Quality Assurance and Security in the Age of Machine Learning

CS 4301

Wei Yang

Department of Computer Science
University of Texas at Dallas

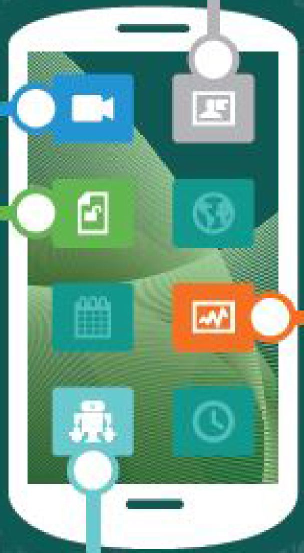
Outline

- Application of learning-based techniques
- Quality assurance of learning-based techniques
- Security of learning-based techniques
- Interpretation of learning-based security techniques

Outline

- Application of learning-based techniques
  - Mobile Security
  - Testing
  - Other
- Quality assurance of learning-based techniques
- Security of learning-based techniques
- Interpretation of learning-based security techniques

Mobile Security



Surveillance
- Audio
- Camera
- Call logs
- Location
- SMS messages

Impersonation
- SMS redirection
- Sending email messages
- Posting to social media

Data theft
- Account details
- Contacts
- Call logs
- Phone number
- Stealing data via app vulnerabilities
- Stealing international mobile equipment identity number (IMEI)

Financial
- Sending premium rate SMS messages
- Stealing transaction authentication numbers (TANs)
- Extortion via ransomware
- Fake antivirus
- Making expensive calls

Botnet activity
- Launching DDoS attacks
- Click fraud
- Sending premium rate SMS messages

The number of mobile malware keeps increasing

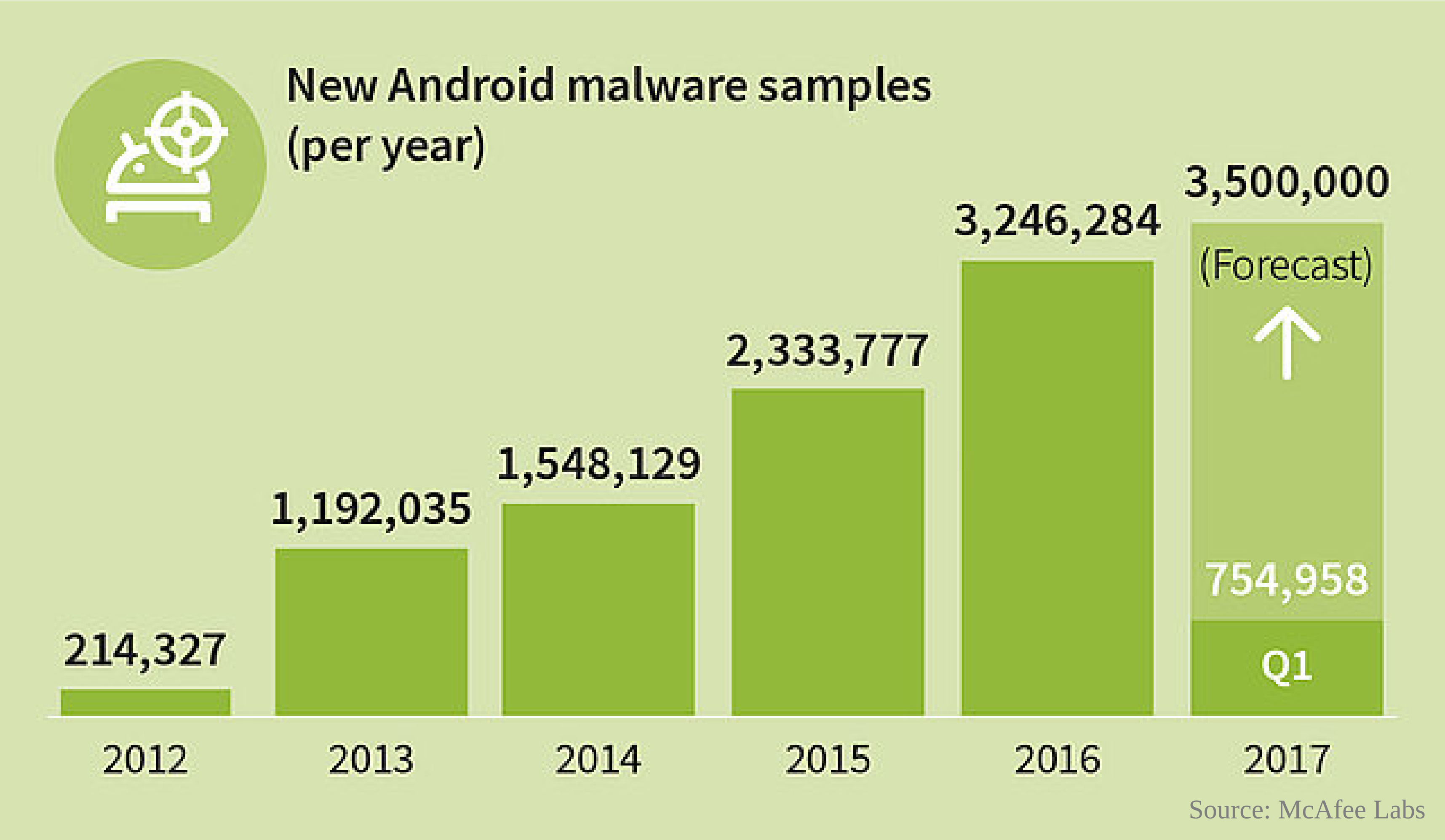
Report: I … n 2017, up

Andro … ctim to

drive- …

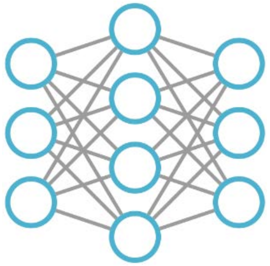
In jus … devices

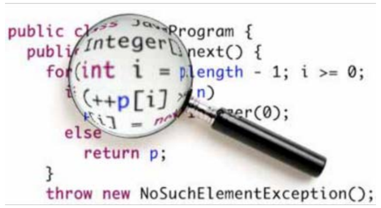
are c … net



Opportunities and challenges in mobile analysis

- Automated techniques have been introduced

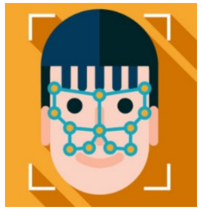




Machine Learning

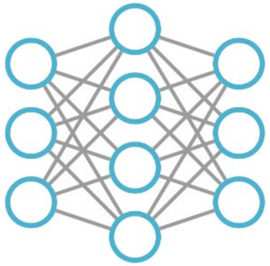
Natural Language Processing

Program Analysis



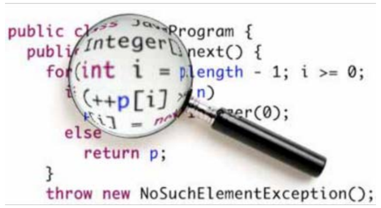
Opportunities and challenges in mobile analysis

- Automated techniques have been introduced



Machine Learning

Natural Language Processing

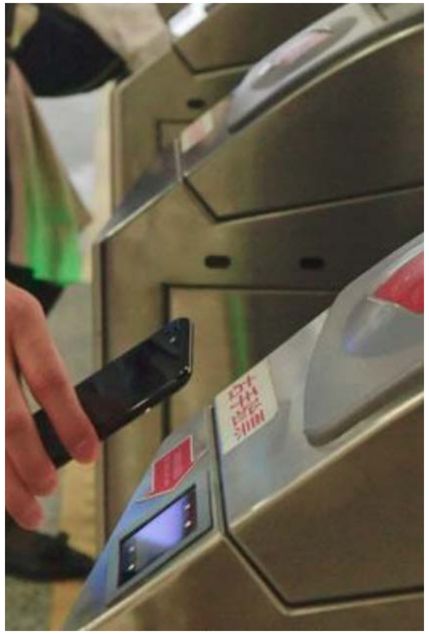


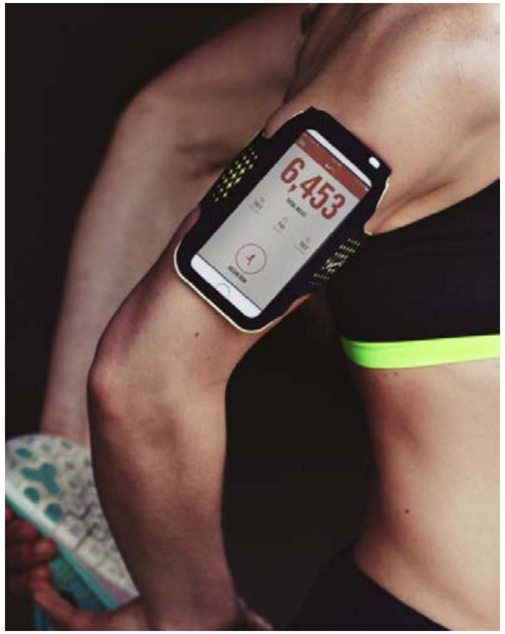
Program Analysis

How can we adapt these techniques to keep up their effectiveness in adversarial settings?

Opportunities and challenges in mobile analysis

- Automated techniques have been introduced
- Characteristics of mobile platform
  - Devices frequently interact with environment and other devices





Opportunities and challenges in mobile analysis

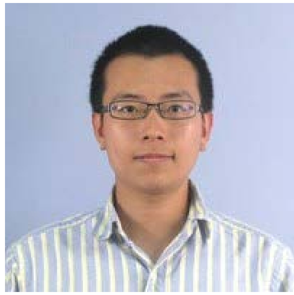
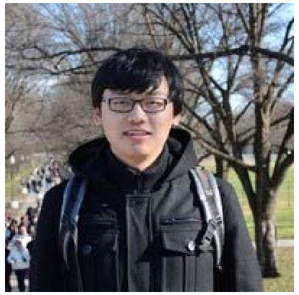
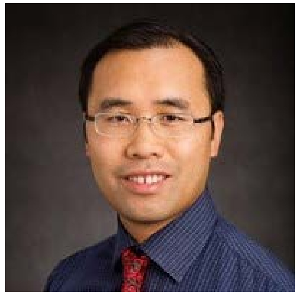
- Intelligent techniques have been introduced
- Characteristics of mobile platform
    - Devices frequently interact with environment and other devices
    - The mobile app ecosystem changes the way users install and use apps

Google play

Windows Phone

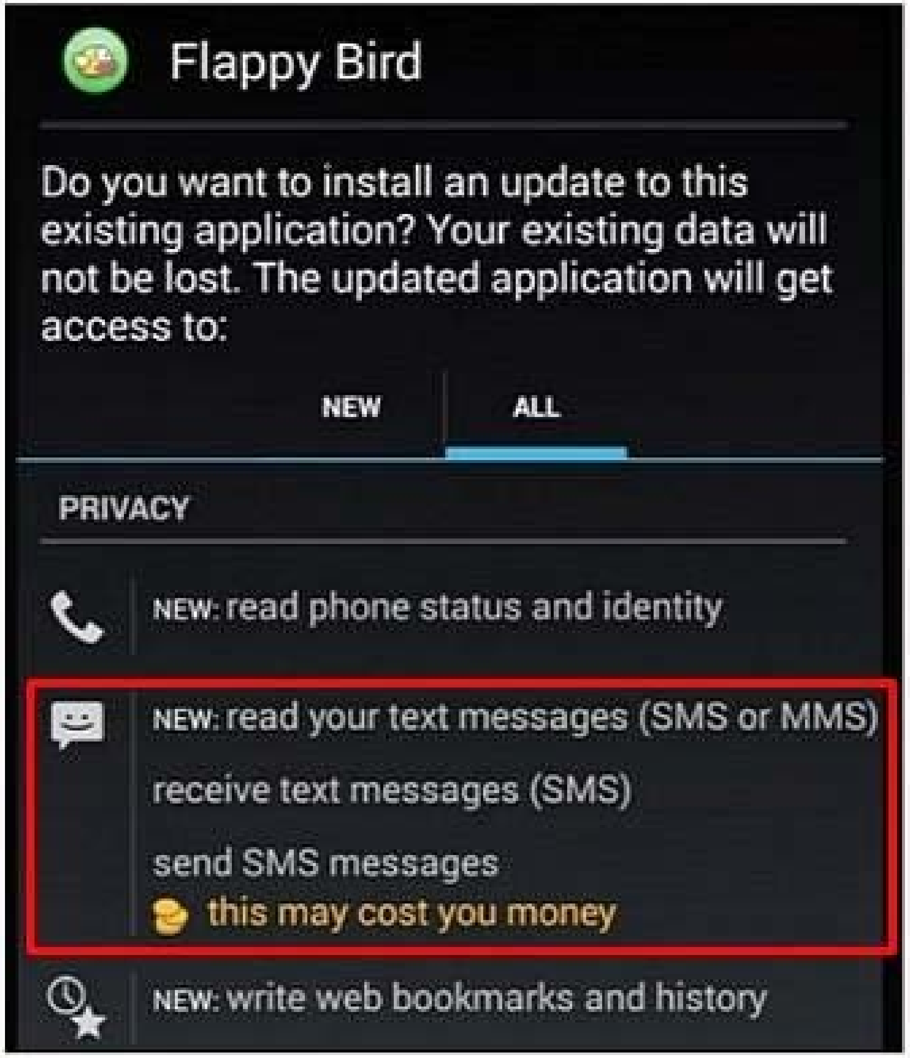
Malware Detection

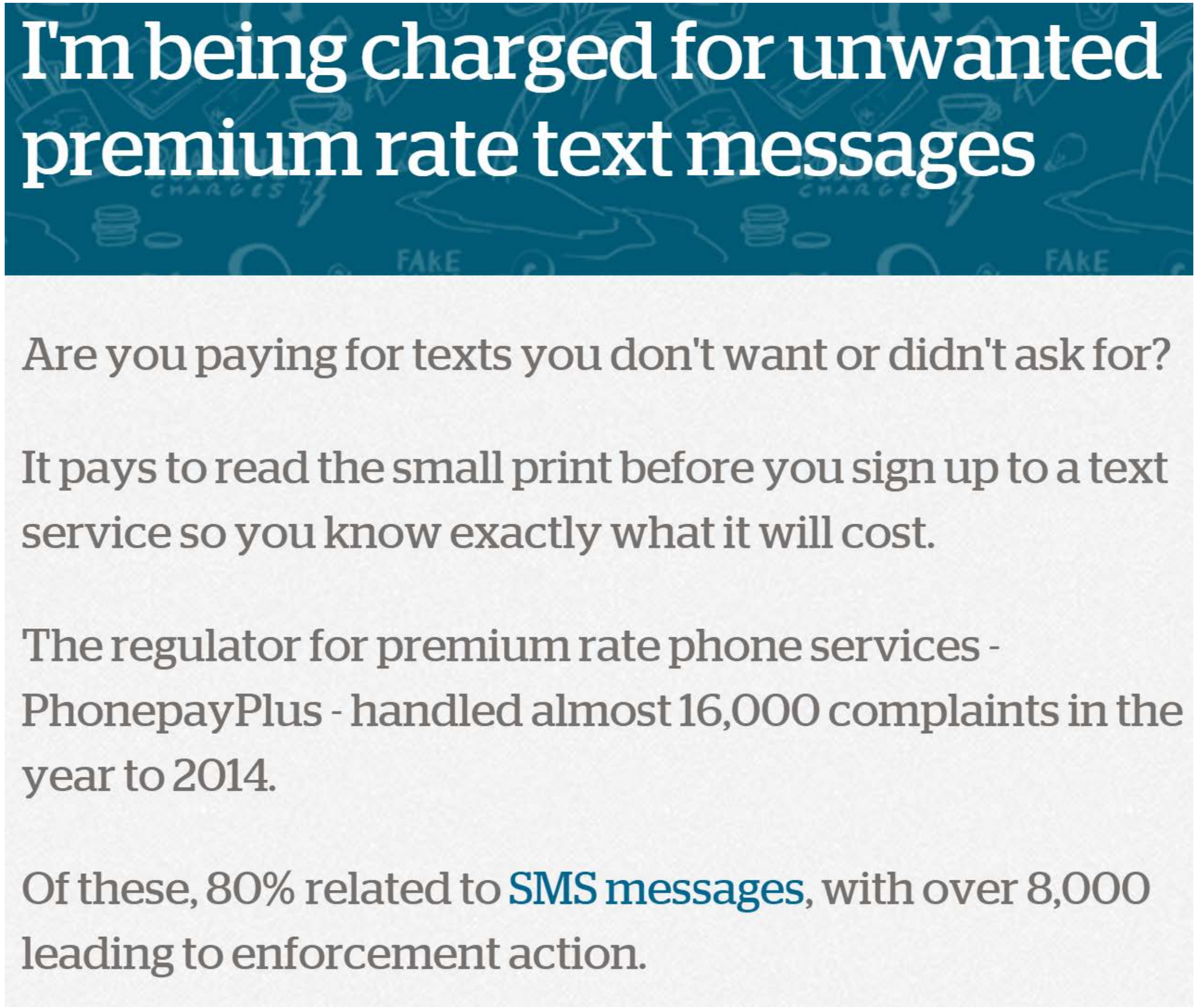
AppContext: Differentiating Malicious and Benign Mobile App Behavior using Contexts

ICSE

Example - A malicious app





I'm being charged for unwanted premium rate text messages

Are you paying for texts you don't want or didn't ask for?

It pays to read the small print before you sign up to a text service so you know exactly what it will cost.

The regulator for premium rate phone services - PhonepayPlus - handled almost 16,000 complaints in the year to 2014.

Of these, 80% related to SMS messages, with over 8,000 leading to enforcement action.

http://www.which.co.uk/consumer-rights/problem/im-being-charged-for-unwanted-premium-rate-text-messages

Checking security-sensitive behaviors

Permission List

Do you want to install this application?

