: Reducing Retraining Cost

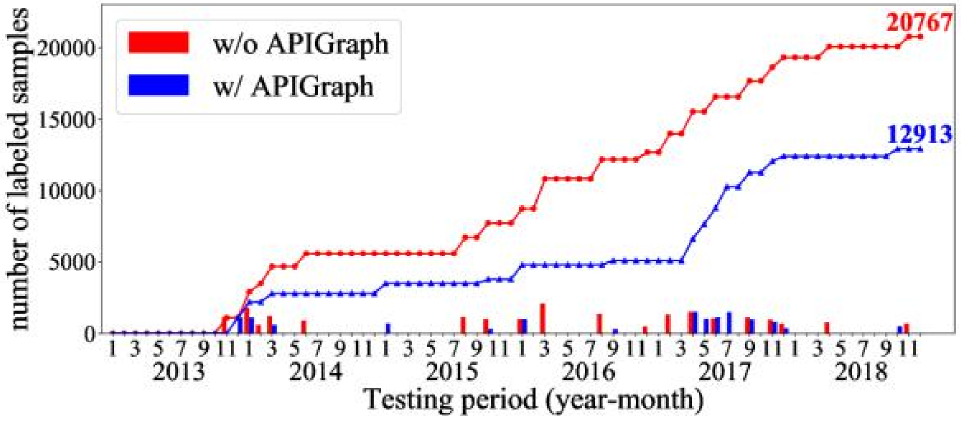
- Retraining cost metric:
  - retraining **frequency**
  - number of new **samples to label**
- Experiment settings:
  - Train a detector on 2012 samples and test from Jan 2013 to Dec 2018
  - When f1 below T_l(e.g. 0.8), retrain with active learning until f1 reaches T_h (e.g. 0.9)

	retrain frequency (months/retrain)			# labeled samples		
	w/o APIGraph	w/ APIGraph	Improves	w/o APIGraph	w/ APIGraph	Improves
MAMADROID	1.6	2.1	22.22%	22,411	14,999	33.07%
DROIDEVOLVER	3.8	4.8	21.05%	20,767	12,913	37.82%
DREBIN	1.3	5.5	76.79%	167,005	6,173	96.30%
DREBIN-DL	2.8	4.5	38.46%	28,408	9,292	67.29%

