MACE:
Model-inference-Assisted Concolic Exploration

Domagoj Babic
http://www.domagoj.info/
joint work with Chia Yuan Cho, Pongsin Poosankam,
Kevin Zhijie Chen, Edward XueJun Wu, Dawn Song
UC Berkeley
Software Security

- Malware deluge: >60,000 samples / day
- One of the main attack vectors
  - Software flaws!
- Thousands of CVEs each year
- Cures: verification, testing

Domagoj Babic, USENIX 2011
Outline

• Dynamic Symbolic Execution (a.k.a. DART, concolic execution)
  – High-level intro
  – Aspects that could be improved
• Model-inference-Assisted Concolic Exploration
  – How it works
  – How it improves over dynamic symbolic execution
• Experimental results
Dynamic Symbolic Execution

- Independently invented by several groups in 2004/2005
- Main components:
  - Concrete execution
  - Symbolic execution
  - Solver (decision procedure)
- Very effective in practice

\[
\begin{align*}
x_0 &= a + b; \\
\text{if } x_0 < 0 & \text{ then } \\
x_1 &= -x_0; \\
y_1 &= y_0; \\
\text{else } \\
x_1 &= x_0; \\
y_1 &= y_0 + x_0; \\
\end{align*}
\]
Learning

• Dynamic symbolic execution
  – Repeats iterations (concrete + symbolic) until terminated
  – Knowledge gained from iterations discarded

• Research questions:
  – What can be learned from iterations?
  – How can one represent the gained knowledge?
  – How could that knowledge prune the search space?
MACE – The Main Ideas

- Learning + dynamic symbolic execution
- Learns a state-machine abstracting the program
  - Guides further search
    - Initialize the program to certain state
    - Explore the neighborhood
  - Specifies sequences of inputs required to get to a certain state
The L* Algorithm

- MACE uses an improved L* [CCS’2010]
- Polynomial in the number of states and size of the input message set $M_i$
- Constructs an observation table
- Reads off states and transitions from the table

**Observation Table**

**Black box**

**L***

Sequences of input messages from $M_i$

Sequences of output messages from $M_o$
The MACE Approach

Seed messages

L*

Shortest sequence generator

State Explorer

Input sequences

Finite state-machine

Filter

Set of input messages

Input and output sequences

Domagoj Babic, USENIX 2011
Key Difficulty: Abstraction of Messages

- Inferring the state-machine over all messages
  - Computationally infeasible
  - Useless for guidance
- $L^*$ operates over an abstract set of messages
- In prior work [CCS’10] – manually written abstractions
- MACE: automatic abstraction of input messages
Filtering Function

• The main idea: keep only the messages that refine the state-machine
• Exact check too expensive, use an approximation
  • If the current state-machine can produce the given output sequence, no refinement
  • Otherwise, add all the input messages from the corresponding input sequence

\[ A \times M \times M \rightarrow \]

\[ = \left\{ \exists \in \lambda = \right\} \]
Implementation

• Dynamic symbolic execution engine
  – BitBlaze infrastructure

• L*
  – Our implementation with improvements from the CCS’2010 botnet analysis paper

• Scripts
  – For gluing the components together
Applications of MACE

• Guiding dynamic symbolic execution
  – Different abstractions suitable for different types of applications
  – E.g., inference of context-free grammars for automated testing of applications with parsers
• Protocol reverse engineering
  – Comparative analysis (e.g., for extracting signatures)
  – Protocol state-machine model checking
Experimental Setup

• DETER Security testbed (3GHz Intel Xeon processors)
• State-space exploration done in parallel
  – One job per state in the inferred state-machine
  – 2.5 hr timeout per state
  – Each newly discovered state explored only once
• For coverage measurement experiments
  – Baseline got extra time, compensates for the time spent in learning
Benchmarks

• Inference done on
  – Remote Frame Buffer (RFB) protocol: Vino 2.26.1
  – Server Message Block (SMB) protocol: Samba 3.3.4

• State-space exploration also done on
  – RealVNC
  – Win XP SMB

• Seed message set
  – Vino: 45 sec session of a remote desktop session
  – Samba: used gentest suite
## Results: Iterations and Runtime

<table>
<thead>
<tr>
<th>Program</th>
<th>Iteration</th>
<th>States</th>
<th>Input alphabet size</th>
<th>Output alphabet size</th>
<th>Learning time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vino</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>142</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7</td>
<td>12</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Samba</td>
<td>1</td>
<td>40</td>
<td>40</td>
<td>14</td>
<td>2028</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>84</td>
<td>54</td>
<td>24</td>
<td>1840</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>84</td>
<td>55</td>
<td>25</td>
<td>307</td>
</tr>
</tbody>
</table>
Results: Inferred Protocol Models

Inferred 84-state SMB protocol implementation abstraction
## Results: Discovered Vulnerabilities

<table>
<thead>
<tr>
<th>Program</th>
<th>Vulnerability</th>
<th>New</th>
<th>MACE (hrs)</th>
<th>Baseline (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vino</td>
<td>CVE-2011-0906</td>
<td>✓</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>CVE-2011-0905</td>
<td>✓</td>
<td>4</td>
<td>&gt;105</td>
</tr>
<tr>
<td></td>
<td>CVE-2011-0904</td>
<td>✓</td>
<td>15</td>
<td>&gt;105</td>
</tr>
<tr>
<td>Samba</td>
<td>CVE-2010-2063</td>
<td>✓</td>
<td>12</td>
<td>602</td>
</tr>
<tr>
<td></td>
<td>CVE-2010-1642</td>
<td>✓</td>
<td>14</td>
<td>&gt;1260</td>
</tr>
<tr>
<td></td>
<td>Fixed without CVE</td>
<td></td>
<td>124</td>
<td>&gt;1260</td>
</tr>
<tr>
<td>RealVNC</td>
<td>CVE-2011-0907</td>
<td>✓</td>
<td>2</td>
<td>&gt;105</td>
</tr>
<tr>
<td>Win XP SMB</td>
<td>None</td>
<td></td>
<td>&gt;210</td>
<td>&gt;1260</td>
</tr>
</tbody>
</table>
## Results: Coverage Improvement

<table>
<thead>
<tr>
<th>Program</th>
<th>Instruction Coverage Baseline</th>
<th>Instruction Coverage MACE</th>
<th>Coverage Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vino</td>
<td>129762</td>
<td>138232</td>
<td>6.53</td>
</tr>
<tr>
<td>Samba</td>
<td>66693</td>
<td>105946</td>
<td>58.86</td>
</tr>
<tr>
<td>RealVNC</td>
<td>39300</td>
<td>47557</td>
<td>21.01</td>
</tr>
<tr>
<td>Win XP</td>
<td>90431</td>
<td>112820</td>
<td>24.76</td>
</tr>
</tbody>
</table>

Domagoj Babic, USENIX 2011
Results: Exploration Depth (SMB)

Domagoj Babic, USENIX 2011
Why MACE Works so Well?

• Uses a relatively cheap technique (L*) to infer an abstraction of the search space and reduce the search space

• The abstraction is used to guide the search
  – Especially useful for constructing sequences of messages to get to certain state

• More control over the search
  – E.g., decreases the probability of getting stuck in loops
Summary

• Model-inference-Assisted Concolic Execution
  – How it works
  – How it improves dynamic symbolic execution

• Experimental results
  – 7X more vulnerabilities found
  – Up to 58% better coverage
  – Deeper states explored

Domagoj Babic, http://www.domagoj.info/