Vulnerability Detection in Mobile Applications using State Machine Modeling

Wesley van der Lee
TUDelft
April 23th, 2018

Security Protocol Implementations: Development and Analysis (SPIDA)
Introduction

1. Introduction

WhatsApp Security Flaw Lets Hackers Enter Any Group Unnoticed

Vulnerability in mAadhaar Android app allows anyone to steal your ...

Digit - 11 Ian. 2018

Researchers find 147 vulnerabilities in 34 SCADA mobile applications

SC Magazine - 11 Jan. 2018

IoActive and Embedi researchers released a whitepaper outlining 147 vulnerabilities in 34 mobile applications used in tandem with Supervisory Control and Data Acquisition (SCADA) systems. The vulnerabilities could allow an attacker to ...

It’s Open Season for Hackers: 90% of Mobile Cryptocurrency Apps ...

Cryptovest (Cryptocurrency & Blockchain News) - 3 uur geleden

What’s probably more shocking than all of this is that the analysis they performed found that 77 percent of popular cryptocurrency applications have at least two high-risk vulnerabilities, making the mobile space a massive party venue for hackers. At the beginning of this month, the CryptoShuffle virus ...
Outline

- State Machine Learning
- Mobile State Machine Learning
- Vulnerability Detection
- Results
- Conclusion

1. Introduction
Definition of a State Machine

- States = \{q_0, q_1, q_2\}
- Start = q_0
- Transitions =
- Alphabet = \{a, b\}
- Accepting states = \{q_0, q_2\}

Example State Machine
MAT Framework

Minimally Adequate Teacher (Angluin, 1988)

Learner

Membership Query(input)

output

Yes: done
No: Counterexample

Teacher

simulation(input)

output

• Membership Queries
• Equivalence Queries

SUT

2. State Machine Learning

Active Learning with L*

Observation:
- Alphabet = \{a, b\}
- Membership Query \{ε, a, b\}
- Output: \{ε=1, a=1, b=0\}

<table>
<thead>
<tr>
<th>Observation</th>
<th>ε</th>
<th>b</th>
<th>a</th>
<th>ba</th>
<th>bb</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>bb</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( S = \text{Prefixes (row)} \)
\( E = \text{Suffixes (column)} \)

Hypothesis:
- 2. State Machine Learning
- Rows:
  - Unique rows: identify states

Alphabet = \{a, b\}
Active Learning with L*  

2. State Machine Learning

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>☒</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>☒</td>
</tr>
<tr>
<td>bb</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>aa</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Hypothesis

\[
\begin{align*}
A[\varepsilon] & \xrightarrow{b} A[b] \\
A[\varepsilon] & \xrightarrow{a} A[a] \\
A[b] & \xrightarrow{a, b} \cdot \\
A[a] & \xrightarrow{b} \cdot \\
\cdot & \xrightarrow{a} \cdot \\
\cdot & \xrightarrow{b} \cdot
\end{align*}
\]

NO! YES! Equivalence: Equivalence Query H↓2

Equivalence Queries

- Conformance Testing
  - Random Walk
  - HappyFlow
- W-Method
  - $P \times \Sigma^\star n \times W$ (Chow, 1978)
  - $P \times \Sigma^\star n \times W'$ (Smetsers et al., 2016)

Problems when learning

3. Mobile State Machine Learning
Problems when learning

- Non-deterministic behavior
- Inconsistent cache
- Cache Roll-Back
Video

3. Mobile State Machine Learning
3. Mobile State Machine Learning

9292 Model
Vulnerability Detection

- Enrich Models
- Define Vulnerability Algorithms
  - OWASP Top10 Mobile
    - i.e. Insecure Authentication:
      
      Does there exist a path to a node after the login state without traversing the login state?
Results

- InsecureBankV2
  - Known Vulnerable App

- Fake WhatsApp
  - Adware
  - Malware Spreading
InsecureBankV2

5. Results

POST http://57.97.2.11:8888/login

Insecure Authentication
Insecure Communication
Fake WhatsApp

Background: Unicode trick allows malicious developers to impersonate WhatsApp developers > 1M downloads, now deleted from Play Store
Fake WhatsApp

- Extraneous Functionality
  - SUT > REF: 100%
  - REF > SUT: 100%

- Insecure Communication
  GET http://req.startappservice.com/1.4/..

