Quality Assurance and Security in the Age of Machine Learning

CS 4301

Wei Yang

Department of Computer Science
University of Texas at Dallas
• Application of learning-based techniques

• Quality assurance of learning-based techniques

• Security of learning-based techniques

• Interpretation of learning-based security techniques
• 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

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

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

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

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

Botnet activity
- Launching DDoS attacks
- Click fraud
- Sending premium rate SMS messages
The number of mobile malware keeps increasing.

Report: In 2017, up 754,958 from Q1. In just 3 months, there are now over 3.5 million new Android malware samples, up from 214,327 in 2012. Source: McAfee Labs
Opportunities and challenges in mobile analysis

- Automated techniques have been introduced
Opportunities and challenges in mobile analysis

• Automated techniques have been introduced

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
Malware Detection

AppContext: Differentiating Malicious and Benign Mobile App Behavior using Contexts
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

12
A benign app —— Greetings

Greetings

Wishing you happiness
To welcome each morning,
Wishing you laughter
To make your heart sing.
Wishing you friendship
Sharing and caring,
And all of the joy
The birthday can bring!

Amazing that you were once a helpless little child, but now you a giant helpless person! Have fun blowing out all the candles on your cake!

On this special day of your wedding
wishing you happiness and love
like the unending circle of your wedding ring.

Send
Copy
Remove
Clear list
An adaptive adversary

Permissions

Sensitive API methods

(sendTextMessage(String destinationAddress, String scAddress, String text)
Send a text based SMS.

Your messages
Edit your text messages (SMS or MMS), read your text messages (SMS or MMS), receive text messages (SMS), send SMS messages

Your messages
Edit your text messages (SMS or MMS), read your text messages (SMS or MMS), receive text messages (SMS), send SMS messages

sendTextMessage(String destinationAddress, String scAddress, String text)
Send a text based SMS.
A research methodology

- What intrinsic property makes malware and benign apps different?

**Problem Insights**

- What is the representation of such difference in the mobile programs?

**Program Characteristics**

- How to automatically extract such representation from mobile programs?

**Analysis Techniques**
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)
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.
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.

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

Balance!!!
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**
How to extract contexts automatically?

Control Flow Graph
- Construct & Traverse RICFGs
- Conditional Statements
- Extract Context Factors

Call Graph
- Identify Activation Events
- Activation Events

Locate Security-Sensitive Behaviors

Security-Sensitive Methods

Context Factors

Contexts

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

ECG: Extended Call Graph;
RICFG: Reduced Inter-procedure Control Flow Graph
How to extract contexts automatically?

**Call Graph**

- **Method Invocation**
  - `OnCreate()`
  - `b()`
  - `sendTextMessage()`

**Control-flow Graph**

- **Dependency**
  - `Date date = new Date();`
  - `if(date.getHours>23 || date.getHours<5)`
  - `sendTextMessage()`

**Security Insights**

**Program Characteristics**

**Analysis Techniques**
Example

**Unexpected Context**

```java
ActionReceiver.OnReceive()
Date date = new Date();
if (data.getHours > 23 || date.getHours < 5)
    ContextWrapper.StartService(MainService);
...
ContextWrapper.StartService()
```

**Expected Context**

```java
DummyMainMethod()
MainService.OnCreate()
SendTextActivity$4.onClick()
SendTextActivity$5.run()
MainService.b()
SmsManager.sendTextMessage()
ContextWrapper.StartService()
```

**Security Insights**

**Program Characteristics**

**Analysis Techniques**
The app will send an SMS when
• user clicks a button in the app
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