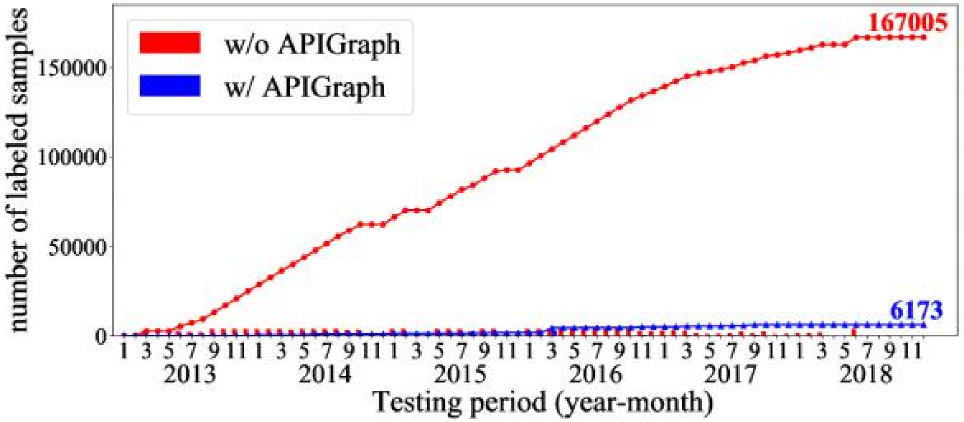
Experiment 2: Reducing Retraining Cost



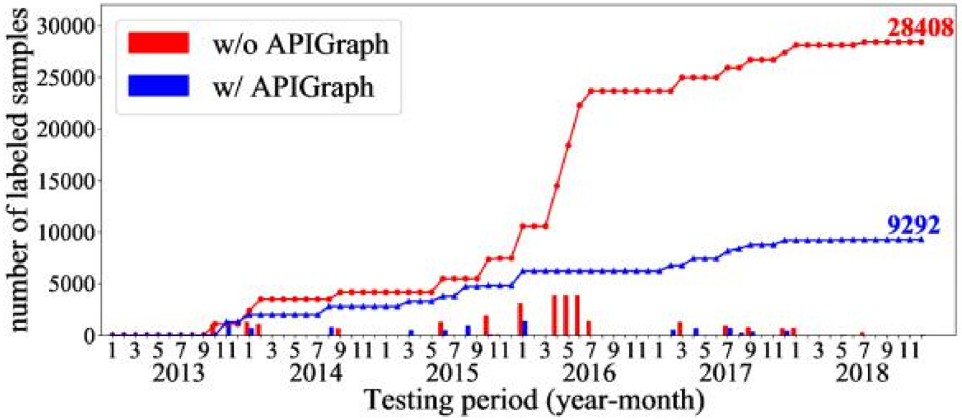
(a) The efforts in sample labeling for MAMADROID



(b) The efforts in sample labeling for DROIDEVOLVER



(c) The efforts in sample labeling for DREBIN



(d) The efforts in sample labeling for DREBIN-DL

Please refer to our paper for more experiment results

Conclusion

- Observe that many behavior semantics are still preserved during Android malware evolution, with different implementation

- Propose APIGraph to extract **API semantics** from API documentation and enhance existing detectors with such semantics

- Evaluate 4 SOTA Android malware detectors with APIGraph on a large-scale dataset spanning 7 years, and demonstrate promising results

- Release Code and Dataset
  - https://github.com/seclab-fudan/APIGraph

Rethinking

- Data-perspective VS. domain perspective
  - Model aging is complex, it needs study from both perspective
  - “Incorperating domain knowledge into models” -- CCS 2020 keynote
  - The idea of APIGraph may also be applied to other tasks

- Standard dataset/evaluation is yet to be complete in security tasks
  - Dataset: MNIST, CIFAR, etc
  - How to do fair experiments and compare different works?

复旦大学系统软件与安全实验室

杨珉 教授、博导

国务院网络空间安全学科评议组成员
教育部长江学者特聘教授
计算机科学技术学院，科研副院长
中国网络空间战略研究所，副所长
上海青年联合会，副主席
973项目首席科学家

研究方向

- 恶意代码检测
- 人工智能安全
- 漏洞分析挖掘
- 隐私数据保护

团队成员

- 杨哲慜、张源、张谧、张磊、张晓寒
- 博士生16人、硕士生60+人

学术成果

- 截至2021年，第一单位发表网络安全四大顶会论文 17 篇
- 2013年，ACM CCS发表 2 篇
- 2020年，四大顶会发表 7 篇研究论文

CTF战队

- 白泽安全攻防战队
  - 获国内外高水平安全竞赛 16 项一等奖
    - 国际顶尖攻防赛DEFCON总冠军（腾讯、复旦、上交、浙大联队，2020）
    - 第十三届全国大学生信息安全竞赛创新实践能力赛 特等奖（2020）
    - 全国高校网安联赛团队赛 特等奖（2020）
    - 全国工业互联网安全技术技能大赛 一等奖（2020）
    - WCTF世界黑客大师赛 冠军（2019）
    - 全国高校网安联赛团队赛 特等奖（2019）
    - 全国高校网安联赛个人赛 冠军（2019）
    - 全国大学生网络安全邀请赛 一等奖（2019）
    - 腾讯信息安全争霸赛TCTF新星赛 冠军（2019）
    - 全国大学生网络安全邀请赛 一等奖（2018）
    - 鹏城杯网络安全竞赛 一等奖（2018）

复旦白泽战队










白泽战队：我们的安全攻防战队

欢迎各位同学联系 & 申请2021复旦大学夏令营！

因兴趣而相遇

因相遇而幸运

用知识武装头脑，用零食武装胃

Q&A

THANK YOU!

Contact me about this talk or joining our lab:

xh_zhang AT fudan.edu.cn