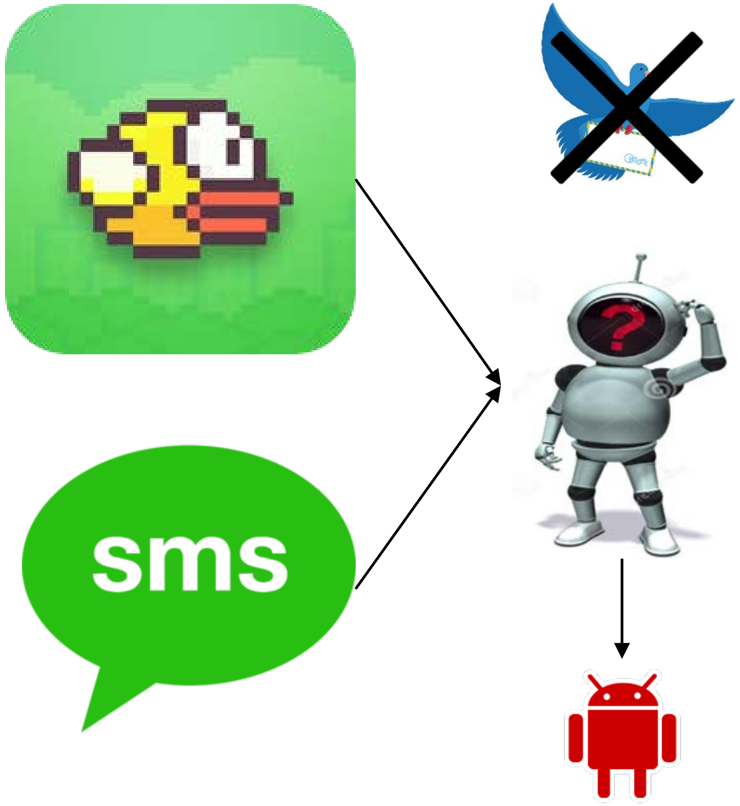
Allow this application to:

- **Your location**
  coarse (network-based) location, fine (GPS) location
- **Network communication**
  full Internet access
- **Storage**
  modify/delete SD card contents
- **Services that cost you money**
  directly call phone numbers
- **Phone calls**
  read phone state and identity

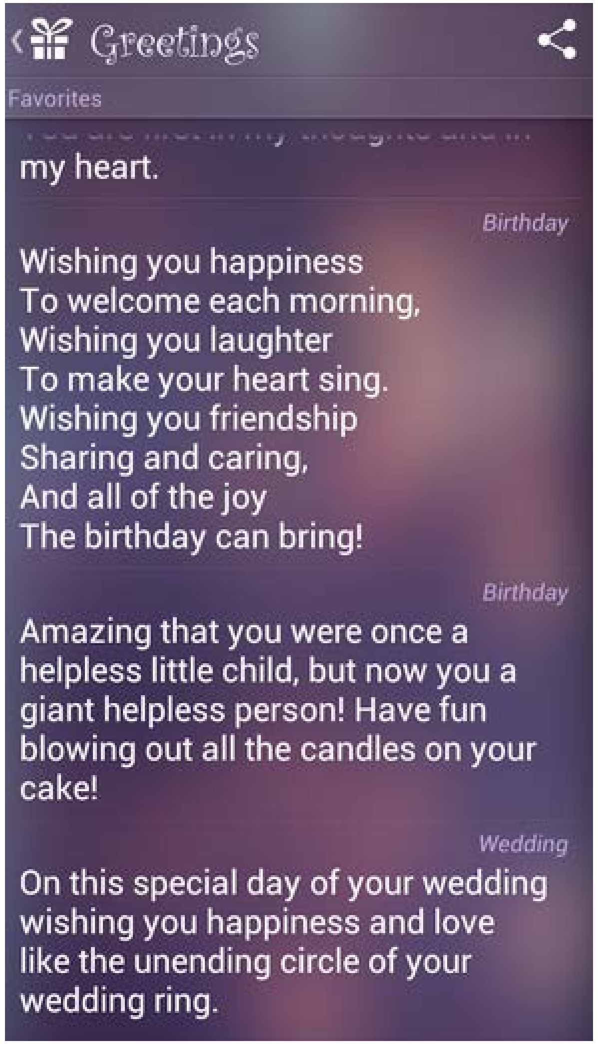
API Documents

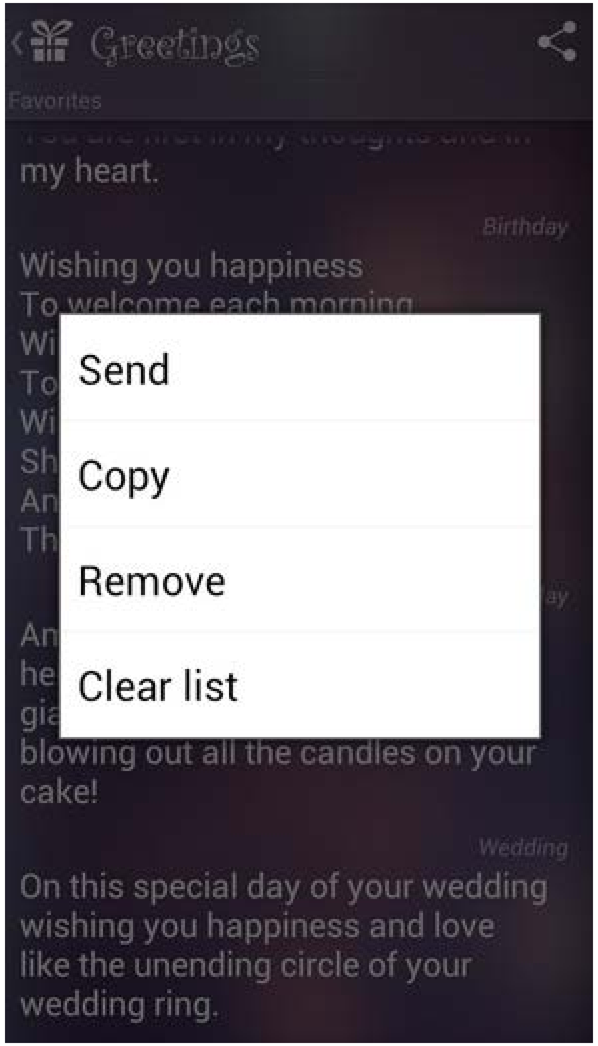
`SmsManager.sendTextMessage()`

Sensitive APIs



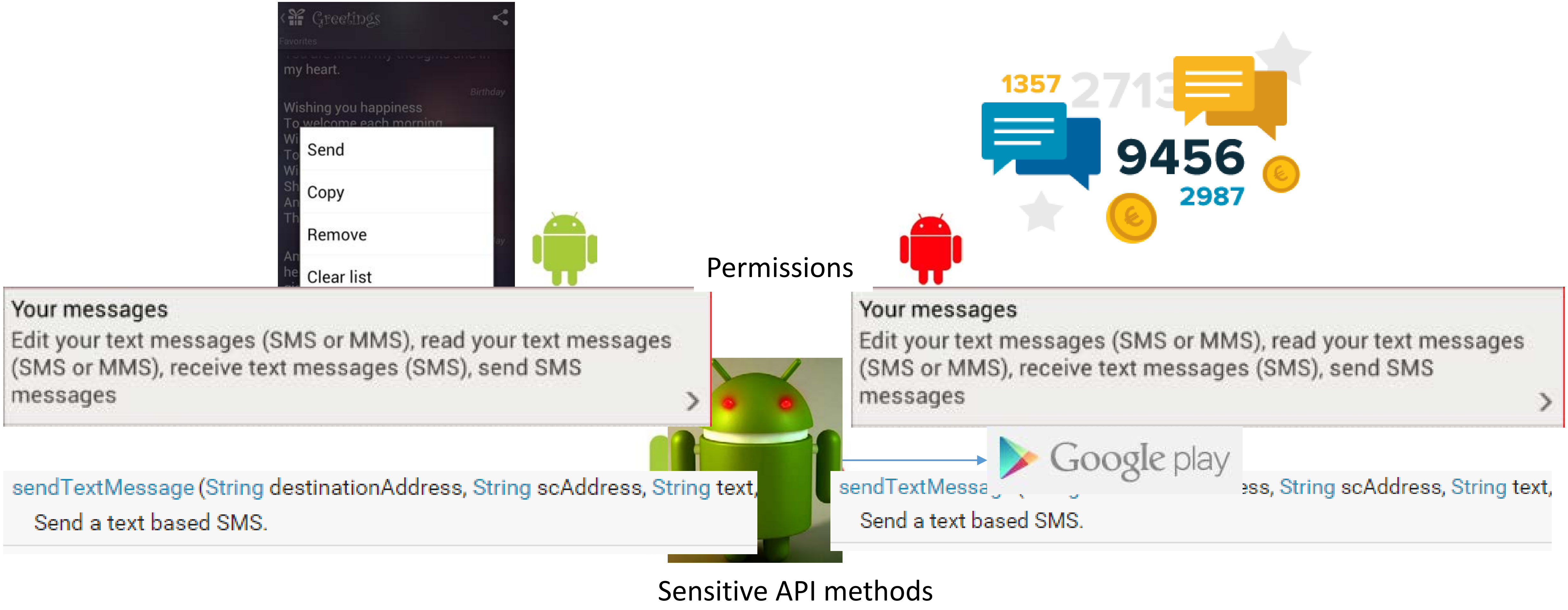
A benign app —— Greetings





13

An adaptive adversary



A research methodology

- What intrinsic property makes malware and benign apps different?

Problem Insights

Program Characteristics

Analysis Techniques

- What is the representation of such difference in the mobile programs?
- How to automatically extract such representation from mobile programs?

Our Insight

Different inherent goals of benign apps vs. malware as differentiating factors

- Benign apps
  - Meet requirements from users (as delivering utility)
- Malware
  - Trigger malicious behaviors frequently (as maximizing profits)
  - Evade detection (as prolonging lifetime)

Problem Insights — Program Characteristics — Analysis Techniques

Differentiating characteristics

Mobile malware (vs. benign apps)

- **Frequently enough** to meet the need: **frequent** occurrences of **imperceptible** system events;
  - E.g., many malware families trigger malicious behaviors via background events.

Security Insights — Program Characteristics — Analysis Techniques

Differentiating characteristics

Mobile malware (vs. benign apps)

- **Frequently enough** to meet the need: **frequent** occurrences of **imperceptible** system events;
  - E.g., many malware families trigger malicious behaviors via background events.

Balance!!!

- **Not too frequently** for users to notice anomaly: **indicative** states of external environments
  - E.g., Send premium SMS every 12 hours

Security Insights — Program Characteristics — Analysis Techniques

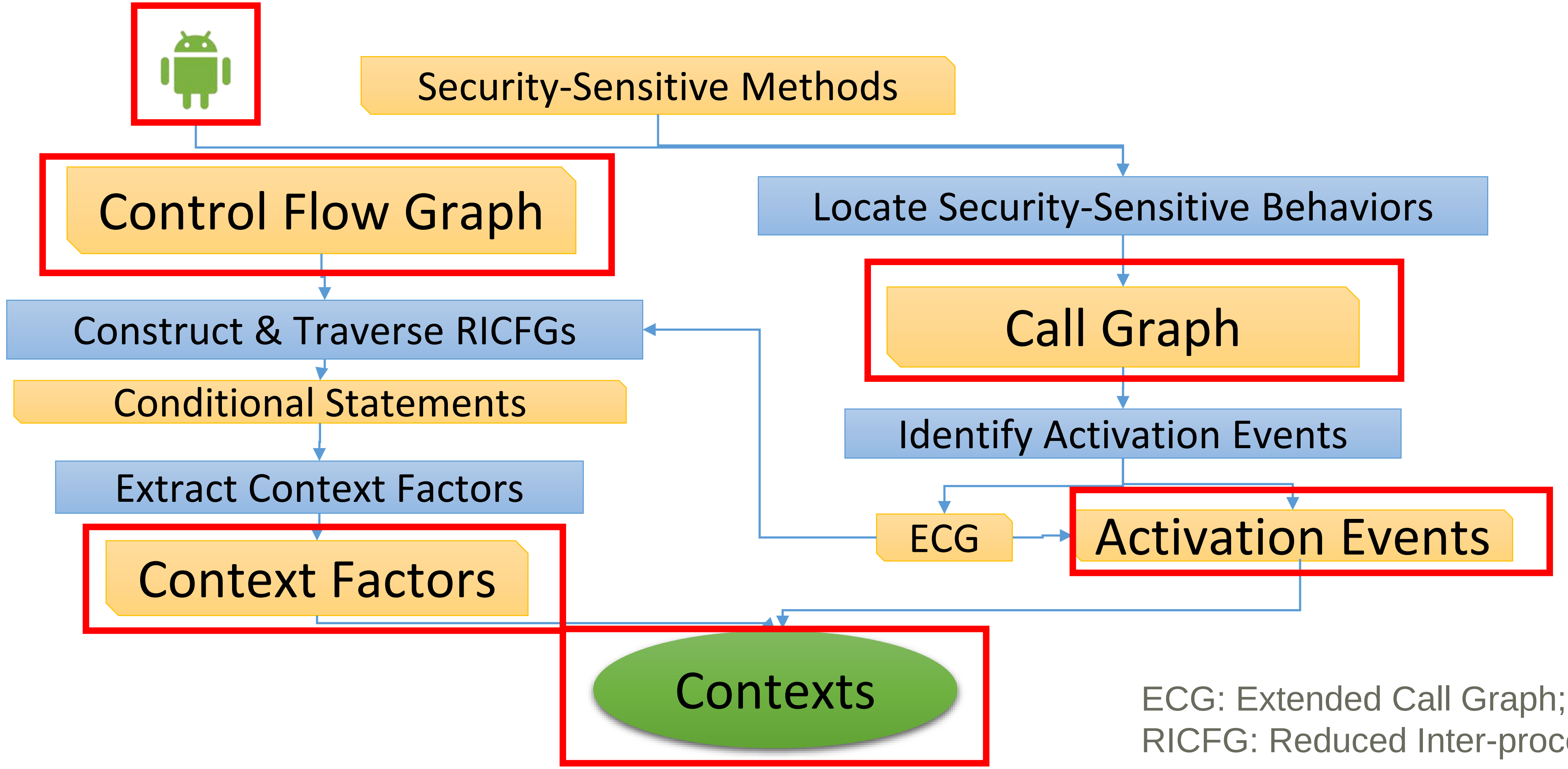
Differentiating characteristics

Mobile malware (vs. benign apps)

- **Frequently enough** to meet the need: **frequent** occurrences of **imperceptible** system events;
  - E.g., many malware families trigger malicious behaviors via background events.
  - Activation events, e.g., signal change
- **Not too frequently** for users to notice anomaly: **indicative** states of external environments
  - E.g., Send premium SMS every 12 hours
  - Context factors, e.g., current system time

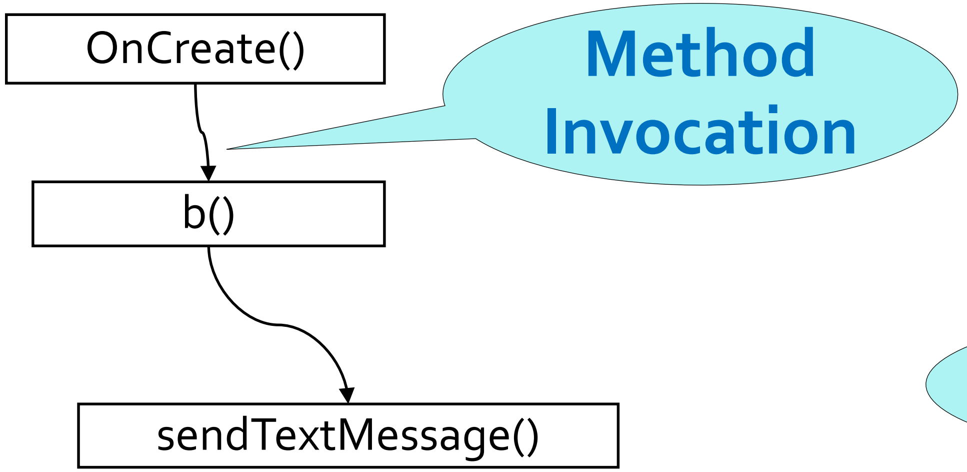
Security Insights — Program Characteristics — Analysis Techniques

How to extract contexts automatically?



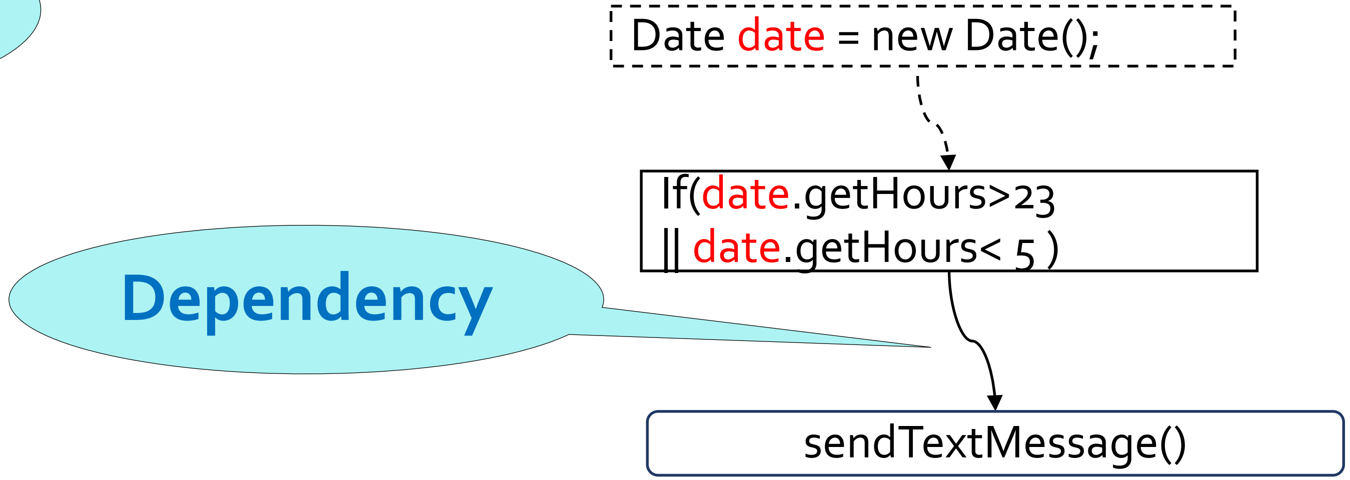
ECG: Extended Call Graph;
RICFG: Reduced Inter-procedure Control Flow Graph

How to extract contexts automatically?



Call Graph

Representing triggering relationship

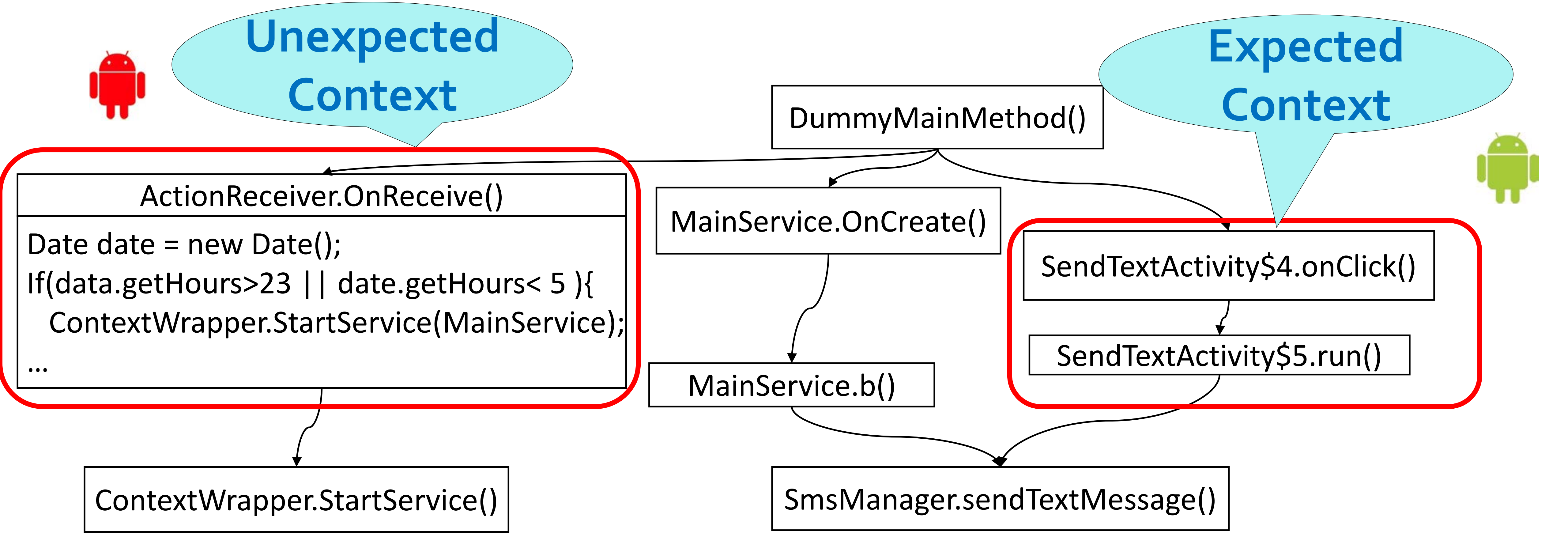


Control-flow Graph

Representing controlling relationship

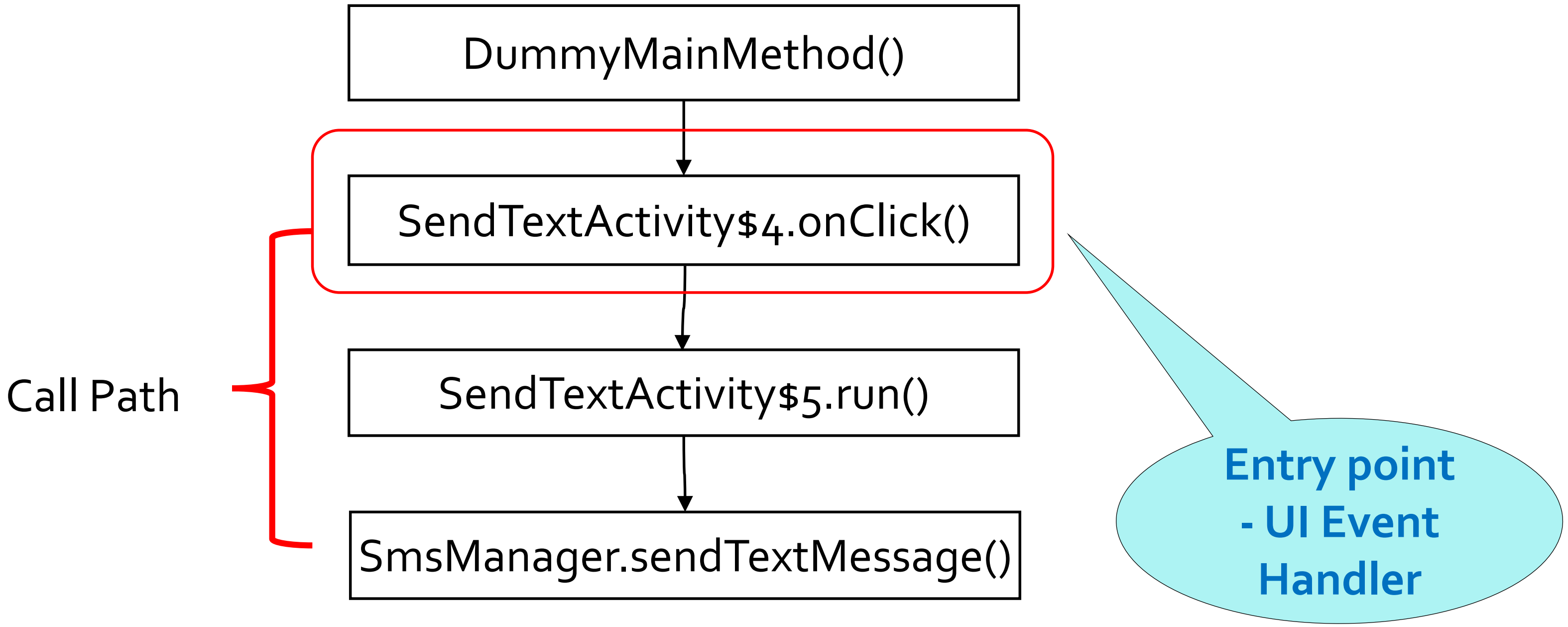
Security Insights — Program Characteristics — Analysis Techniques