**Context 1:**
- **Event:** Signal strength changes,
- **Factor:** Calendar

**Context 2:**
- **Event:** Clicking a button

### Existing Features

<table>
<thead>
<tr>
<th>Permission</th>
<th>Method</th>
<th>...</th>
<th>Hardware</th>
<th>System</th>
<th>UI</th>
<th>F₁</th>
<th>F₂</th>
<th>F₃</th>
<th>F₄</th>
<th>F₅</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEND_SMS</td>
<td>SendTextMessage</td>
<td></td>
<td>N/A</td>
<td>SIG_STR</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SEND_SMS</td>
<td>SendTextMessage</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>Click</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**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
People were asked to read aloud the terms and conditions for popular apps and were shocked by what they actually agreed to.
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 $\leftrightarrow$ a permission of an app
Also you can share the yoga exercise to your friends via Email and SMS.
Social Network for Mobile Security

MalScan: Fast Market-Wide Mobile Malware Scanning by Social-Network Centrality Analysis
• 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 2\(^{nd}\)Q
  • 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

<table>
<thead>
<tr>
<th>WeChat</th>
</tr>
</thead>
<tbody>
<tr>
<td># of executable Java code lines:</td>
</tr>
<tr>
<td># of Java classes:</td>
</tr>
<tr>
<td># of Android activities:</td>
</tr>
<tr>
<td># of C or C++ code lines:</td>
</tr>
</tbody>
</table>
Now— UI testing agent with reinforcement learning

State $S_t$ → Brain $C$ → Action $a_t$ → Reward $r_t$ → Neural network

Traditional program (control flow graph) vs. Neural network:
- Traditional program has a control flow graph.
- Neural network inputs program logic and outputs covered or not covered.

$x=0$
If ($x==8$)
$x+=1$
$x+=2$

$x+=$

$S_t$ $a_t$ $r_t$
Cooperative Mobile Testing
REINAM: Reinforcement Learning for Input-Grammar Inference.
Motivation

- Many programs take input strings that form a grammar.

<table>
<thead>
<tr>
<th>Program</th>
<th>Valid Input String</th>
<th>Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td>URIdecoder</td>
<td><a href="https://google.com">https://google.com</a></td>
<td>URI = scheme:[]/authority]path[?query][#fragment]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>authority = [userinfo@]host[:port]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>......</td>
</tr>
</tbody>
</table>

- Knowing the grammar helps us understand the input structure.
• Input grammar could be useful in a wide range of applications:

Fuzz Testing

Reverse Engineering
Existing Approach

- Target Program
- Seed Input

Program:
- Unanalyzable Code

Language:
- Low-quality
- Low-variety
- Seed Inputs

Machine:
- Lack of Seed Inputs

Synthesized CFG
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.

[1] Osbert Bastani, Rahul Sharma, Alex Aiken, and Percy Liang. Synthesizing program input grammars in PLDI’17
• **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.
• **Phase 2**: REINAM converts the CFG from Phase 1 to a PCFG, and then uses reinforcement learning to refine this PCFG.

• 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

1. **Phase 1**
   - **Target Program**
     - **Symbolic Execution Engine**
       - **Seed Inputs**
         - **Language Inference Algorithm**
           - **Initial CFG**
             - **Initial PCFG**
               - **Generalization**
                 - **Refined PCFG**
                   - **Probability Adjustment**
                     - **Reward Calculation**
                       - **Input Sampling**
                         - **Mutated PCFG**
                           - **Result CFG**
Generalizing PCFG via Reinforcement Learning

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

**Reinforcement Learning for Maze Solving [2]**

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
**Problem Statement**

Generating Regular Expressions from NL

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**

<table>
<thead>
<tr>
<th>Domain</th>
<th>NL Specification</th>
<th>Program 1</th>
<th>Program 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regex</td>
<td>Match lines that start with an uppercase vowel and end with ‘X’</td>
<td><code>([AEIOUaeiou][A-Z]).*X</code></td>
<td><code>([AEIOU].*)&amp;(.X)</code></td>
</tr>
<tr>
<td>Bash</td>
<td>Rename file ‘fl’ to ‘fl.txt’</td>
<td><code>mv 'fl' 'fl.txt'</code></td>
<td><code>cp 'fl' 'fl.txt'; rm 'fl'</code></td>
</tr>
<tr>
<td>Python</td>
<td>Assign the greater value of ‘a’ and ‘b’ to variable ‘c’</td>
<td><code>c = a if a &gt; b else b</code></td>
<td><code>c = [b, a][a &gt; b]</code></td>
</tr>
</tbody>
</table>

A semantically equivalent program may have various syntactically different forms.