5. Results
Run statistics

**Tab. 4.1:** Statistics for active learning the inferred state machine for the 9292 application using $L^*$ and RandomWalk

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>$L^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalence Oracle</td>
<td>RandomWalk</td>
</tr>
<tr>
<td>Membership Queries</td>
<td>1778</td>
</tr>
<tr>
<td>Equivalence Queries</td>
<td>1</td>
</tr>
<tr>
<td>States</td>
<td>9</td>
</tr>
<tr>
<td>Transitions</td>
<td>25</td>
</tr>
<tr>
<td>Learning Time</td>
<td>26:06 (hh : mm)</td>
</tr>
</tbody>
</table>

**Tab. 4.2:** Statistics for active learning the inferred machine for the 9292 application using TTT and RandomWalk

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>TTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalence Oracle</td>
<td>RandomWalk</td>
</tr>
<tr>
<td>Membership Queries</td>
<td>55</td>
</tr>
<tr>
<td>Equivalence Queries</td>
<td>2</td>
</tr>
<tr>
<td>States</td>
<td>2</td>
</tr>
<tr>
<td>Transitions</td>
<td>4</td>
</tr>
<tr>
<td>Learning Time</td>
<td>00:21 (hh : mm)</td>
</tr>
</tbody>
</table>

**Tab. 4.3:** Statistics for Active Learning the Inferred Machine for the 9292 application using TTT, $L^*$ and WMethod-Minimal

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>TTT</th>
<th>$L^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalence Oracle</td>
<td>WMethod-Minimal</td>
<td>WMethod-Minimal</td>
</tr>
<tr>
<td>Membership Queries</td>
<td>15619</td>
<td>16966</td>
</tr>
<tr>
<td>Equivalence Queries</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>States</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Transitions</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Learning Time</td>
<td>23:14 (hh : mm)</td>
<td>27:37 (hh : mm)</td>
</tr>
</tbody>
</table>
Run statistics

<table>
<thead>
<tr>
<th>InsecureBankv2</th>
<th>Fake WhatsApp</th>
<th>Real WhatsApp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Time</td>
<td>3:13</td>
<td>7:22</td>
</tr>
<tr>
<td>Learning Algorithm</td>
<td>TTT</td>
<td>TTT</td>
</tr>
<tr>
<td>Equivalence Oracle</td>
<td>WMethod-Minimal</td>
<td>WMethod-Minimal</td>
</tr>
<tr>
<td>Equivalence Queries</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Membership Queries</td>
<td>11627</td>
<td>29.751</td>
</tr>
<tr>
<td>States</td>
<td>6</td>
<td>22</td>
</tr>
<tr>
<td>Alphabet Size</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>Cache Hit</td>
<td>49%</td>
<td>71%</td>
</tr>
<tr>
<td>Fast Forwarded</td>
<td>42%</td>
<td>67%</td>
</tr>
</tbody>
</table>

Tab. 6.1: Learning statistics for InsecureBankv2

<table>
<thead>
<tr>
<th></th>
<th>Fake WhatsApp</th>
<th>Real WhatsApp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Time</td>
<td>7:22</td>
<td>5:47</td>
</tr>
<tr>
<td>Learning Algorithm</td>
<td>TTT</td>
<td>TTT</td>
</tr>
<tr>
<td>Equivalence Oracle</td>
<td>WMethod-Minimal</td>
<td>WMethod-Minimal</td>
</tr>
<tr>
<td>Equivalence Queries</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Membership Queries</td>
<td>29.751</td>
<td>25.083</td>
</tr>
<tr>
<td>States</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>Alphabet Size</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>Cache Hit</td>
<td>71%</td>
<td>69%</td>
</tr>
<tr>
<td>Fast Forwarded</td>
<td>67%</td>
<td>62%</td>
</tr>
</tbody>
</table>

Tab. 6.3: Learning statistics for both WhatsApp versions
Conclusion

How can one identify weaknesses in mobile Android applications through feasible behavioral state machine learning?

L* TTT

• RandomWalk + HappyFlow
• W-Method Minimal
Future work

- It is feasible, but slow, possible speedups:
  - Obtaining the possible actions without testing
  - Using other information, e.g. the screen content, during learning
  - …

- Performing the security analysis using a model checker

- Solving inconsistencies caused by popups or the keyboard not closing
Thank You
\begin{enumerate}
\item $S = E = \{\lambda\}$
\item $(S, E, T) \leftarrow MQ(\lambda) \cup \forall a \in A : MQ(a) \neq (S, E, T)$ is the observation table
\item While $M$ is incorrect: \# $M$ is the conjecture
  \begin{enumerate}
  \item While $(S, E, T)$ is not consistent or not closed
    \begin{enumerate}
    \item if $(S, E, T)$ is not consistent:
      \begin{enumerate}
      \item $\exists(s_1, s_2) \in S, a \in A, e \in E : \text{row}(s_1) = \text{row}(s_2)$ and $T(s_1 \cdot a \cdot e) \neq T(s_2 \cdot a \cdot e)$
      \end{enumerate}
      \begin{enumerate}
      \item $E \leftarrow a \cdot e$
      \item extend $T$ to $(S \cup S \cdot A) \cdot E$ using $MQ$
      \end{enumerate}
    \end{enumerate}
    \item if $(S, E, T)$ is not closed:
      \begin{enumerate}
      \item $\exists s_1 \in S, a \in A : \forall \text{row}(s_1 \cdot a) \neq \text{row}(s)$
      \item $S \leftarrow s_1 \cdot a$
      \item extend $T$ to $(S \cup S \cdot A) \cdot E$ using $MQ$
      \end{enumerate}
    \end{enumerate}
  \end{enumerate}
\item $M = M(S, E, T) \neq (S, E, T)$ is closed and consistent
\item if Teacher replies with a counterexample $t$:
  \begin{enumerate}
  \item add $t$ and all its prefixes to $S$
  \item extend $T$ to $(S \cup S \cdot A) \cdot E$ using $MQ$
  \end{enumerate}
\item Halt and output $M$
\end{enumerate}
Fig. 2.4: Formal progression of an incorrect conjecture: (a) inconsistent model for distinguishing suffix $v$ from state $q$, (b) consistent model after splitting $q$ into new states $q_1$ and $q_2$. 
<table>
<thead>
<tr>
<th></th>
<th>Improper Platform Usage</th>
<th>Insecure Communication</th>
<th>Insecure Authentication</th>
<th>Extraneous Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>InsecureBank</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>--</td>
</tr>
<tr>
<td>FreeCola</td>
<td>+/−</td>
<td>−/−</td>
<td>−/+</td>
<td>−/−</td>
</tr>
<tr>
<td>9292</td>
<td>−/−</td>
<td>−/−</td>
<td>−/−</td>
<td>−/−</td>
</tr>
<tr>
<td>Benign WhatsApp</td>
<td>−/−</td>
<td>−/−</td>
<td>−/−</td>
<td>−/−</td>
</tr>
<tr>
<td>Malicious WhatsApp</td>
<td>++</td>
<td>++</td>
<td>−/−</td>
<td>+/+</td>
</tr>
</tbody>
</table>