Example



Unexpected Context
Expected Context
DummyMainMethod()
ActionReceiver.OnReceive()
Date date = new Date();
If(data.getHours>23 || date.getHours< 5){
ContextWrapper.StartService(MainService);
...
MainService.OnCreate()
MainService.b()
SendTextActivity$4.onClick()
SendTextActivity$5.run()
ContextWrapper.StartService()
SmsManager.sendTextMessage()

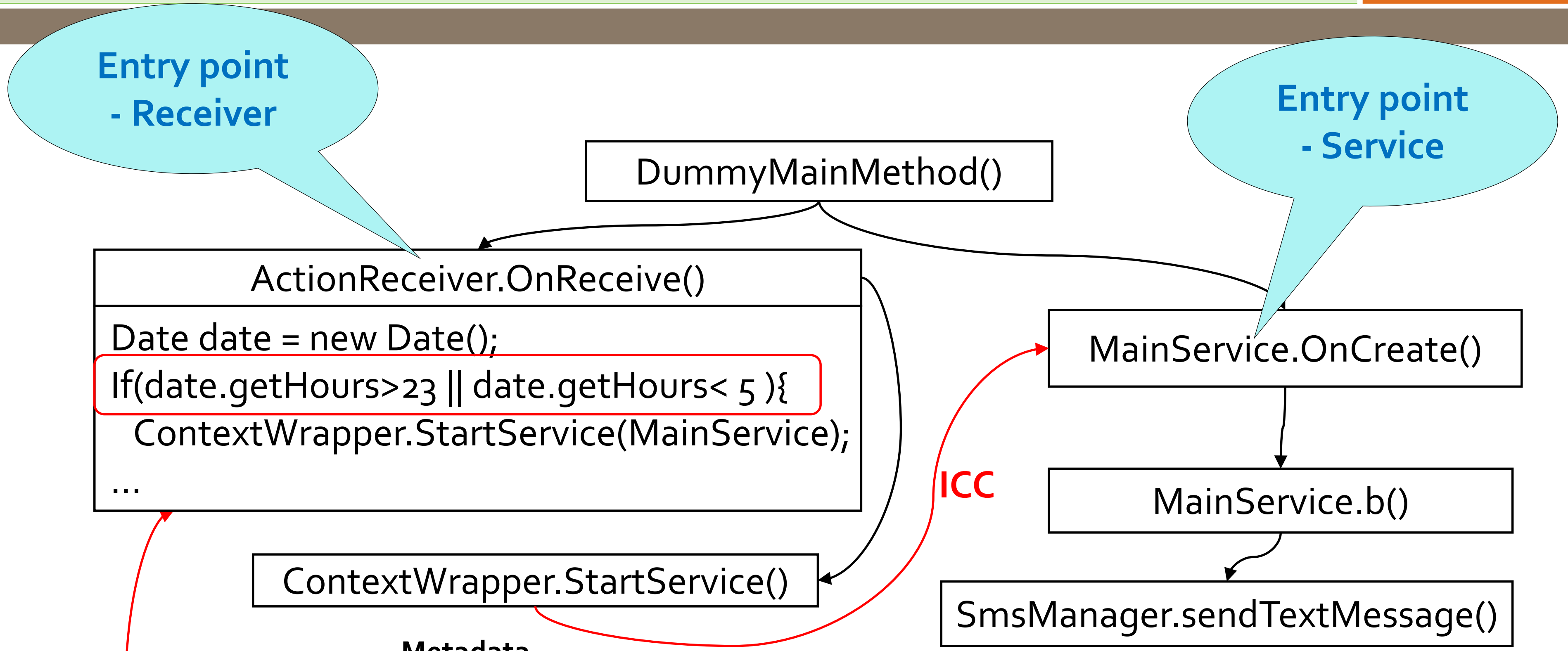
Expected Context

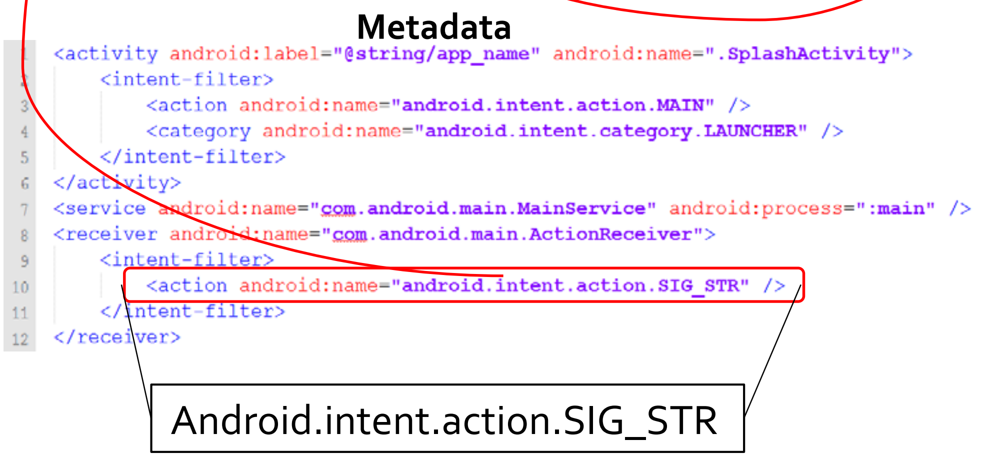


The app will send an SMS when

- user clicks a button in the app

Unexpected Context





The app will send an SMS when

- phone signal strength changes (frequent)
- current time is within 11PM-5 AM (not too frequent, User not around)

Context-based Security-Behavior Classification

Context1:
(Event: Signal strength changes),
(Factor: Calendar)

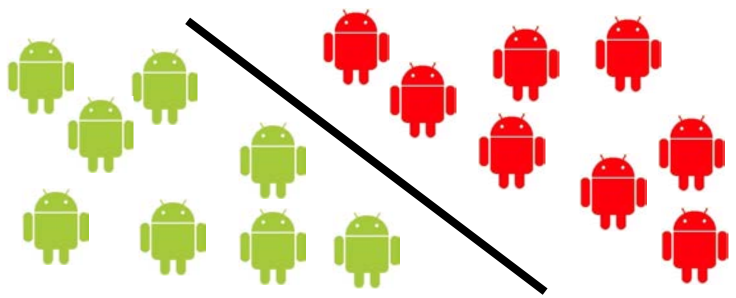
Context2:
(Event: Clicking a button)

Existing Features

Permission	Method	...	Hardware	System	UI	F_1	F_2	F_3	F_4	F_5	...
SEND_SMS	SendTextMessage	...	N/A	SIG_STR	N/A	0	0	1	0	0	...
SEND_SMS	SendTextMessage	...	N/A	N/A	Click	0	0	0	0	0	...

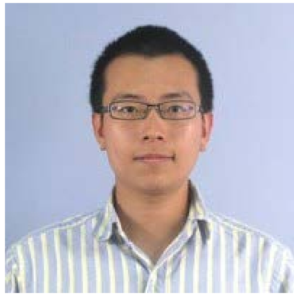
F_3 = Calendar

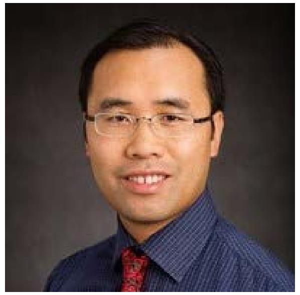
Support Vector Machine (SVM)
- Resilient to over-fitting
- Effective for high dimension data



Security Insights — Program Characteristics — Analysis Techniques

WHYPER: Towards Automating Risk Assessment of Mobile Applications

Usenix Security

It is NOT that People Don’t Care

BUSINESS INSIDER Tech Finance Politics Strategy Life Sports Video All

People were asked to read aloud the terms and conditions for popular apps and were shocked by what they actually agreed to

http://www.businessinsider.com/app-permission-agreements-privacy-video-2015-2

WHYPER: Automated Risk Assessment

- User Perceptions: **App Description**
- App Behaviors: **Permission Request**
- A framework using NLP techniques to construct traceability between a sentence in app description ←→ a permission of an app

Intermediate Representation Generator

Also you can share the yoga exercise to your friends via Email and SMS

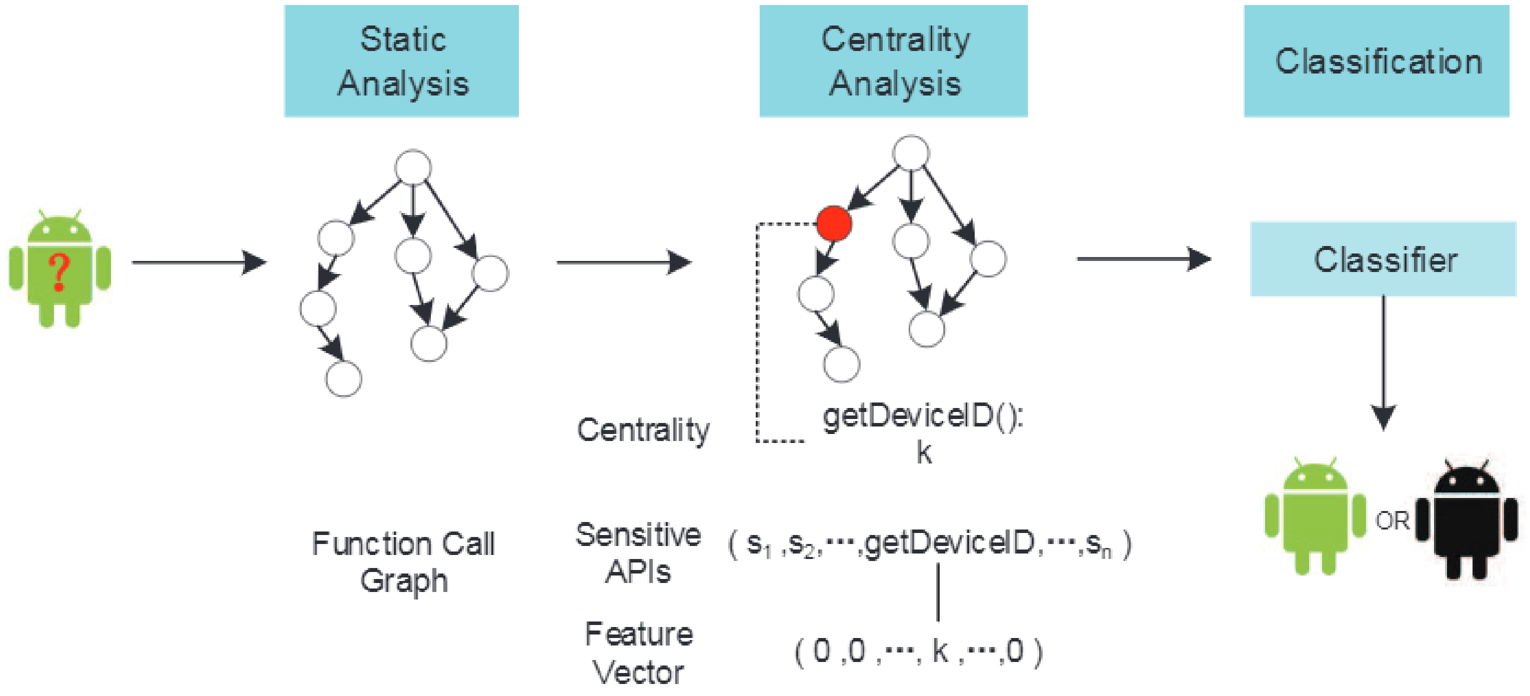
RB PRP MD VB DT NN NN PRP NNS NNP NNP

share
- **advmod** Also
- **nsubj** you
- **aux** can
- **dobj** exercise
  - **det** the
  - **nn** yoga
- **prep_to** friends
  - **poss** your
  - **prep_via** Email
    - **conj_and** SMS

to
- share
  - *you*
  - *yoga exercise*
- owned
  - *you*
  - via
    - *friends*
    - and
      - *email*
      - *SMS*

RB: adverb; PRP: pronoun; MD: verb, modal auxillary; VB: verb, base form; DT: determiner; NN: noun, singular or mass; NNS: noun, plural; NNP: noun, proper singular
http://www.clips.ua.ac.be/pages/mbsp-tags

MalScan: Fast Market-Wide Mobile Malware Scanning by Social-Network Centrality Analysis



ASE

Outline

- Application of learning-based techniques
  - Mobile Security
  - Testing
  - Other
- Quality assurance of learning-based techniques
- Security of learning-based techniques
- Interpretation of learning-based security techniques
- Future work

Our Past Work: Android App Testing

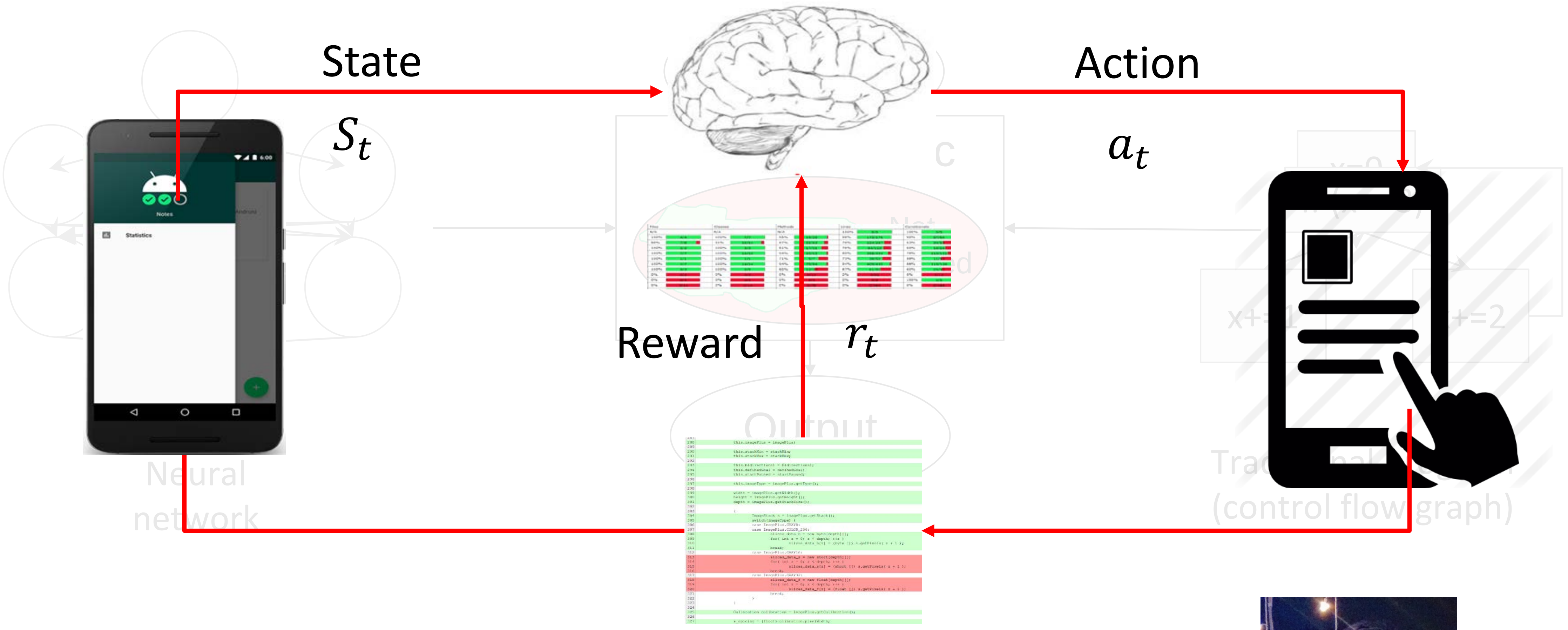
- 2 years of collaboration with Tencent Inc. WeChat testing team
  - Guided Random Test Generation Tool improved over Google Monkey
- Resulting tool deployed in daily WeChat testing practice
  - WeChat = WhatsApp + Facebook + Instagram + PayPal + Uber …
  - #monthly active users: **963 millions** @2017 2ndQ
  - Daily#: dozens of billion messages sent, hundreds of million photos uploaded, hundreds of million payment transactions executed
- First studies on testing industrial Android apps [FSE’16IN][ICSE’17SEIP][ASE’18]
  - Beyond open source Android apps focused by academia

WeChat

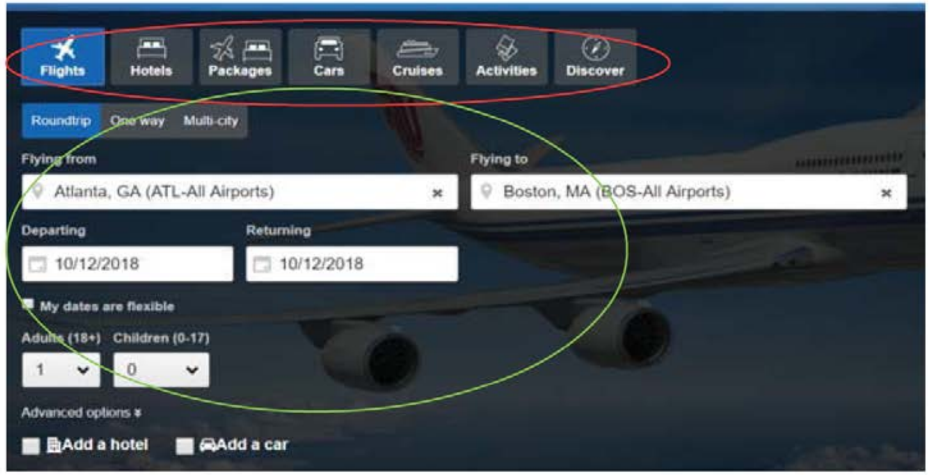
# of executable Java code lines:	610,629
# of Java classes:	8,425
# of Android activities:	607
# of C or C++ code lines:	~40,000

F-Droid

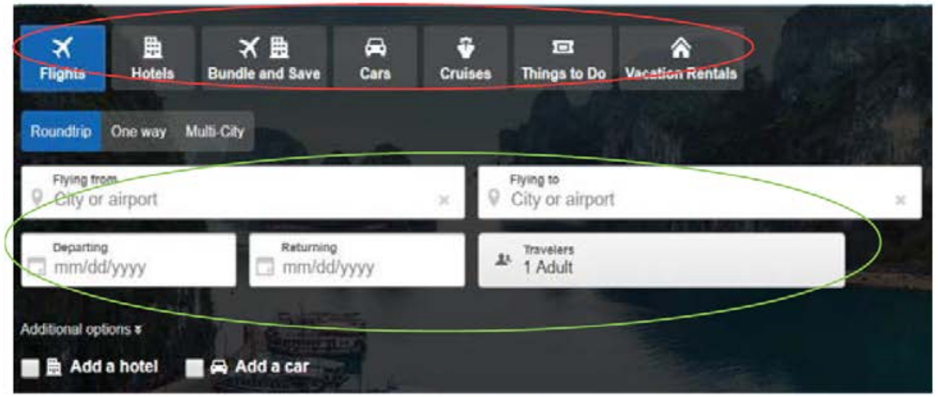
Now—— UI testing agent with reinforcement learning



Cooperative Mobile Testing



(a) Orbitz



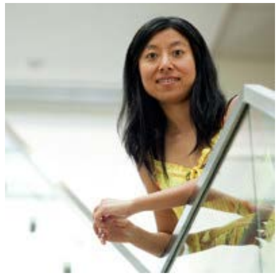
(b) Expedia

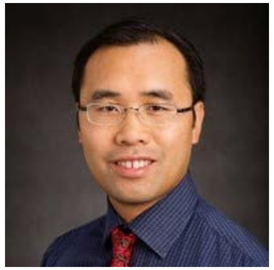
(c) Kayak

REINAM: Reinforcement Learning for Input-Grammar Inference.