---

**SemRegex:** A Semantics-Based Approach for Generating Regular Expressions from Natural Language Specifications  
Zhong et al.  
EMNLP 2018
# Cross-language Vulnerability Detection

<table>
<thead>
<tr>
<th>SARD dataset</th>
<th>CWEs</th>
<th>#</th>
<th>CWE</th>
<th>Languages</th>
<th>Fixes similar</th>
<th>Good cases in Test Suit (Java/C/C++)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>112</td>
<td>1</td>
<td>CWE: 15 External Control of System or Configuration Setting</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Java,C</td>
<td>yes</td>
</tr>
<tr>
<td>C++</td>
<td>118</td>
<td>2</td>
<td>CWE: 23 Relative Path Traversal</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td>Common</td>
<td>56</td>
<td>3</td>
<td>CWE: 36 Absolute Path Traversal</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>CWE: 78 OS Command Injection</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>CWE: 90 LDAP Injection</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>CWE: 114 Process Control</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>CWE: 134 Uncontrolled Format String</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Java,C</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>CWE: 190 Integer Overflow</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>CWE: 191 Integer Underflow</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Java,C</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>CWE: 197 Numeric Truncation Error</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Java,C</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>CWE: 252 Unchecked Return Value</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>CWE: 253 Incorrect Check of Function Return Value</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CWE: 256 Plaintext Storage of a Password.</td>
<td></td>
<td>Read the password from a Properties file or a regular file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In the good case, read the file from the console.</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>CWE: 259 Hard Coded Password</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>CWE-319: Cleartext Transmission of Sensitive Information</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>CWE-325: Missing Required Cryptographic Step</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>CWE-327: Use of a Broken or Risky Cryptographic Algorithm</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>CWE-328: Reversible One-Way Hash</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Independent</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>CWE-369: Divide By Zero</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Java,C,C#</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22</td>
<td>CWE-396: Declaration of Catch for Generic Exception</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Java,C,C#</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23</td>
<td>CWE-397: Declaration of Throws for Generic Exception</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Java,C,C#</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>CWE-398 Indicator of Poor Code Quality</td>
<td><a href="https://cwe.mitre.org/">https://cwe.mitre.org/</a></td>
<td>Java,C</td>
<td>yes</td>
</tr>
</tbody>
</table>
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

Traditional program (control flow graph)

If (x==8)

x+=1

x+=2

Program Logic

Input

Covered

Not Covered

Output

Neural network
Automated Generation of Test Oracle for Deep Learning Application

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

Adversarial Machine Learning

(Eykholt et al, 2017)
• 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
Property Inference Attacks on Deep Neural Networks using Permutation Invariant Representations
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

Yes!

Is Smiling?

No!

Neural Network

Smile Detector
An example: A simple property of the training dataset
An example: A simple property of the training dataset

\[ P: \text{Skewed towards attractive people} \]
\[ \bar{P}: \text{Only ordinary people} \]
An example: A simple property of the training dataset

- Skewed towards attractive people
- Only ordinary people

Neural Network
Smile Detector

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

\[ P \]
- Shadow Training Set 1
- \( \vdots \)
- \( \vdots \)
- \( \vdots \)
- Train
- Shadow Classifier 1
- \( \vdots \)
- \( \vdots \)

\[ \bar{P} \]
- Shadow Training Set \( k \)
- \( \vdots \)
- \( \vdots \)
- \( \vdots \)
- Train
- Shadow Classifier \( k \)
- \( \vdots \)
- \( \vdots \)

\[ (F_1, P) \]
- \( \vdots \)
- \( \vdots \)
- \( \vdots \)
- \( (F_k, \bar{P}) \)

Train
Feature Extraction

Target Classifier

Feature Extraction

Meta-Classifier

\[ P / \bar{P} \]
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

Property Inference

[Tahir et al. RAID’17]

Patched or Not?
• 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
Malware Detection in Adversarial Settings: Exploiting Feature Evolutions and Confusions in Android Apps
Why not using everything in malware bytecode as features?