**Tab. 7.2:** The expected results versus the expected results for all applications under test.
Algorithm 7 Improper Platform Usage Identification

Input: Inferred State Machine $M = \{Q, \Sigma, \delta, q_0, F\}$
Output: Returns a set $R$ of activities that induce supplementary behavior in machine $M$. There possibly exists a vulnerability if $R$ is non-empty.

1: $R \leftarrow \emptyset$
2: $A \leftarrow$ activities from $M$
3: for all $a$ in $A$ do
4:     if $a$ is a callable activity then
5:         add $a$ to $R$
6:     end if
7: end for
8: for all $q$ in $Q$ and $R$ is not empty do
9:     remove $q$'s activity from $R$
10: end for
11: return $R$

Algorithm 8 Insecure Communication Identification

Input: Inferred State Machine $M = \{Q, \Sigma, \delta, q_0, F\}$
Output: Returns a set $R$ of requests that made by actions in machine $M$ that do not adhere to common network security standards. There possibly exists a vulnerability if $R$ is non-empty.

1: $R \leftarrow \emptyset$
2: for all $t$ in $\delta$ do
3:     $r \leftarrow$ request_insecure($t$.request)
4:     if $r$ then
5:         add $r$ to $R$
6:     end if
7: end for
8: return $R$
Algorithm 9 Insecure Authentication Identification
Input: Inferred State Machine $M = \{Q, \Sigma, \delta, q_0, F\}$
Output: Returns a set $R$ of authentication bypass techniques in machine $M$.

There possibly exists a vulnerability if $R$ is non-empty.
1: $a \leftarrow$ authentication state of $M$ \quad $\triangleright$ $a \in Q$
2: if $a$ is null then \quad $\triangleright$ no authentication $\rightarrow$ no authentication bypass
3: return $R$
4: end if
5: Marks $\leftarrow$ subset of nodes possible to reach after $a$
6: $M' \leftarrow M - a$ \quad $\triangleright$ the machine without the authentication state
7: for all $m$ in Marks do
8: if a path from the $q_0$ to $m$ exists in $M'$ then
9: add path to $R$
10: end if
11: end for
12: $Q'' \leftarrow Q - Marks$
13: $A \leftarrow$ callable activities
14: for all $q$ in $Q''$ and $A$ is not empty do
15: remove $q$’s activity from $A$
16: end for
17: add $A$ to $R$
18: return $R$

Algorithm 10 Code Tampering Identification
Input: Inferred machine $M = \{Q, \Sigma, \delta, q_0, F\}$ and reference machine $M' = \{Q', \Sigma', \delta', q_{0}', F'\}$
Output: Finds difference in $M$ and $M'$ Returns a set $R$ of sequences that yield a different output for the two machines $M$ and $M'$. The sequences are divided into sets $R_1$ and $R_2$ which depict what machine models the sequence. There possibly exists a vulnerability if $R$ is non-empty.
1: $R_1, R_2 \leftarrow \emptyset$
2: $TCS_1 \leftarrow TCS(M)$
3: $TCS_2 \leftarrow TCS(M')$
4: for all $w$ in $TCS_1$ do
5: if $\lambda^M(w) \neq \lambda^{M'}(w)$ then
6: $R_1 \leftarrow w$
7: end if
8: end for
9: for all $w$ in $TCS_2$ do
10: if $\lambda^M(w) \neq \lambda^{M'}(w)$ and $w \notin R$ then
11: $R_2 \leftarrow w$
12: end if
13: end for
14: $R \leftarrow R_1, R_2$
15: return $R$