FSE

Motivation

- Many programs take input strings that form a grammar.

Program	Valid Input String	Grammar
URIdecoder	https://google.com	URI = scheme:[//authority]path[?query][#fragment] authority = [userinfo@]host[:port] ……

- Knowing the grammar helps us understand the input structure.

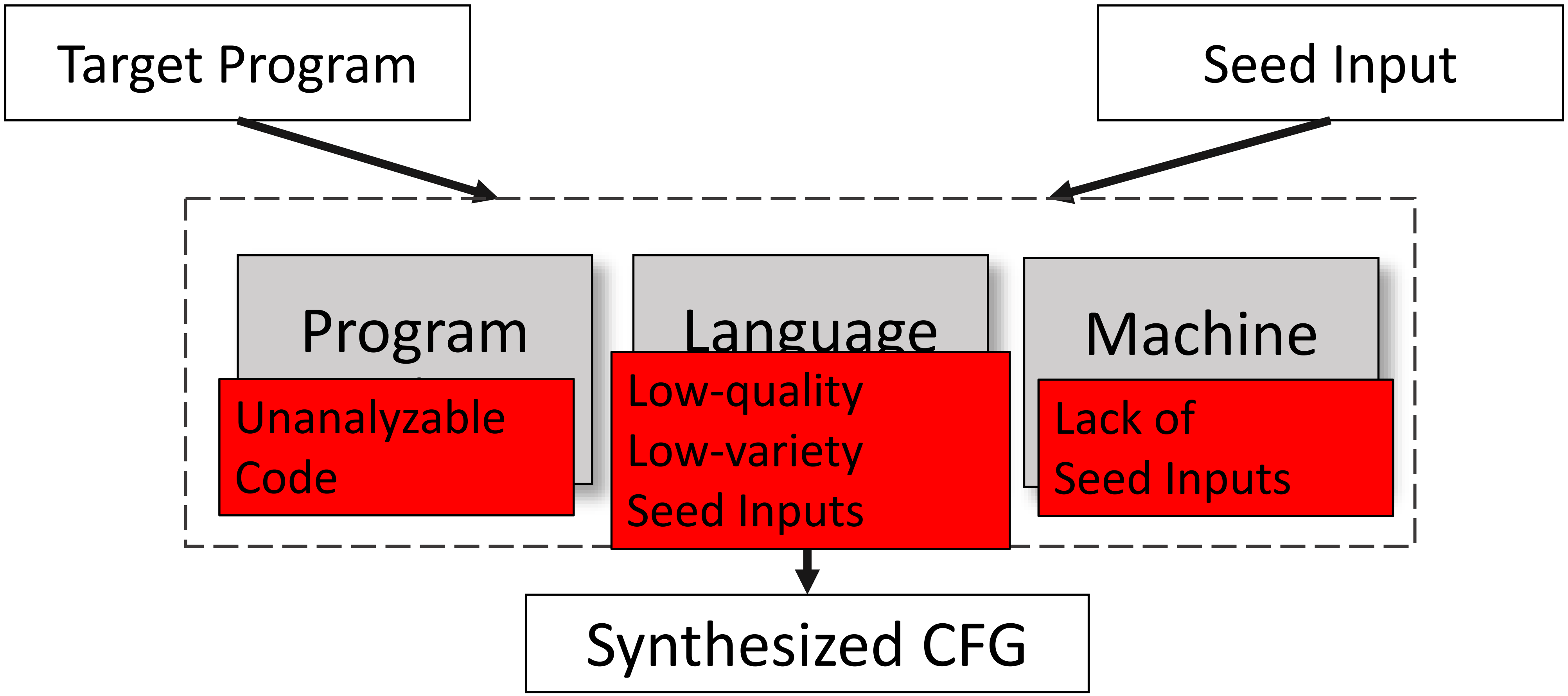
Application

- Input grammar could be useful in a wide range of applications:

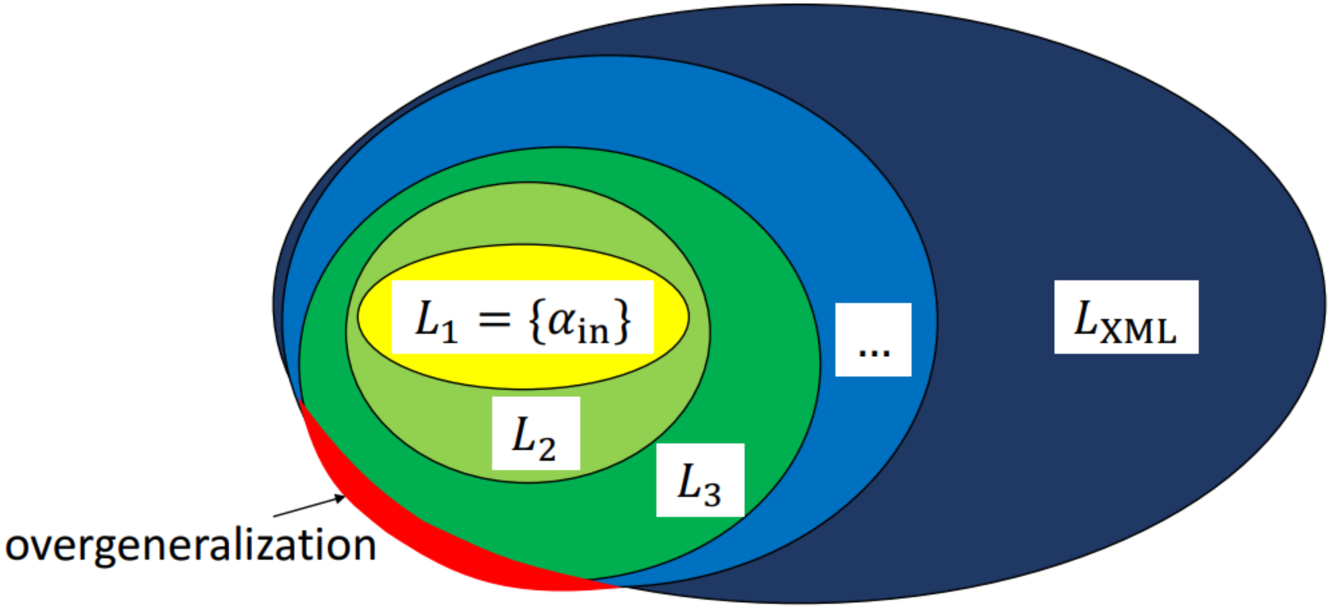
Fuzz Testing

Reverse Engineering

Existing Approach



Existing Approach

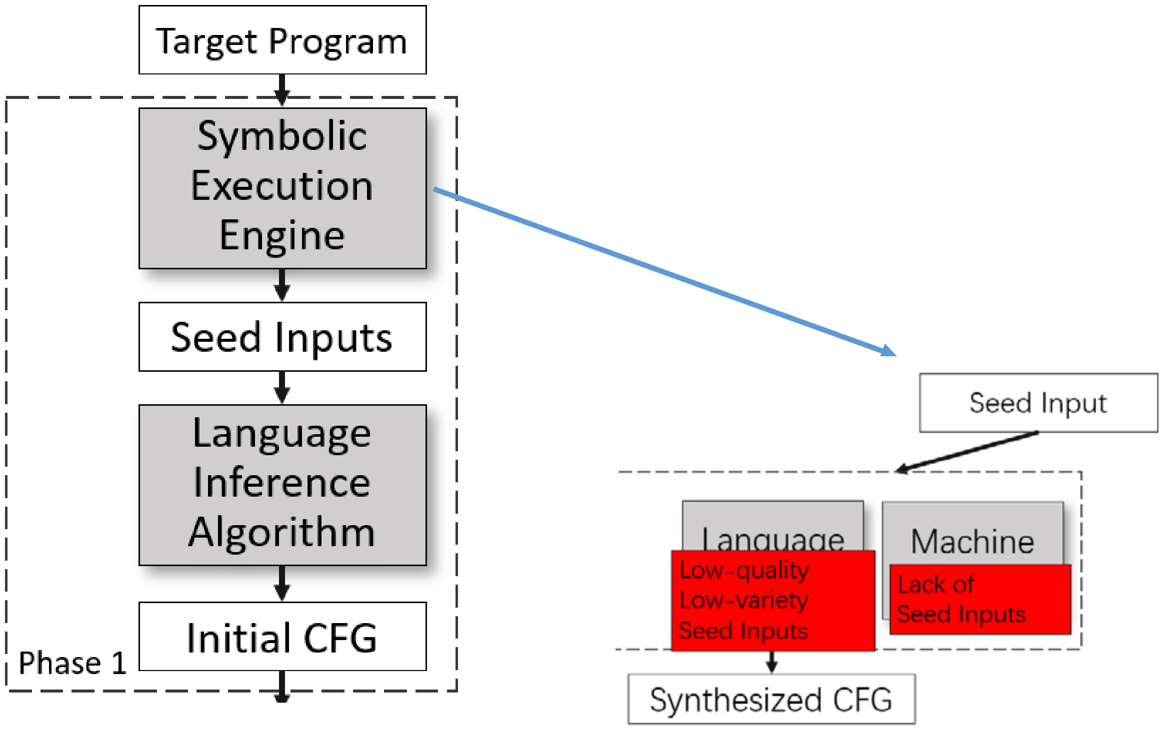


Active Learning Approach[1]

[1] Osbert Bastani, Rahul Sharma, Alex Aiken, and Percy Liang. Synthesizing program input grammars in PLDI'17

- State-of-the-art approaches use **active learning** to iteratively generalize the grammar.
- However, they work hard to try to avoid any **overgeneralization** which could be useful.

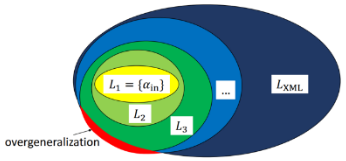
Workflow of REINAM



- **REINAM** takes the target program as input.
- **Phase 1**: REINAM generates seed inputs using automatic test generation and then uses a grammar synthesizer to synthesize an initial CFG.
- Using **dynamic symbolic execution engine** in automatic test generation alleviates the shortcoming of low-quality, low-variety and insufficient seed inputs.

Workflow of REINAM

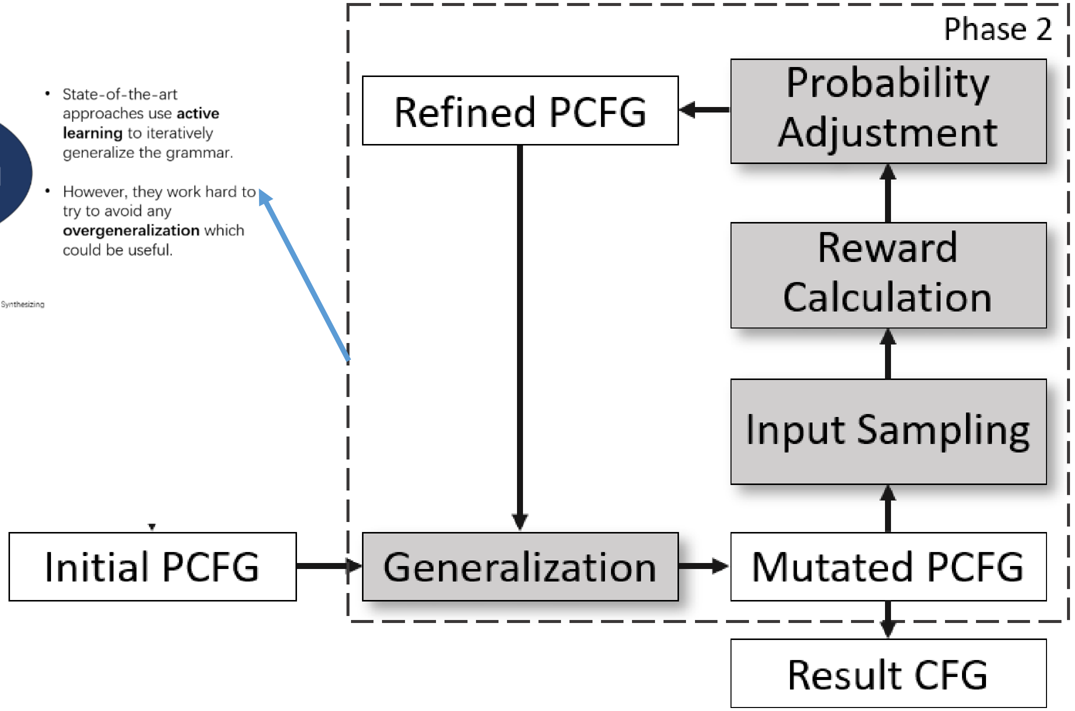
- **Phase 2**: REINAM converts the CFG from Phase 1 to a PCFG, and then uses **reinforcement learning** to refine this PCFG.



- State-of-the-art approaches use **active learning** to iteratively generalize the grammar.
- However, they work hard to try to avoid any **overgeneralization** which could be useful.

Active Learning Approach[1]

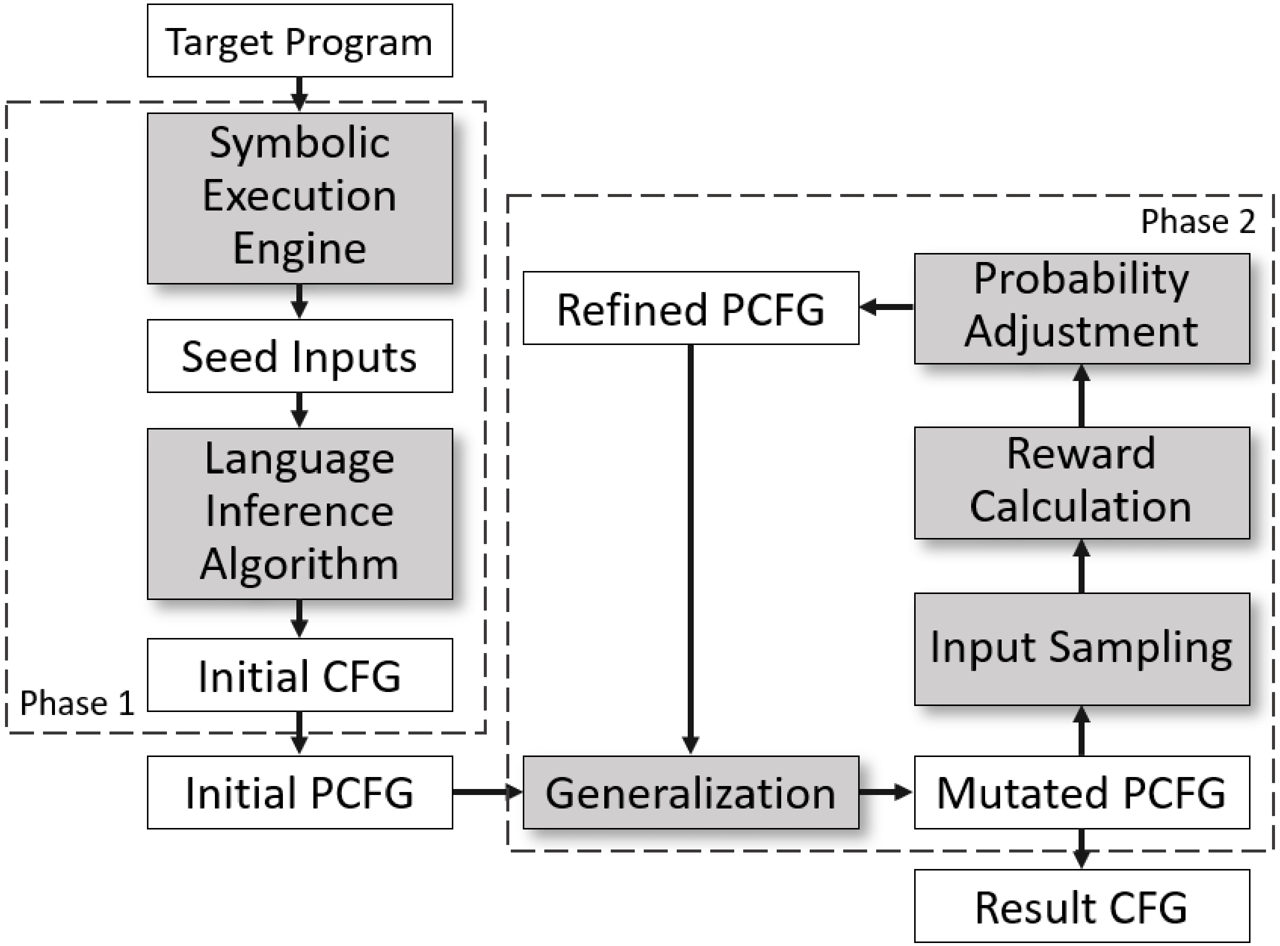
[1] Osbert Bastani, Rahul Sharma, Alex Aiken, and Percy Liang. Synthesizing program input grammars in PLDI'17



- To allow **Overgeneralization,** We present the grammar of the program as a **Probalistic Context-Free Grammar (PCFG)** rather than a deterministic **Context-Free Grammar (CFG).**

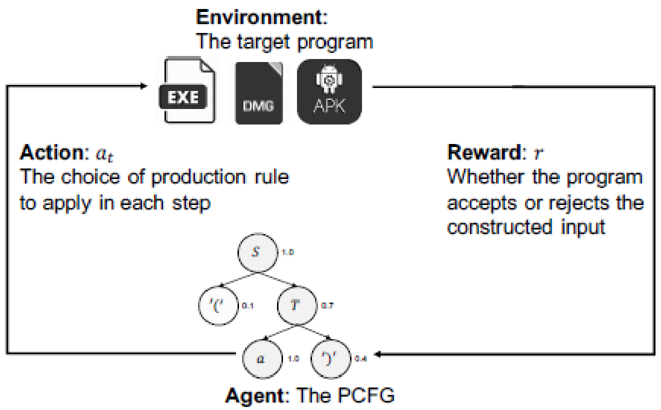
- To optimize the **PCFG**, we formulate the **Input Grammar Synthesis** task as a **Reinforcement Learning** problem.

Workflow of REINAM

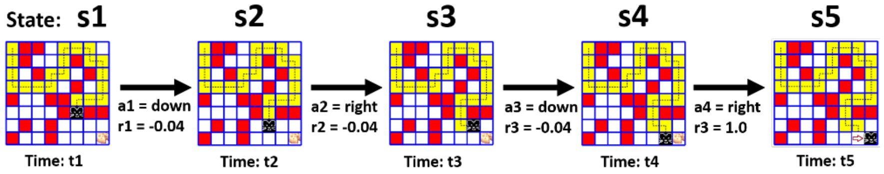


Generalizing PCFG via Reinforcement Learning

Task	Construct valid input string	Solve maze
Agent	The PCFG	The robot
Environment	The target program	The maze
State	Current state of the string (a partial derivation)	(Row, Column, Last action)
Action	The choice of production rule to apply	The choice of the direction to move
Reward	Whether the constructed input is accepted or not	-0.04 for each move, +1 for hitting target



Reinforcement Learning for Input Construction



Reinforcement Learning for Maze Solving [2]

[2] https://www.samyzaf.com/ML/rl/qmaze.html

Outline

- Application of learning-based techniques
  - Mobile Security
  - Testing
  - Other

- Quality assurance of learning-based techniques
- Security of learning-based techniques
- Interpretation of learning-based security techniques
- Future work

SemRegex: A Semantics-Based Approach for Generating Regular Expressions from Natural Language Specifications

Zhong et al.