- Discriminative features are not resilient in adversarial settings.

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

```java
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);
Why not using everything in malware bytecode as features?

• Discriminative features are not resilient in adversarial settings

Unmatched!
Generating adversarial example helps build better classifiers

Figure Credit: GoodFellow 2016
Not all evasive samples are good adversarial testing inputs

• 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!
• 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

Output($G_1$) $\geq$ 0.5  Output($G_2$) $<$ 0.5  Output($G_3$) $\geq$ 0.5

$G_1$  $G_2$  $G_3$

Residual Block B1  Residual Block B2  Residual Block B3

Original  Gaussian Noise  Static  Borderline  Modified L2
Generating malicious messages that can avoid bot detection.
• 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

<table>
<thead>
<tr>
<th>Instance</th>
<th>If</th>
<th>Predict</th>
</tr>
</thead>
<tbody>
<tr>
<td>I want to play(V) ball.</td>
<td>previous word is PARTICLE</td>
<td>play is VERB.</td>
</tr>
<tr>
<td>I went to a play(N) yesterday.</td>
<td>previous word is DETERMINER</td>
<td>play is NOUN.</td>
</tr>
<tr>
<td>I play(V) ball on Mondays.</td>
<td>previous word is PRONOUN</td>
<td>play is VERB.</td>
</tr>
</tbody>
</table>

Part of speech tagging

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
    \[ r = \{ (pos_i, val_i) \} \quad i = 1, 2, \ldots \| r \| \]
    An input \( h \) satisfy the rule condition:
    \[ \forall i = 1, 2, \ldots , \| r \|, \quad h[pos_i] = val_i \]
  • Prediction is sufficient conditional inference
    \[ [r_j(x) = 1] \implies [C(x) = 1] \]

<table>
<thead>
<tr>
<th>Hex Sequence</th>
<th>0x90</th>
<th>0x90</th>
<th>0xb8</th>
<th>0xa7</th>
<th>0x87</th>
<th>0x71</th>
<th>0x00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal Sequence</td>
<td>144</td>
<td>144</td>
<td>184</td>
<td>167</td>
<td>135</td>
<td>113</td>
<td>00</td>
</tr>
</tbody>
</table>

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](http://example.com)
• 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

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

$p_{i,j}$ is the probability of $j$th neuron in the $i$th layer being activated

$P_i^r \approx A_i(v_i^r)$
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 attributes 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.
With the help of Matrix, we could Linearize the decision boundary. compute the contribution toward final output separately for each feature value.

\[ F(x) = D \cdot (P_L^r(W_L(P_{L-1}^r(W_{L-1}(\cdots P_0^r(W_0x_0 + b_0)) + b_{L-1}))) + b_L)) \]

(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 step 2.
(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

<table>
<thead>
<tr>
<th>Hex Sequence</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xc7,0x45,0xbc,0x00,0x00,0x00,0x00</td>
<td>movl,$0x0,-0x44(%rbp)</td>
</tr>
<tr>
<td>0xc7,0x45,0xfc,0x00,0x00,0x00,0x00</td>
<td>movl,$0x0,-0x4(%rbp)</td>
</tr>
<tr>
<td>0xc7,0x45,0xf0,0x00,0x00,0x00,0x00</td>
<td>movl,$0x0,-0x10(%rbp)</td>
</tr>
<tr>
<td>0xc7,0x05,0x84,0xa3,0x31,0x00,0x00</td>
