A Roadmap to Combinatorial Security Testing

Dimitris E. Simos

SBA Research, Vienna
April 10, 2015
Outline of the Talk

Theory of Combinatorial Testing
Arrays and Algorithms
Outline of the Talk

Theory of Combinatorial Testing
  Arrays and Algorithms

Applications for Information Security
  Application Security
  Embedded Systems Security
  Network Security
  Hardware Malware
Distilling the Vision

SBA Research Area:
Applied Mathematics for Information Security
- Combinatorics and Codes for Future Cryptography
- Combinatorial Security Testing

Long-Term Plan and Objectives
- Combinatorial security testing will go mainstream in 2016-2021
- Optimization and automation of security tests
- Mathematically guaranteed levels of trustworthiness in evaluation results

Positioning within SBA Research
- Strategic research area
- Interplay between basic and applied research
- Feedback cycle to secure software engineering
Arrays and Algorithms for CT

Classes of Arrays for Security Testing

Advancements needed in:

- MCAs (efficient test generation algorithms)
- FLAs/ELAs (identification of faulty interactions)
  - Literature beyond binary alphabet extremely sparse

ACO Variant for IPOG Family of Algorithms

- Use shared knowledge to make efficient decisions regarding the parameter value selection
- Targeting to improve horizontal and vertical growth of IPOG family
- Develop new CA building strategies

Other Evolutionary Approaches

- Competent GAs, Messy GAs, Hybrid Approaches
- Model in tuple level
Application Security

Next Steps in Web Security Testing

- Automated testing for real world applications
  - Access to large-scale test servers and automated setup environments is needed!
- Prioritization of XSS attack vectors; Guided combinatorial testing
- Wider study of how CT algorithms affect XSS (report)
- Release of XSSInjector for the research community

Future Directions

- Modelling SQL Injection attacks
- Directly applicable to database and application security
- **Notorious Examples:** Microsoft SQL Server Databases (2009), Yahoo! stolen credentials $\approx 500k$ (2012), Russian hackers theft of 1.2 billion credentials (2014)
Embedded Systems Security

**ERIS**∞ for Combinatorial Kernel Testing

- Extend to Android APIs $\Rightarrow$ Mobile security
- OS Kernels $\Rightarrow$ Systems security
  - Industrial Automation Control Systems (IACS) testing
  - Cyber-physical Systems (CPS) testing

**ERIS**∞ in a Nutshell

- Sequences of systems calls $\Rightarrow$ Sequence CAs
- Continuous integration of kernel versions
- Web monitoring platform
- Testing of security patches to ensure attack-free environments
Network Security

Protocol Testing

- **Motivation:** Major security breaches recently; NIST is currently revising the RFCs (standards)
- **SUTs:** TLS, SSL, SSH (c.f. Internet Protocol Suite)
- **Goal:** Quality assurance of protocol implementations

Protocol Interaction Testing

- **Aim:** Assure proper error handling
- Test protocol implementations for erroneous configurations that lead to security flaws; IPM for protocol parameters

Table: TLS Cipher Suite Registry

<table>
<thead>
<tr>
<th>Value</th>
<th>Parameters</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00,0x07</td>
<td>TLS_RSA_WITH_IDEA_CBC_SHA</td>
<td>[RFC5469]</td>
</tr>
<tr>
<td>0x00,0x08</td>
<td>TLS_RSA_EXPORT_WITH_DES40_CBC_SHA</td>
<td>[RFC4346]</td>
</tr>
<tr>
<td>0x00,0x0A</td>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>[RFC5246]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Network Security

Certificate Testing

- Standards for public key infrastructure (PKI)
- Attack vectors have the purpose to forge certificates
- **Oracle:** Test whether the server/client accepts them as valid

(Sample) Structure of an X.509v3 Certificate

- Version
- Serial Number
- Algorithm ID
- Issuer
- Validity
  - Not Before
  - Not After
- Public Key Algorithm
- Subject Public Key
Network Security

**Figure:** Handshake Message Sequence Diagram

**t-way Sequence Testing**
Model the event sequence of TLS Handshake (MBT, conformance testing)
Hardware Trojan Horses

Targeting cryptographic circuits that perform encryption and decryption in one FPGA design

- Block ciphers (AES), Stream Ciphers (Mosquito)

Instances of Hardware Trojan Horses

- **Combinational**: Activate when a specific combination of key bits appear
- **Sequential**: Activate after a counter has elapsed (time-bombs)
Hardware Trojan Horse Function

- The trigger part consists of 7 AND gates and monitors 8 key bits
- When at least one input is "0", the Trojan does nothing malicious
- When all inputs are "1", the Trojan payload part (just 1 XOR gate!) is activated
- The payload part reverses the mode of operation (encryption or decryption) \[\implies\] DoS attack until key is changed
Exciting Hardware Trojan Horses

IPM for Ciphers

• **Attack vectors:** Model triggering sequences of the trojan
• For example, in stream ciphers $K$, $P$, $IV$
• **Input space:** $2^{128}$ for 128 bits key

Detection and Identification

• Large covering arrays for detection (triggering sequence)
  ▶ Thanks to Jose Torres-Jimenez (NIST/ACTS group)
• Error/fault locating arrays for identification (exact location of key bits)

Table: Test Suite Sizes

<table>
<thead>
<tr>
<th>$k$</th>
<th>$t$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>128</td>
<td>3</td>
<td>51</td>
</tr>
<tr>
<td>128</td>
<td>4</td>
<td>144</td>
</tr>
<tr>
<td>128</td>
<td>5</td>
<td>252</td>
</tr>
<tr>
<td>128</td>
<td>6</td>
<td>1236</td>
</tr>
<tr>
<td>128</td>
<td>7</td>
<td>2467</td>
</tr>
<tr>
<td>128</td>
<td>8</td>
<td>17549</td>
</tr>
</tbody>
</table>
Figure: The Mosquito realization in an FPGA including some Transient Effect Ring Oscillators (red box) for detecting the Trojan. One Trojan in the yellow box and the other in the blue box (in total, three different implementations).
Analysis of Security Vulnerabilities

Analysis of Test Suites

- Countless Common Vulnerabilities and Exposures (CVEs)
- Dedicated CVE database for security community

Can we Do Better?

- No notion of combinatorial coverage measurement has been applied
- Number of parameters of the SUT and exact parameter value configurations that trigger security vulnerabilities is (mostly) undetermined so far (e.g. Heartbleed bug, Hardware trojans)

Measuring $t$-wise Coverage for Combinatorial Security Testing

- **Requirements**: CCM, classifiers, feature model extraction
- Analogue to NIST study for NASA spacecrafts, medical devices..
- **Goal**: Automation, reduction, fault-localization as proactive defenses
Questions - Comments

Thank you for your Attention!

dsimos@sba-research.org