EMNLP 2018

Problem Statement

Generating Regular Expressions from NL

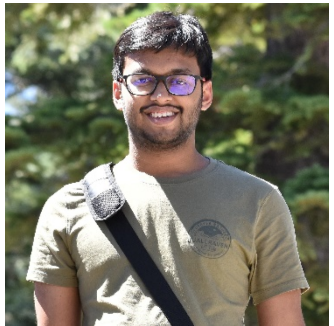
regex × 197003

Regular expressions provide a declarative language to match patterns within strings. They are commonly used for string validation, parsing, and transformation. Since regular expressions are not fully …

96 asked today, 418 this week

NL: String that begin with at least two digits.

Regex: `(([0-9]){2,})(.*)`



Challenges

Program Aliasing

Domain	NL Specification	Program 1	Program 2
Regex	Match lines that start with an uppercase vowel and end with 'X'	`([AEIOUaeiou]&[A-Z]).*X`	`([AEIOU].*)&(.*X)`
Bash	Rename file 'f1' to 'f1.txt'	`mv 'f1' 'f1.txt'`	`cp 'f1' 'f1.txt'; rm 'f1'`
Python	Assign the greater value of 'a' and 'b' to variable 'c'	`c = a if a > b else b`	`c = [b, a][a > b]`

A semantically equivalent program may have various syntactically different forms.

Cross-language Vulnerability Detection

SARD dataset	CWEs	#	CWE		Languages	Fixes similar	Good cases in Test Suit (Java/C/C++)
Java	112	1	CWE: 15 External Control of System or Configuration Setting	https://cwe.mitre.org/	Java,C	yes	no
C++	118	2	CWE: 23 Relative Path Traversal	https://cwe.mitre.org/	independent	yes	no
Common	**56**	3	CWE: 36 Absolute Path Traversal	https://cwe.mitre.org/	independent	yes	no
		4	CWE: 78 OS Command Injection	https://cwe.mitre.org/	independent	yes	no
		5	CWE: 90 LDAP Injection	https://cwe.mitre.org/	independent	yes	no
		6	CWE: 114 Process Control	https://cwe.mitre.org/	independent		no
		7	CWE: 134 Uncontrolled Format String	https://cwe.mitre.org/	Java,C	yes	yes
		8	CWE: 190 Integer Overflow	https://cwe.mitre.org/	independent	yes	yes
		9	CWE: 191 Integer Underflow	https://cwe.mitre.org/	Java,C	yes	yes
		10	CWE: 197 Numeric Truncation Error	https://cwe.mitre.org/	Java,C	no	no
		11	CWE: 252 Unchecked Return Value	https://cwe.mitre.org/	independent	yes	yes
		12	CWE: 253 Incorrect Check of Function Return Value	https://cwe.mitre.org/	independent	yes	yes
		13	CWE: 256 Plaintext Storage of a Password. Read the password from a Properties file or a regular file. In the good case, read the file from the console.	https://cwe.mitre.org/	independent	yes	yes
		14	CWE: 259 Hard Coded Password	https://cwe.mitre.org/	independent	yes	no
		15	CWE-319: Cleartext Transmission of Sensitive Information	https://cwe.mitre.org/	independent	yes	yes
		16	CWE-325: Missing Required Cryptographic Step	https://cwe.mitre.org/	independent	yes	yes
		17	CWE-327: Use of a Broken or Risky Cryptographic Algorithm	https://cwe.mitre.org/	independent	yes	yes
		18	CWE-328: Reversible One-Way Hash	https://cwe.mitre.org/	independent	yes	yes
		19	CWE-338: Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)	https://cwe.mitre.org/	independent	yes	yes
		20	CWE-369: Divide By Zero	https://cwe.mitre.org/	Java,C,C#	yes	yes
		21	CWE-390: Detection of Error Condition Without Action	https://cwe.mitre.org/	independent	no	yes
		22	CWE-396: Declaration of Catch for Generic Exception	https://cwe.mitre.org/	Java,C,C#	no	yes
		23	CWE-397: Declaration of Throws for Generic Exception	https://cwe.mitre.org/	Java,C,C#	no	yes
		24	CWE: 398 Indicator of Poor Code Quality	https://cwe.mitre.org/	Java,C	yes	yes
		25	CWE-400: Uncontrolled Resource Consumption	https://cwe.mitre.org/	independent	yes	yes

→ φ

Outline

- Application of learning-based techniques
- Quality assurance of learning-based techniques
- Security of learning-based techniques
- Interpretation of learning-based security techniques

Price of Autonomy

- Deployment scale: too large for humans to effectively monitor
  - Sculley et al., 2015

Price of Autonomy

- Deployment scale: too large for humans to effectively monitor
  - Sculley et al., 2015

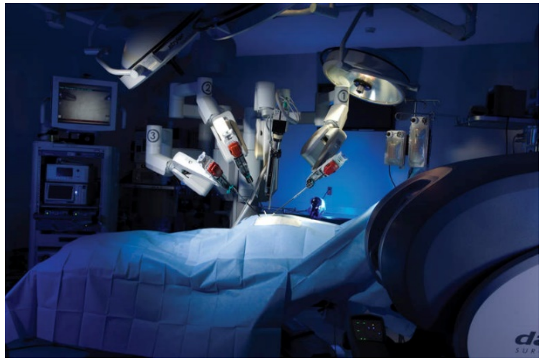
- Time scale: too short to wait for human feedback
  - autonomous vehicles: Temizer et al., 2010; Geiger et al., 2012

Price of Autonomy

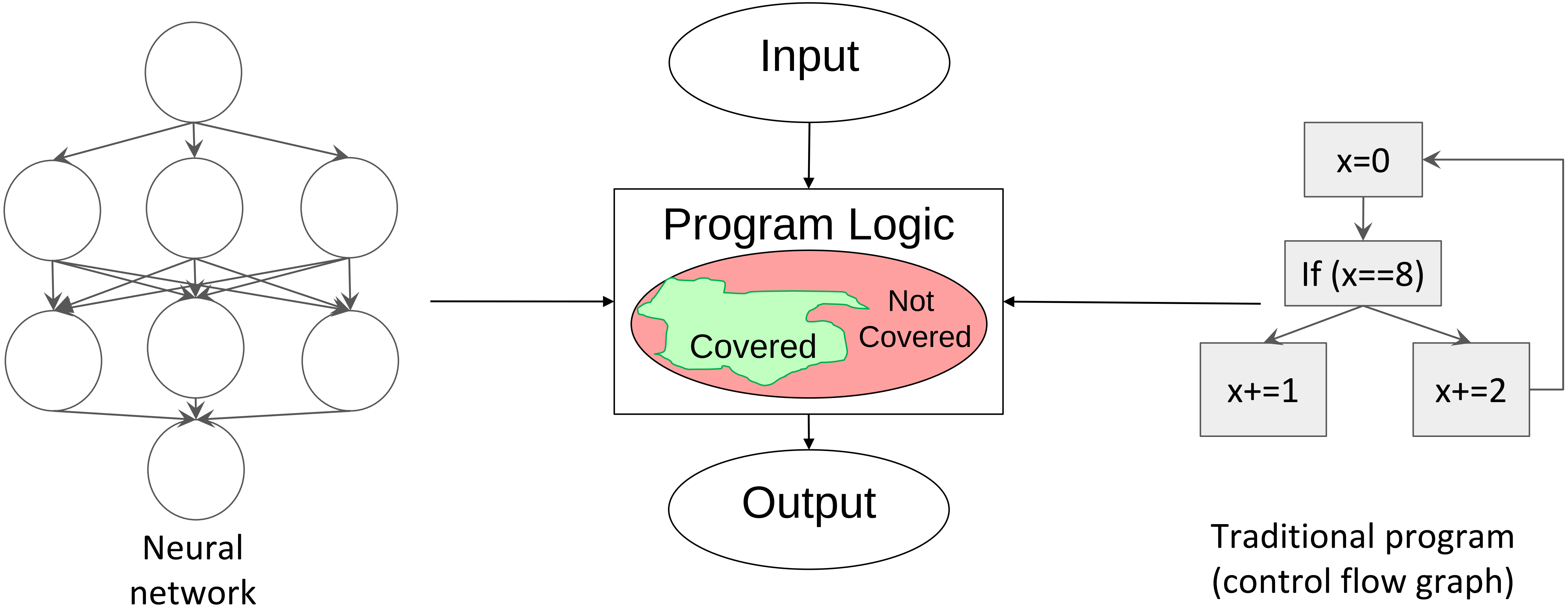
- Deployment scale: too large for humans to effectively monitor
  - Sculley et al., 2015

- Time scale: too short to wait for human feedback
  - autonomous vehicles: Temizer et al., 2010; Geiger et al., 2012

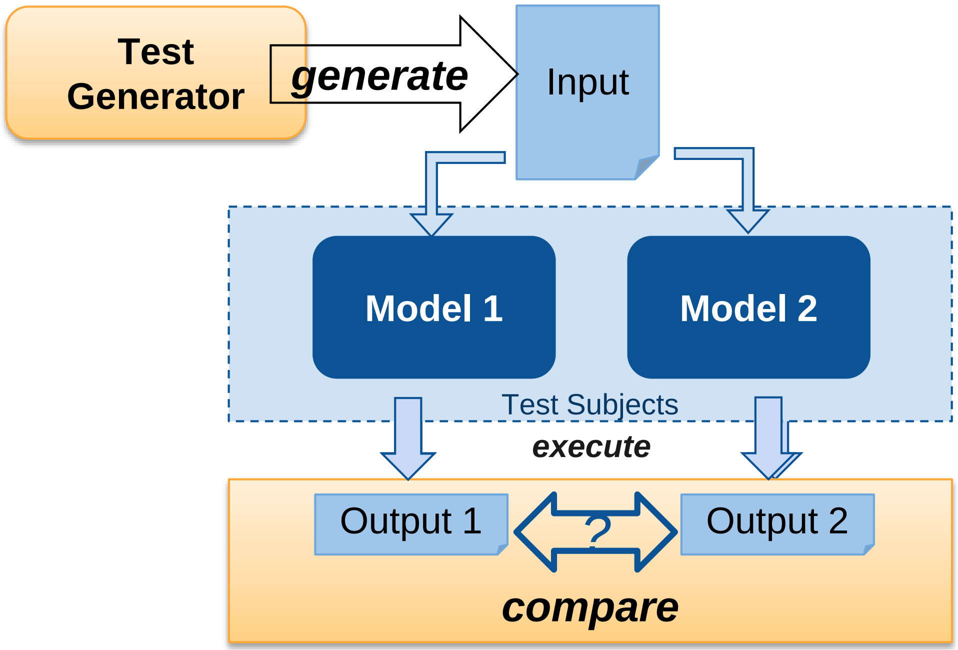
- Stakes: too high to tolerate errors
  - surgery: Taylor et al., 2008

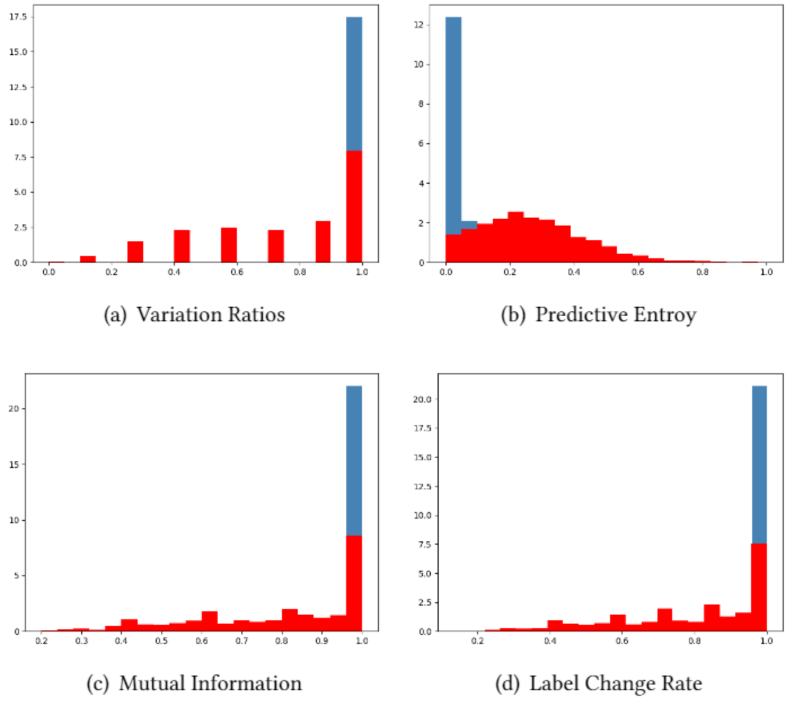


Testing Machine Learning Apps



Automated Generation of Test Oracle for Deep Learning Application

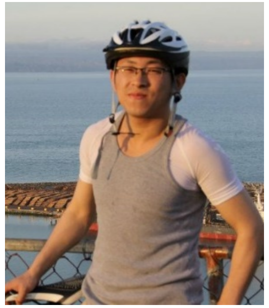




(a) Variation Ratios (b) Predictive Entroy

(c) Mutual Information (d) Label Change Rate

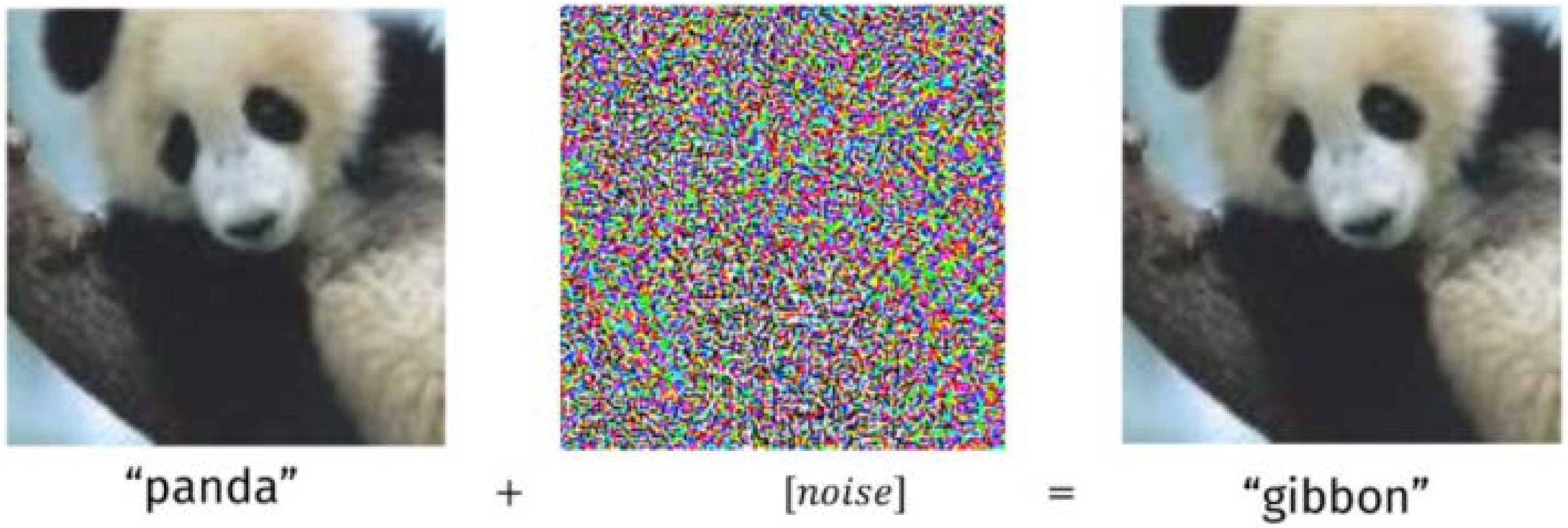
https://sites.google.com/view/dloracle/home

Outline

- Application of learning-based techniques
- Quality assurance of learning-based techniques
- Security of learning-based techniques
- Interpretation of learning-based security techniques
- Future work

Adversarial Machine Learning



Example from: Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. ICLR 2015.

Adversarial Machine Learning

(Eykholt et al, 2017)

Outline

- Application of learning-based techniques
- Quality assurance of learning-based techniques
- Security of learning-based techniques
  - Privacy of DNN
  - Attacking Malware Detector
  - Other
- Interpretation of learning-based security techniques

DNN Privacy

Property Inference Attacks on Deep Neural Networks using Permutation Invariant Representations





CCS

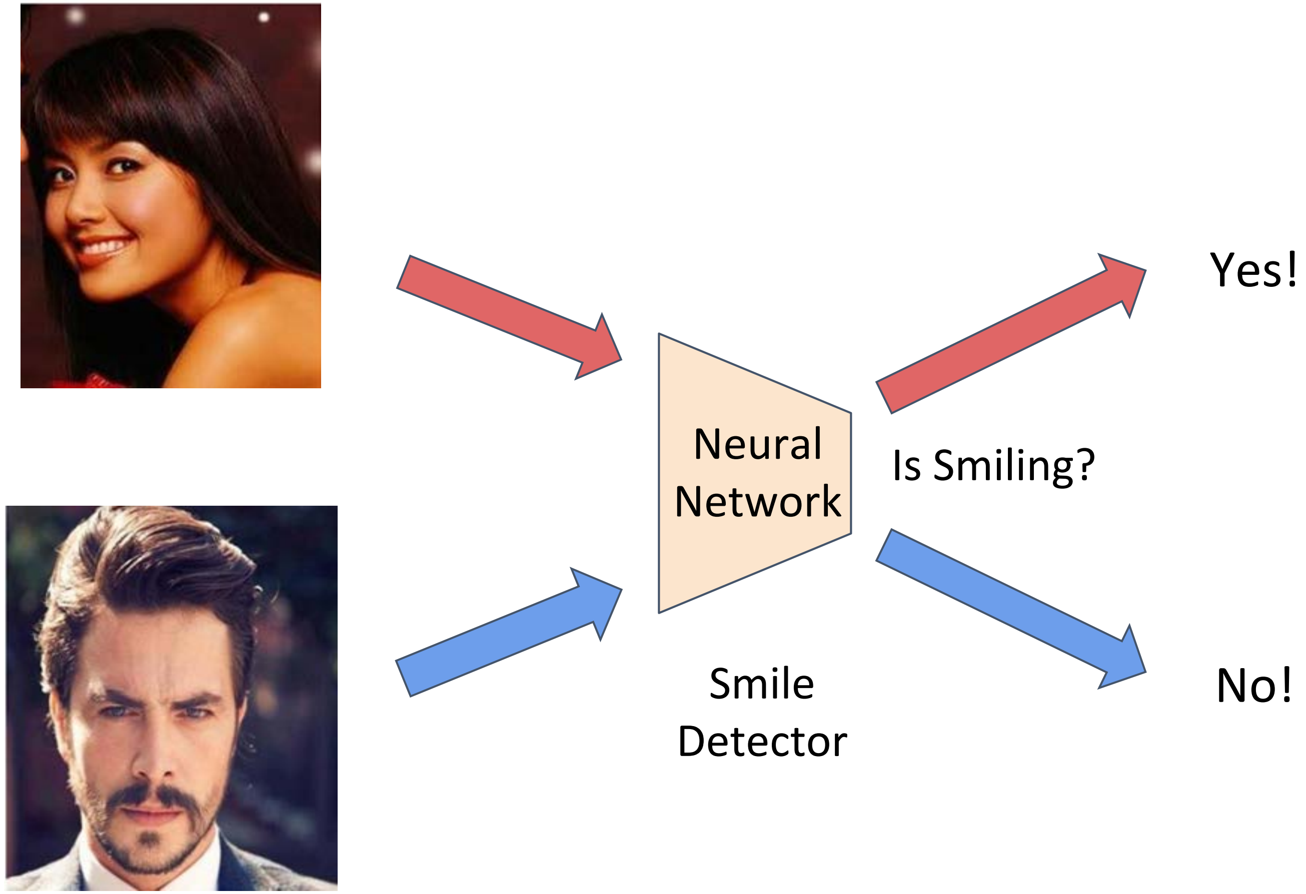
Property Inference Attack

Given a whitebox ML model, can the model consumer (attacker) infer some global properties of the training dataset the model producer did not intend to share?