<td>movl,$0x0,-0x31a384(%rbp)</td>
</tr>
</tbody>
</table>

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

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

Extensible Domain Knowledge Constraints
(Markov Chain)

Figure 3: Example of Markov Chain
Why DENAS is useful...
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

<table>
<thead>
<tr>
<th>ID</th>
<th>F.Start</th>
<th>Binary Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x56</td>
<td>0x56 0x57 0x53 0x83 0xec 0x70</td>
</tr>
<tr>
<td>2</td>
<td>0x55</td>
<td>0x90 0x55 0x89 0xe5</td>
</tr>
<tr>
<td>3</td>
<td>0x55</td>
<td>0xc3 0x55 0x89 0xe5 0x53 0xec 0x04</td>
</tr>
<tr>
<td>4</td>
<td>0x83</td>
<td>0xc3 0x83 0xec 0x14</td>
</tr>
<tr>
<td>5</td>
<td>0x53</td>
<td>0xc3 0x53 0x83 0xec 0x24</td>
</tr>
</tbody>
</table>

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 |
| 0x55 | 0x90 | 0x55 | 0x57 | 0x54 |
| 0x53 | 0xc3 | 0x90 | 0x53 | 0x56 |

ret; push esi; push esi; push esi;
nop; push ebp; push edi; push esp;
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

<table>
<thead>
<tr>
<th>Assembling Code</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>push esi; push edi; push ebx; sub 0x70, esp;</td>
<td>6.6%</td>
</tr>
<tr>
<td>nop; push ebp; mov esp, ebp;</td>
<td>4.4%</td>
</tr>
<tr>
<td>ret; push ebp; mov esp,ebp; push ebx; sub 0x4,esp;</td>
<td>3.0%</td>
</tr>
<tr>
<td>ret; sub 0x14, esp;</td>
<td>2.2%</td>
</tr>
<tr>
<td>ret; push ebx; sub 0x24, esp;</td>
<td>0.7%</td>
</tr>
</tbody>
</table>
Troubleshooting beforehand and patching model errors

Indicators for function start appear in the middle of a function.

“[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

Pinpointing the important features → Synthesizing some new training samples → Retraining the target model

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

<table>
<thead>
<tr>
<th>Metric</th>
<th>Before Patch</th>
<th>kp=5</th>
<th>kp=10</th>
<th>kp=20</th>
<th>kp=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>95.13%</td>
<td>99.08%</td>
<td>96.45%</td>
<td>96.51%</td>
<td>97.00%</td>
</tr>
<tr>
<td>R</td>
<td>95.90%</td>
<td>89.62%</td>
<td>97.46%</td>
<td>97.47%</td>
<td>97.00%</td>
</tr>
<tr>
<td>F1</td>
<td>95.52%</td>
<td>94.11%</td>
<td>96.95%</td>
<td>96.99%</td>
<td>97.00%</td>
</tr>
<tr>
<td>A</td>
<td>99.97%</td>
<td>99.96%</td>
<td>99.98%</td>
<td>99.98%</td>
<td>99.98%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>No.</th>
<th>Before Debugging</th>
<th>kp=5</th>
<th>kp=10</th>
<th>kp=20</th>
<th>kp=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>83.68%</td>
<td>51.78%</td>
<td>73.30%</td>
<td>72.16%</td>
<td>51.94%</td>
</tr>
<tr>
<td>2</td>
<td>90.16%</td>
<td>64.98%</td>
<td>85.36%</td>
<td>84.54%</td>
<td>62.80%</td>
</tr>
<tr>
<td>3</td>
<td>92.06%</td>
<td>72.32%</td>
<td>88.62%</td>
<td>88.88%</td>
<td>70.00%</td>
</tr>
<tr>
<td>4</td>
<td>88.02%</td>
<td>62.28%</td>
<td>83.60%</td>
<td>82.30%</td>
<td>56.18%</td>
</tr>
<tr>
<td>5</td>
<td>90.66%</td>
<td>67.56%</td>
<td>85.54%</td>
<td>83.48%</td>
<td>68.30%</td>
</tr>
</tbody>
</table>
Future Work

The target program

RNN in Learn&Fuzz

PCFG in REINAM
Questions?