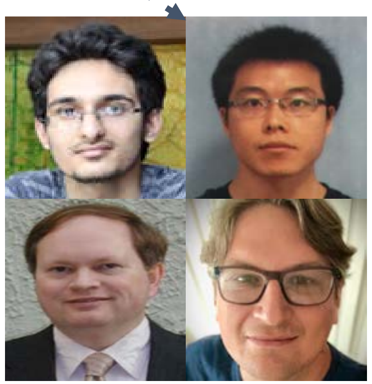
E.g., the environment in which the data was produced

E.g., the fraction of the data that comes from a certain class

An example: Smile detector

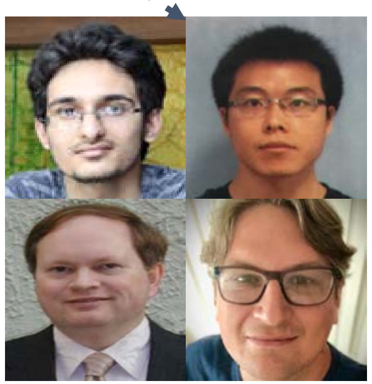


An example: A simple property of the training dataset



An example: A simple property of the training dataset

or



P: Skewed towards attractive people

$\bar{P}$: Only ordinary people

An example: A simple property of the training dataset

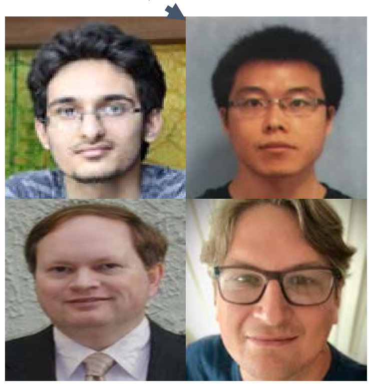
train

Neural Network

Smile Detector

infer

or

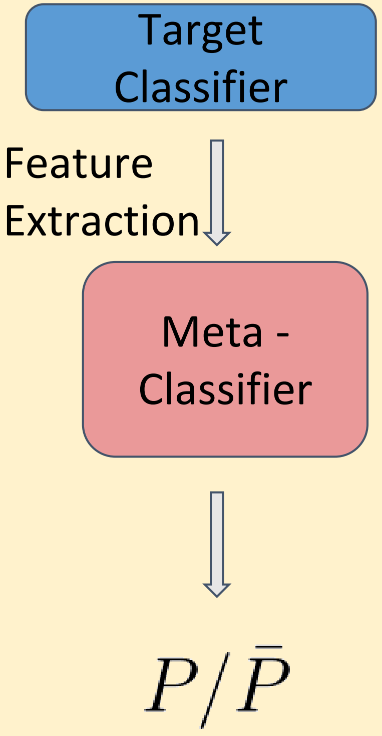


P: Skewed towards attractive people

$\bar{P}$: Only ordinary people

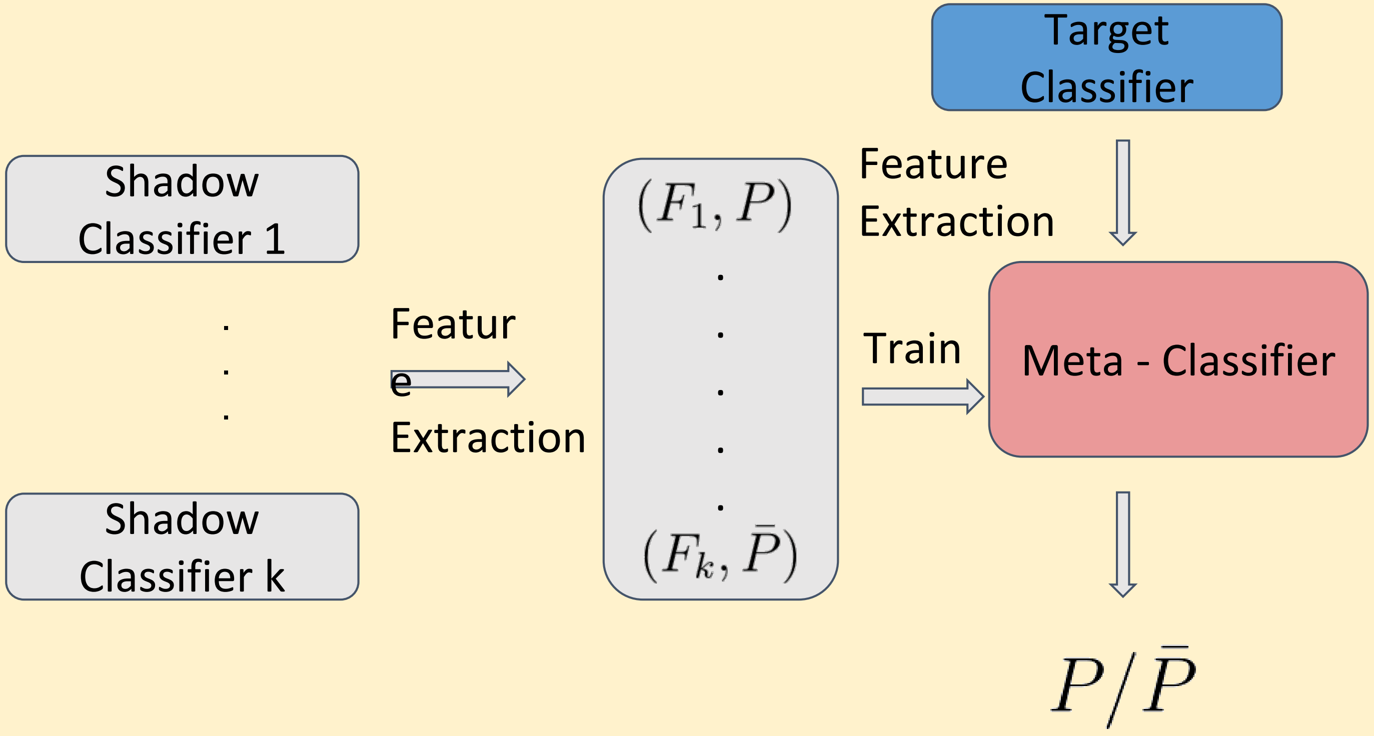
Property Inference Attack Strategy: Meta-training

Models trained on similar datasets using similar training methods should represent similar functions!



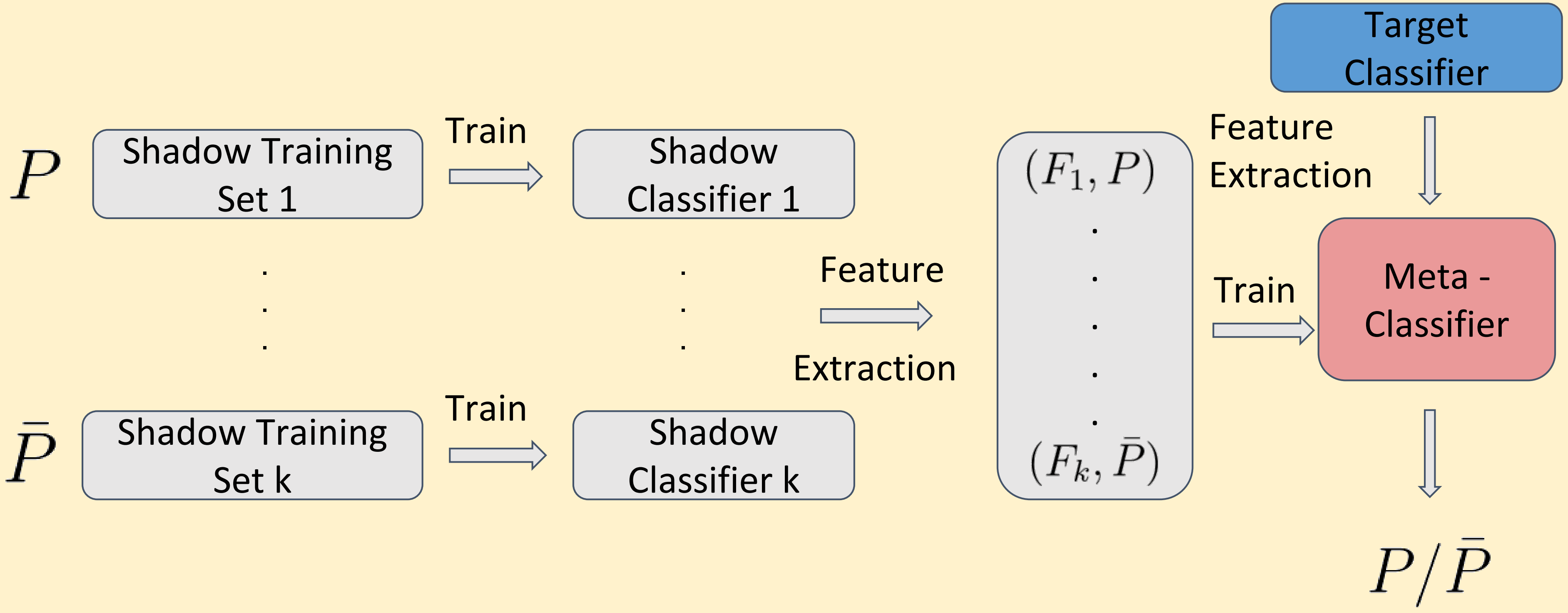
Property Inference Attack Strategy: Meta-training

Models trained on similar datasets using similar training methods should represent similar functions!



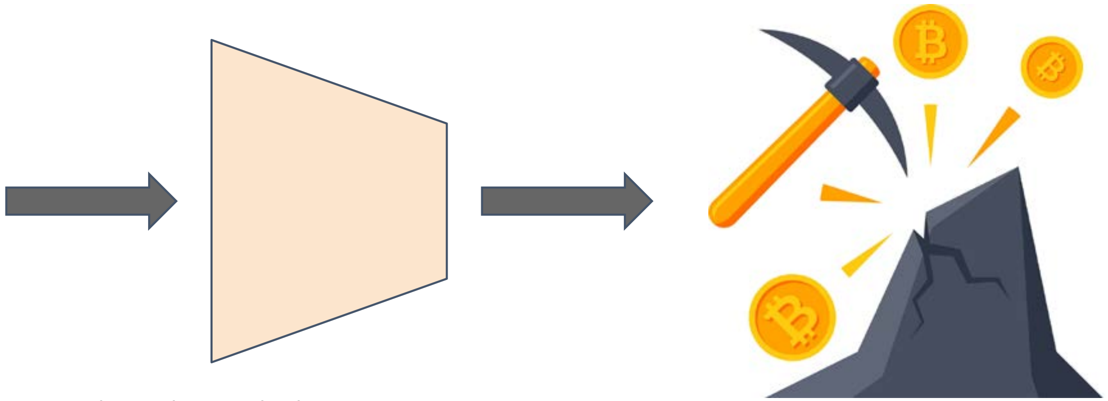
Property Inference Attack Strategy: Meta-training

Models trained on similar datasets using similar training methods should represent similar functions!



Case study: Inferring vulnerabilities

Hardware performance counters values for different applications on a desktop

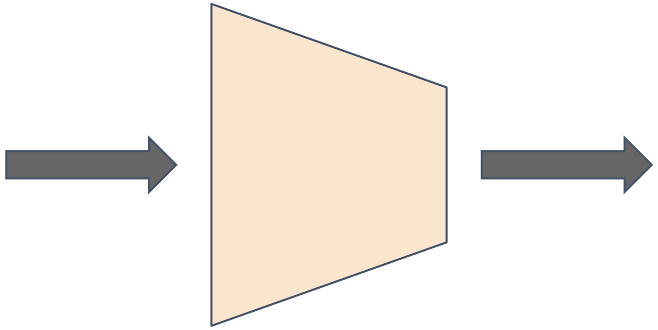


Bitcoin Mining Detector

[Tahir et al. RAID’17]

Case study: Inferring vulnerabilities

Hardware performance counters values for different applications on a desktop



Bitcoin Mining Detector

[Tahir et al. RAID'17]

Property Inference

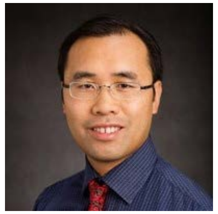
Patched or Not?

Outline

- Application of learning-based techniques
- Quality assurance of learning-based techniques
- Security of learning-based techniques
  - Privacy of DNN
  - Attacking Malware Detector
  - Other
- Interpretation of learning-based security techniques

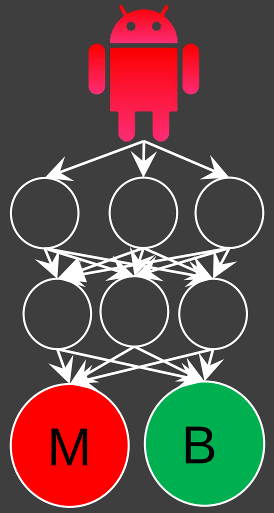
Attacking Learning-based Malware Detector

Malware Detection in Adversarial Settings: Exploiting Feature Evolutions and Confusions in Android Apps



ACSAC

Why not using everything in malware bytecode as features?



Magic happens?

- Performance
  - Slower and need more labelled data.

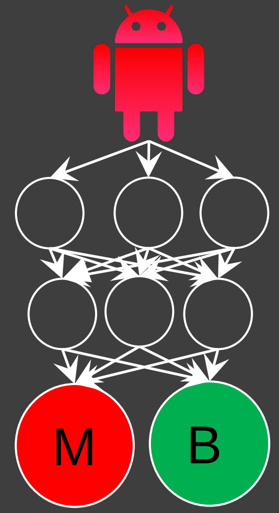
- Discriminative features are not resilient in adversarial settings

Insight: Malware detectors often include non-essential features in code clones as discriminative features.

```
for (int i =0; i < n/2; i++){
        char temp = a[i];
        a[i] = a[n-1-i];
        a[n-1-i] = temp;
} //reverse the SMS message
……
for (int i =0; i < n/2; i++){
        char temp = a[i];
        a[i] = a[n-1-i];
        a[n-1-i] = temp;
} //reverse the SMS message again
sendTextMessage(a);
```

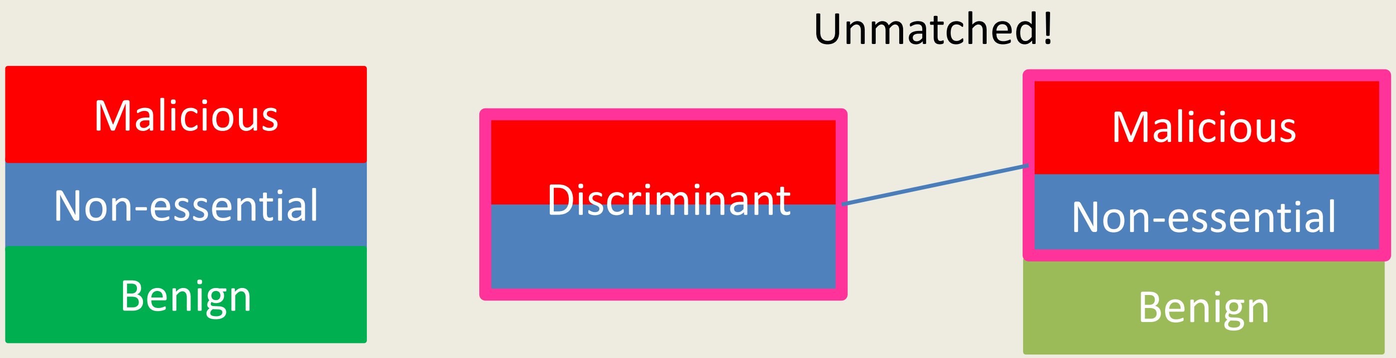
Code clones

Why not using everything in malware bytecode as features?



Magic happens?

- Discriminative features are not resilient in adversarial settings



Generating adversarial example helps build better classifiers

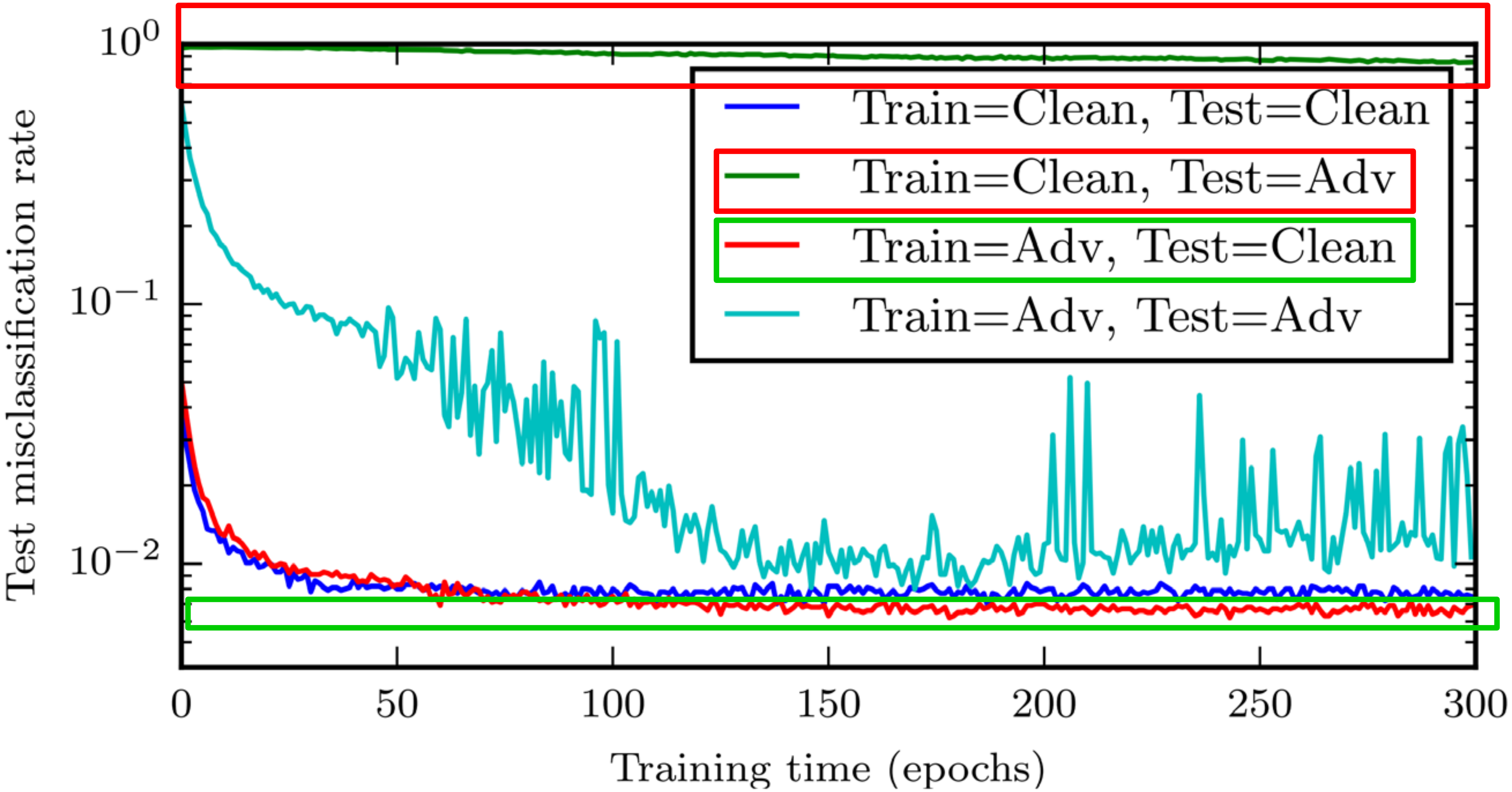


Figure Credit: GoodFellow 2016

Not all evasive samples are good adversarial testing inputs

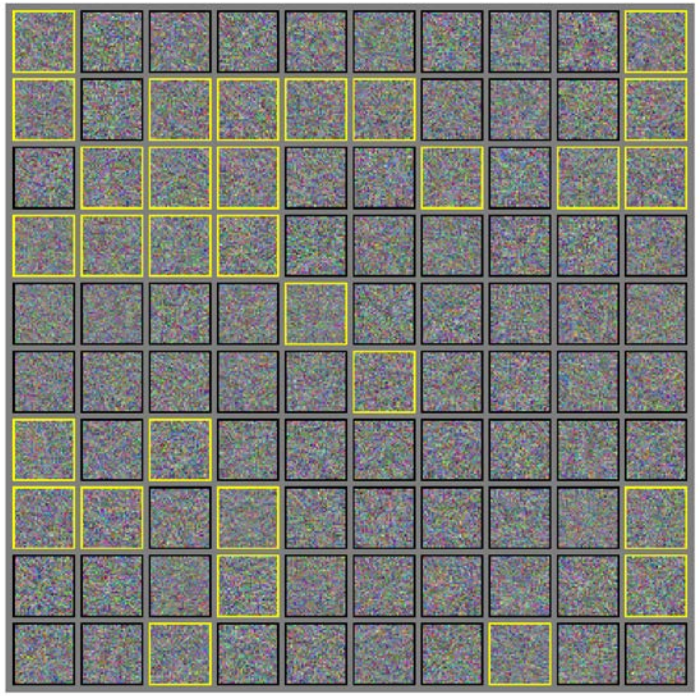


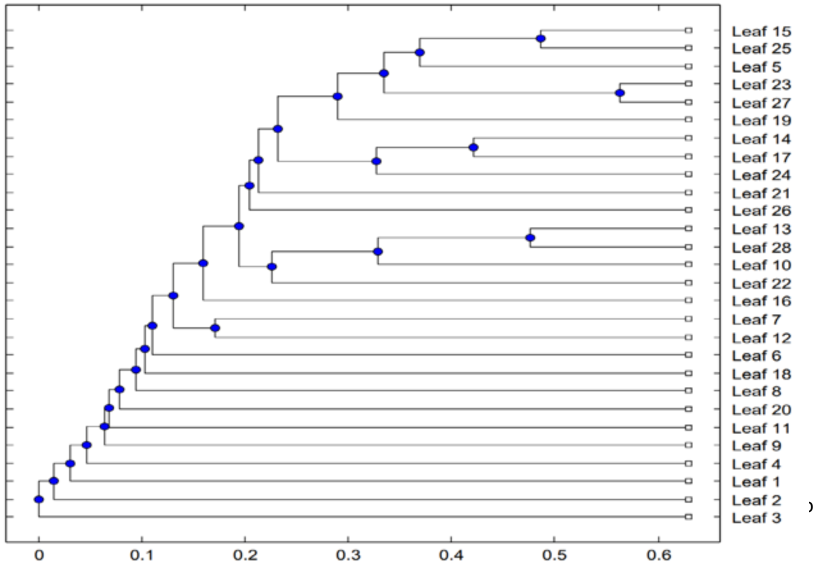
Figure Credit: GoodFellow 2016

- Potential side effect
  - –crash the app
  - –cause undesirable behaviors
  - –disable malicious functionalities.
  - –the code cannot even be compiled.

- Automatically generating meaningful adversarial malware is challenging!

Malware Recomposition Variation (MRV)

- Malware Evolution Strategy
  - Phylogenetic analysis
- Malware Confusion Strategy
  - Similarity metric
- Insight
  - Follow existing patterns!

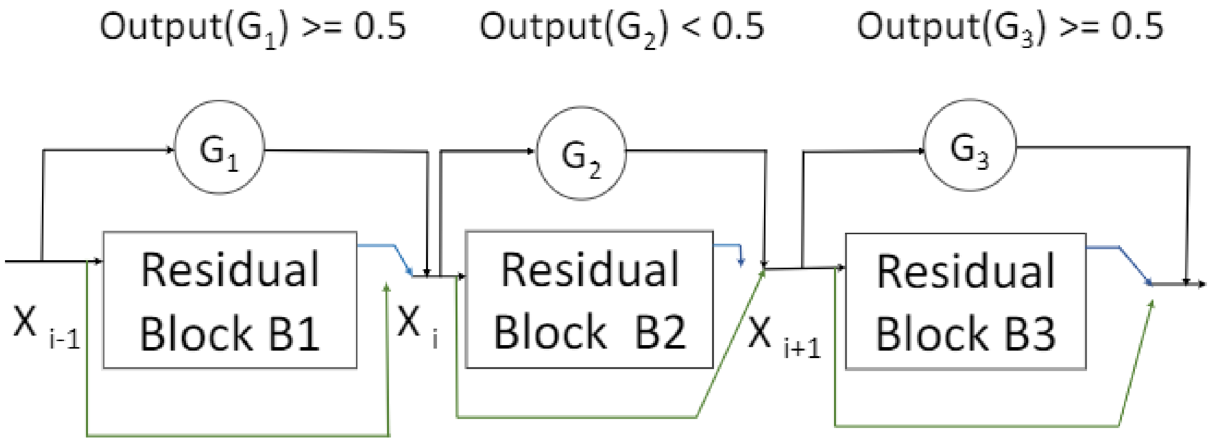


1: 5e2c1f35ea3196a7d81b42d932729c4f253fbc8a -> DroidKungFu.AB.Gen
2: b77e28f4018dfb1e73e83a617db9f36a708ccfae -> DroidKungFu.AB.Gen
3: ba92a5bbb79dace47a76455754865d8fab9ab2cc -> DroidKungFu.AB.Gen
4: c3d37de639c0909ad78cec8c52f63f04742fbe6b -> DroidKungFu.AB.Gen
5: a997762fd1edc5971071ec574e6e8c4e -> DroidKungFu.AW
6: 1e036ab0c29dd1c8d8b95bdd2eb3400d -> DroidKungFu.AW
7: 1e7203279f153c3282eeecadbf2a9b232bc4ffda -> DroidKungFu.AW
8: f1e4a265c516b104d4e2340483fca24d60be4c08 -> DroidKungFu.AW
9: 267043ab471cf7a7d82dda6f16fa4505f719f187 -> DroidKungFu.AW
10: 683e1f3e67b43631690d0fec49cec284 -> DroidKungFu.BB
11: 517dc7b67c852392491ce20baff70ada92496e6d -> DroidKungFu.BL
12: 044f87b435c583ca4aa116aef4b54e09bdb42c7b -> DroidKungFu.BL
13: b9eae89df96d9d62ed28504ee01e868b -> DroidKungFu.C
14: 4d56a6705afa9b162f652303f6c16741 -> DroidKungFu.C
15: f30c4058f1e6cb46f81d21b263cf35454638275a -> DroidKungFu.C
16: 6d3b9cdf559443db567113585b40eef459307c94 -> DroidKungFu.G
17: 537db43883f55b112a4206fb99093bbb31c9c3f2 -> DroidKungFu.G
18: 4c1775c28def41a221e5bf872c5e593de22bb9a6 -> DroidKungFu.G
19: 92888228c556f94b8be3d8bf747a2427481f32d1 -> DroidKungFu.G
20: 0657a70a655dc439b4222c8161b1f5a9667e84e3 -> DroidKungFu.G
21: 8c841b25102569be4a1a5f407108482473fad43e -> DroidKungFu.G
22: e3f0de8ad898f0b5a7c9d327ffc266635b8af32b -> DroidKungFu.G
23: 03ddb783e7ab88c838b1888b38eba992 -> DroidKungFu.M.Gen
24: 00b89bcb196a138f9f20a6853c41673a18a2575f -> DroidKungFu.M.Gen
25: 584f8a2801a8e7590dc466a6ebc58ea01a2d1758 -> DroidKungFu.M.Gen
26: 8edb188204c6ad79006e555b50e63705b68ea65d -> DroidKungFu.M.Gen
27: bd249c0843df2f8acfe7615feba505ead30e5bbc -> DroidKungFu.M.Gen
28: 2cfa26bb22bbdc4e310728736328bde16a69d6b4 -> DroidKungFu.M.Gen

Outline

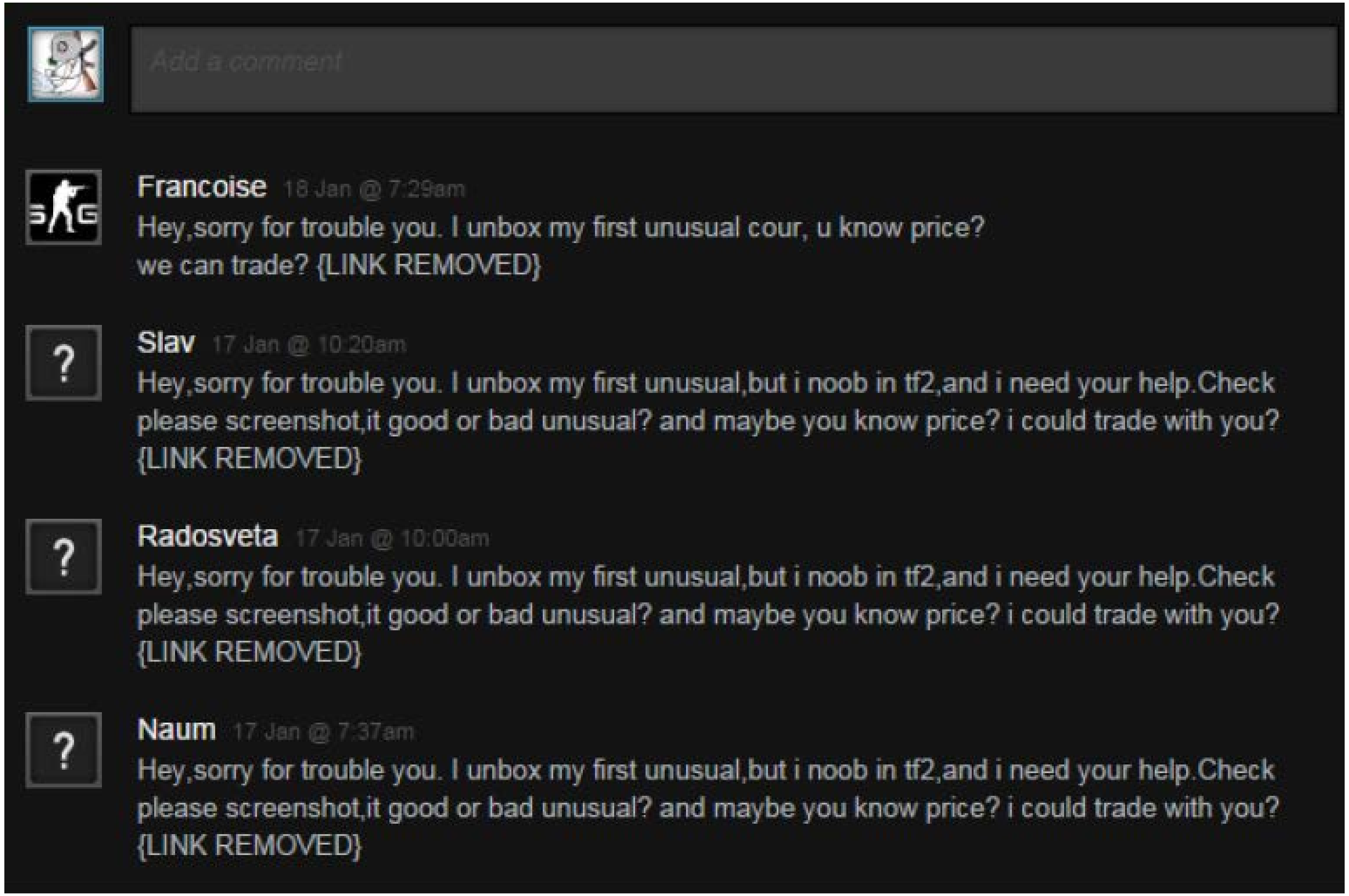
- Application of learning-based techniques
- Quality assurance of learning-based techniques
- Security of learning-based techniques
  - Privacy of DNN
  - Attacking Malware Detector
  - Other
- Interpretation of learning-based security techniques

Adversarial Energy Attack

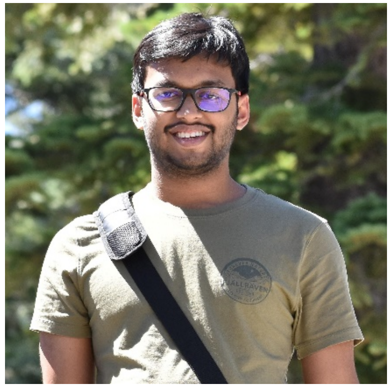


Original | Gaussian Noise | Static | Borderline | Modified L2

Adversarial NLP



Generating malicious messages that can avoid bot detection.



Outline

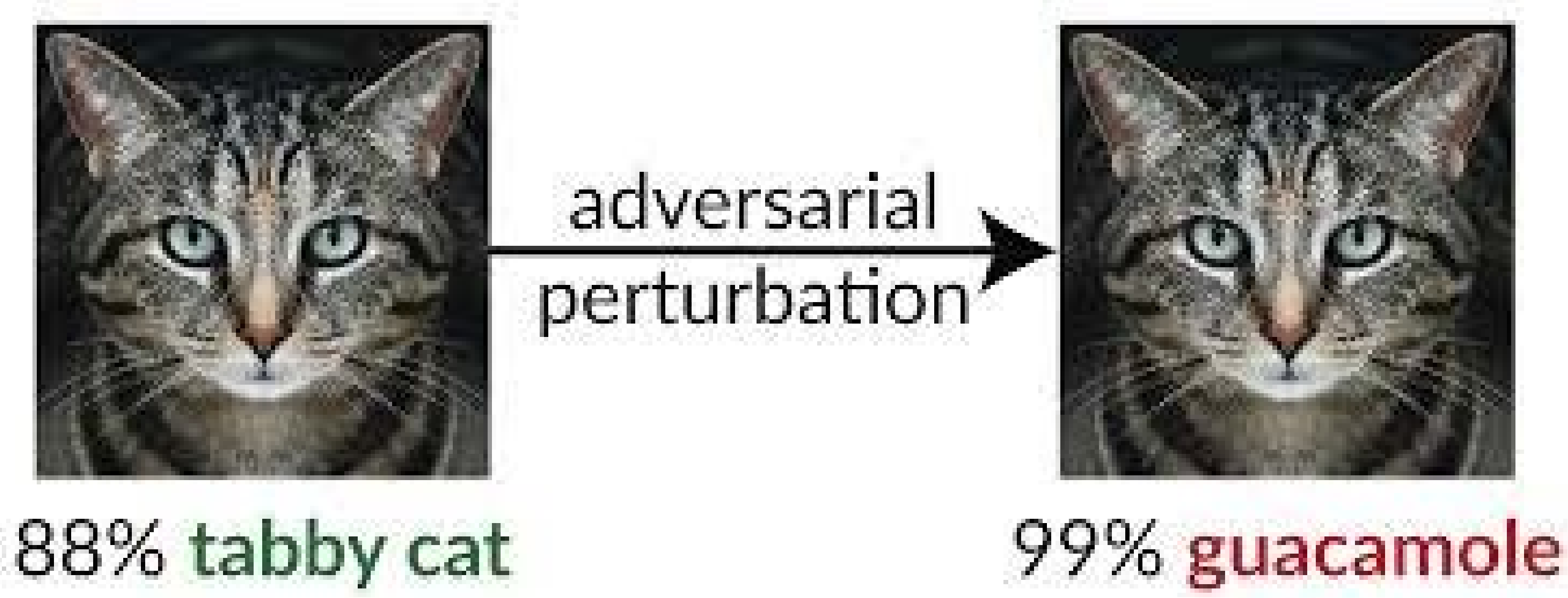
- Application of learning-based techniques
- Quality assurance of learning-based techniques
- Security of learning-based techniques
- Interpretation of learning-based security techniques

DENAS: Input-independent Interpretation for Security-oriented Neural Networks



Concerns of Learning-based Techniques

- Deep Learning give the prediction result without reasons.
- Unlikely to discover the biases in the dataset.
    - the uneven distribution of the dataset.
- Difficult to know why the model makes mistakes and fix the error.



Explaining Machine Learning Models

- Human-interpretable Model
  - Domain experts could validate the model through his domain knowledge.

- Rule-Based Inference

Part of speech tagging

Instance	If	Predict
I want to play(V) ball.	previous word is PARTICLE	play is VERB.
I went to a play(N) yesterday.	previous word is DETERMINER	play is NOUN.
I play(V) ball on Mondays.	previous word is PRONOUN	play is VERB.

How to classify Apple and Banana

If (color == red) and (shape == round)
Then this is Apple

Explaining Machine Learning Models

- Definition of Rule-Based Inference
  - A rule is a IF-THEN statement. IF is the condition, THEN is the prediction.
  - Condition is a collection of tuples $r = \{\langle pos_i, val_i \rangle | \quad i = 1, 2, \cdots \|r\|\}$

An input **h** satisfy the rule condition: $\forall i = 1, 2, \cdots, \|r\| \qquad h[pos_i] = val_i$

- Prediction is sufficient conditional inference $[r_j(x) = 1] \Rightarrow [C(x) = 1]$

Hex Sequence

0x90	0x90	<u>0xb8</u>	0xa7	0x87	0x71	0x00

Decimal Sequence

144	144	<u>**184**</u>	167	135	113	00

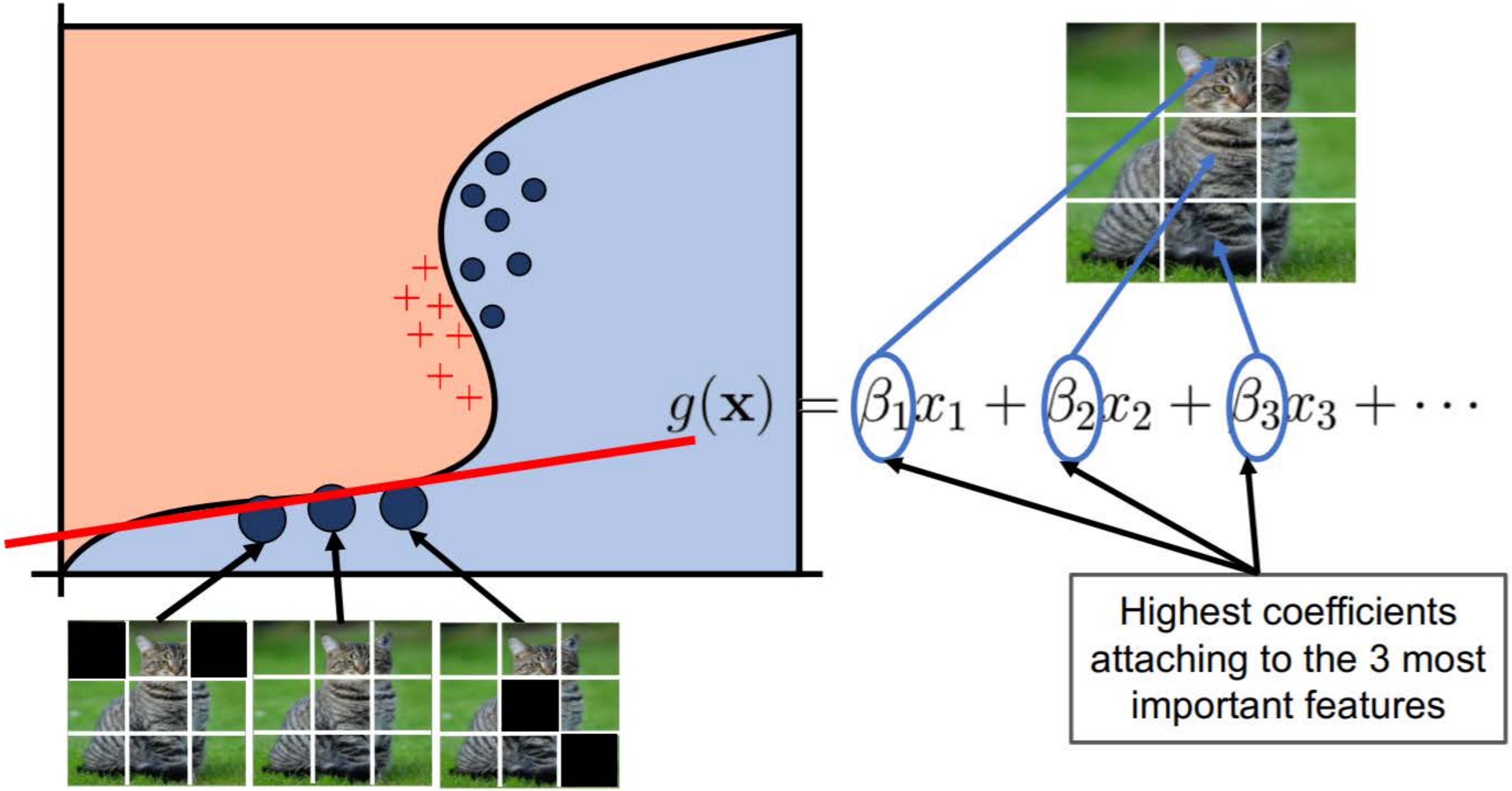
x[1] = 144 and x[2] = 184 and x[3] = 167

184 is the function start

Existing Techniques to Derive Explanations

- Categorized into two kinds
  - Local Explanation: explain the model's prediction result for one input.
    - **LEMNA, LIME, Grad-CAM**
  - Global Explanation: acquire the knowledge learned by the model.
    - **Tree Regression**
- Both are post-hoc explanations. Explanations are given after prediction.
- Existing interpretation methods are based on input data.

One Example of Model Explanation (LEMNA, CCS2018)



Picture from Guo. [LEMNA: Explaining Deep Learning based Security Applications]

Limitation of the Existing Techniques (LEMNA, CCS2018)

- Some Features are just Symbols with no Numerical Meaning.
  - 0x89 0xd1 mov ecx,edx
  - 0x90 nop
  - 89 and 90 have completely different meanings.
- Where to give Explanation.
  - LEMNA give explanation case by case, It is impossible explain every input data. (Cost)

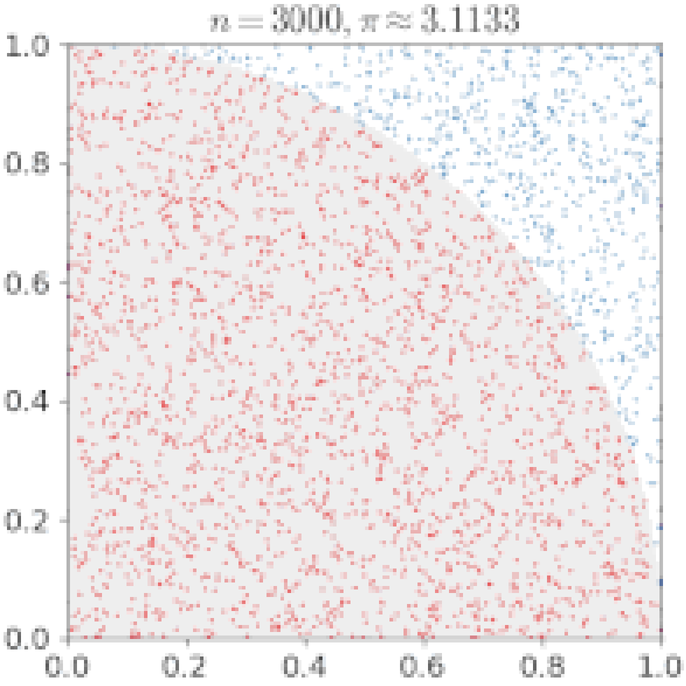
The Challenge of Input-independent Explanation

- How to model the complex nonlinear decision boundary.
  - The decision boundary of a neural network is manifested through the nonlinear function mapping the input to the output of the neural network.
  - The nonlinearity results in great complexity to interpret the decision making of the neural network in a human comprehensible manner, making it almost impossible to solve the function.
- Our Approach
  - We propose neuron activation probability to approximate the nonlinear constraints of the decision boundary.
  - Calculating the neural activation probability through Monte Carlo method.

Conditional Activation Probability

How to decide the activation Matrix A beforehand

Monte Carlo(MC) method

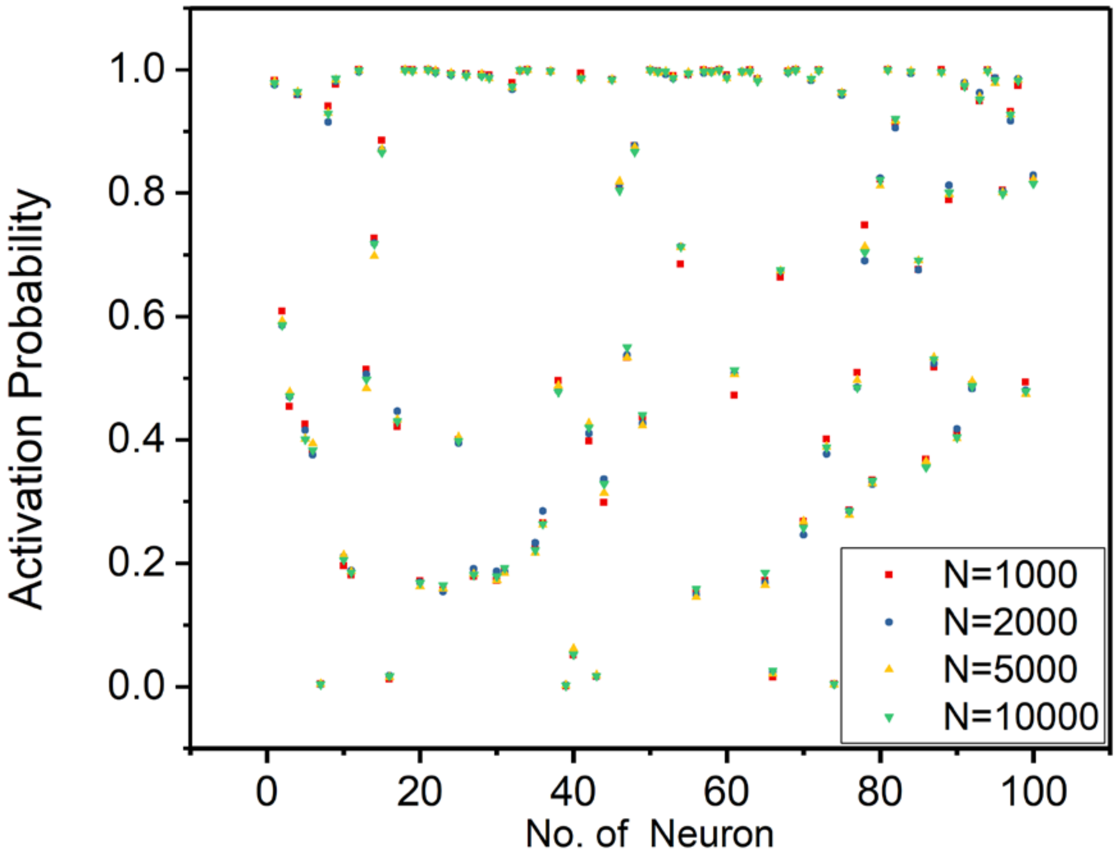


$$P_i^r = \begin{bmatrix} p_{i,1}^r & \cdots & 0 \\ 0 & p_{i,2}^r & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & p_{i,s_i}^r \end{bmatrix}$$

$P_{i,j}$ is the probability of jth neuron in the ith layer being activated

$$P_i^r \simeq A_i(\boldsymbol{v}_0^r)$$

Sensitivity



Experiment results of sensitivity

In our experiments, we set **N** as 1000 in the MC method, and we test whether it would affect the result.

Results showed when **N** > 1000, the activation probability is converged.

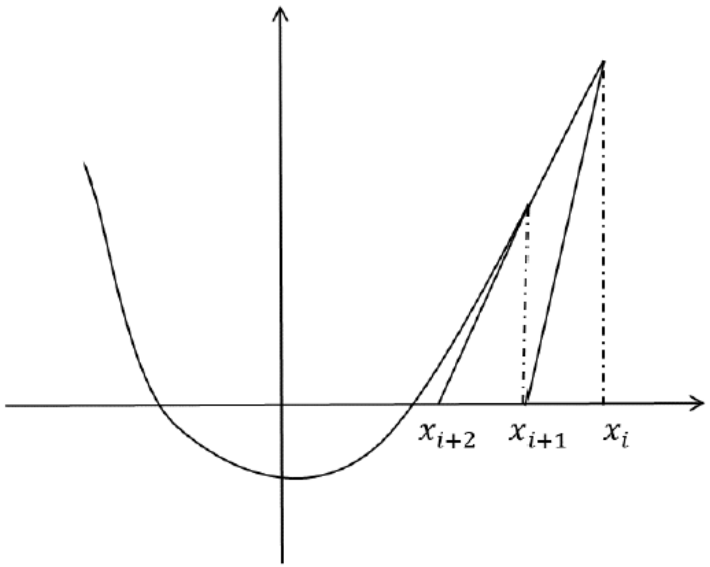
The Challenge of Input-independent Explanation

- How to extract the general rules to represent the model's behavior.
  - One of the key attribute for an input-independent explanation approach is the ability to present saliency (e.g., bias, preference) in the decision making of neural networks.
  - Extracting *most representative rules* is challenging because there are a large (if not infinite) number of rule candidates given high-dimension input data with many possible values for each feature.
- Our approach
  - With the help of neural activation probability, we could linearize the decision boundary then select the feature value contribution most to the prediction result.
  - We propose an iterative approach inspired by Newton-Raphson method to approximate the activation probability step by step.

General Rule Generation

$$\mathcal{F}(x) = D \cdot (P_L^r(W_L(P_{L-1}^r(W_{L-1}(\cdots P_0^r(W_0 x_0 + b_0)) + b_{L-1})) + b_L))$$

With the help of Matrix, we could Linearize the decision boundary. compute the contribution toward final output separately for each feature value.



(1) Initialize an empty rule r
(2) Estimating the activation probability of neurons under current rule r using Monte Carlo method
(3) Select the feature value p make the most contribution to the objective function.
(4) Update the rule $r = r \cup p$
(5) Identify the rule, if the rule do not have enough confidence to predict the behavior of the model, go to step2.

Main Insight

(3) Select the feature value p make the most contribution to the objective function.

- The main insight enabling input-independent explanation of DENAS is that unlike image recognition, security applications feed large amount of discrete data to neural networks.

- DENAS thus leverages the fact that discrete data is *enumerable* to reduce the complexity of computing the decision boundary of neural networks.

The Challenge of Input-independent Explanation

- How to use the model to extract rules under a given data distribution.
  - The input space of a neural network used in security domain is usually non-continuous and irregular. As an input-independent approach, DENAS can provide explanation for all behaviors of a neural network, including the behaviors on invalid or unrealistic inputs (i.e., data outside valid input space).
  - Security analysts may not be interested in explanation of behaviors on invalid inputs.
    - For example, security analysts may not care how a malware classifier makes decisions on a program sample that cannot be compiled or executed.
- Our approach
  - We introduce two kinds of domain specific knowledge as constraints to reject illegal rules.
  - Static knowledge
  - Extensible knowledge

Domain-Specific Rule Generation

Static Domain Knowledge Constraints (Bayesian Statistics)

Table 1: Example of Bayesian Statistics

Hex Sequence	Instruction
0xc7,0x45,0xbc,0x00,0x00,0x00,0x00	movl,$0x0,-0x44(%rbp)
0xc7,0x45,0xfc,0x00,0x00,0x00,0x00	movl,$0x0,-0x4(%rbp)
0xc7,0x45,0xf0,0x00,0x00,0x00,0x00	movl,$0x0,-0x10(%rbp)
0xc7,0x05,0x84,0xa3,0x31,0x00,0x00	movl,$0x0,-0x31a384(%rbp)

$$P(h_2 = \mathrm{0x45} \mid h_1 = \mathrm{0xc7}) = \frac{3}{4} = 0.75$$

$$P(h_3 = \mathrm{0xbc} \mid h_2 = \mathrm{0x05}) = \frac{0}{1} = 0.00$$

Extensible Domain Knowledge Constraints (Markov Chain)

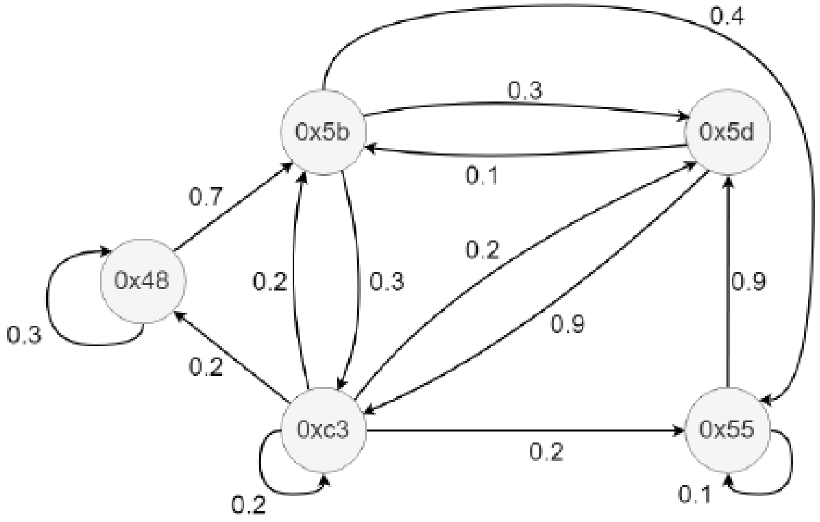


Figure 3: Example of Markov Chain

Why DENAS is useful...

Demonstration of DENAS in Identifying Binary Function Start

1. Summarize the most general rules with the help of DNN.
2. Discovering new knowledge not existing in the visible data.
3. Find the Bias of the model, the uneven distribution.
4. Troubleshooting beforehand and patching model errors.

Demonstration of DENAS in Identifying Binary Function Start

- Summarize the most general rules

ID	F.Start	Binary Code						
1	0x56	0x56	0x57	0x53	0x83	0xec	0x70	
2	0x55	0x90	0x55	0x89	0xe5			
3	0x55	0xc3	0x55	0x89	0xe5	0x53	0xec	0x04
4	0x83	0xc3	0x83	0xec	0x14			
5	0x53	0xc3	0x53	0x83	0xec	0x24		

ByteWeight summarize 1208767 assembly signatures as the function start, We use 1000 binary signatures and could cover more than 80% of the dataset.

- Discovering new knowledge not existing in the visible data
  - Start of utility function and preparations at the function start

0x56	0xc3	0x56	0x56	0x56						ret; push esi; push esi; push esi;
0x55	0x90	0x55	0x57	0x54						nop; push ebp; push edi; push esp;
0x53	0xc3	0x90	0x53	0x56						ret; nop; push ebx; push esi;

Demonstration of DENAS in Identifying Binary Function Start

- Find areas Bias of the model

Top-5 coverage rules in the data set

Assembling Code	Coverage
push esi; push edi; push ebx; sub 0x70, esp;	6.6%
nop; push ebp; mov esp, ebp;	4.4%
ret; push ebp; mov esp,ebp; push ebx; sub 0x4,esp;	3.0%
ret; sub 0x14, esp;	2.2%
ret; push ebx; sub 0x24, esp;	0.7%

Demonstration of DENAS in Identifying Binary Function Start

- Troubleshooting beforehand and patching model errors
  - Indicators for function start appear in the middle of a function.

0x56	0x1a	0x56	0x53	0x81	0xec	0x9c	0x00	0x00	0x00	sbb dl,BYTE PTR[esi+0x53]; sub esp,0x9c
0x83	0x90	0x83	0x7d	0xec	0x00	0x0f				nop; cmp DWORD PTR[ebp-0x14],0x00;
0x55	0x89	0x55	0xec	0x8b	0x45	0xf4				mov DWORD PTR[ebp-0x14],edx; mov eax,DWORD PTR [ebp-0xc];
0x55	0x8b	0x55	0xec	0x89	0x01					mov edx,DWORD PTR[ebp-0x14]; mov DWORD PTR[ecx],eax;
0x83	0xe8	0x57	0xc3	0x83	0xec	0x1c	0xc7			call 0xec83c35e; sbb al,0xc7;
0x55	0x56	0x57	0x53	0x55	0x83	0xec	0x7c			push esi; push edi; push ebx; push ebp; sub esp,0x7c;

"[0x55, 0x83, 0xec]" according to the instruction "[push ebp; sub esp,0x7c;]", ebp register is for a stack frame and "push ebp" is often located at the start of the function, and "[0x83, 0xec]" represents the "sub esp" instruction are used to space allocated on the stack for the local variables. Which are typical appear at the function start.

Demonstration of DENAS in Identifying Binary Function Start

- Patching method: correcting a specific error testing sample

Table 2: : Classification accuracy of the trained classifiers. "P" is precision and "R" is recall, "A" is accuracy

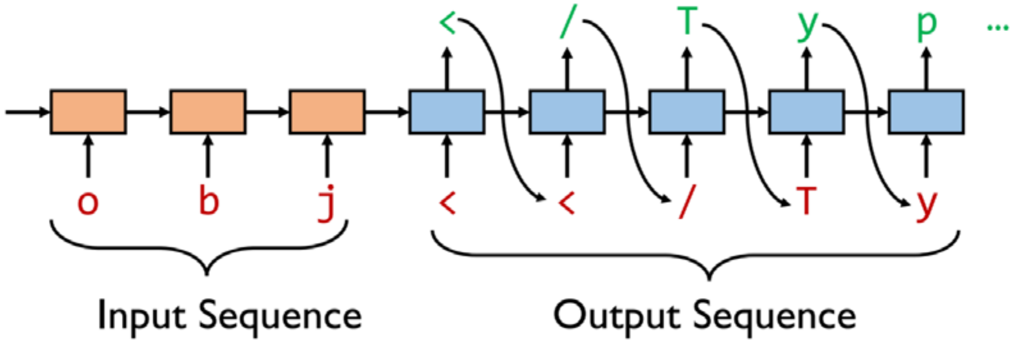
Metric	Before Patch	kp=5	kp=10	kp=20	kp=100
P	95.13%	99.08%	96.45%	96.51%	97.00%
R	95.90%	89.62%	97.46%	97.47%	97.00%
F1	95.52%	94.11%	96.95%	96.99%	97.00%
A	99.97%	99.96%	99.98%	99.98%	99.98%

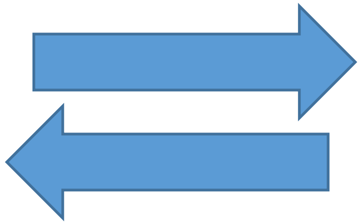
Table 5: The Consistency of the Patched Rule Before and After Debugging

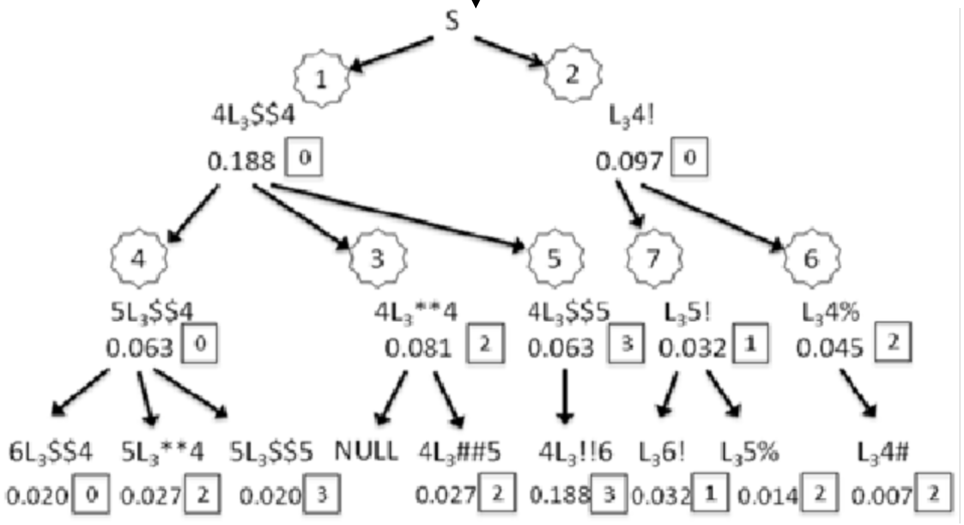
No.	Before Debugging	kp=5	kp=10	kp=20	kp=100
1	83.68%	51.78%	73.30%	72.16%	51.94%
2	90.16%	64.98%	85.36%	84.54%	62.80%
3	92.06%	72.32%	88.62%	88.88%	70.00%
4	88.02%	62.28%	83.60%	82.30%	56.18%
5	90.66%	67.56%	85.54%	83.48%	68.30%

Future Work

The target program







RNN in Learn&Fuzz

PCFG in REINAM

Questions?