Automated Assessment of Secure Search Systems

Nicholas Hwang
on behalf of the MIT LL SPAR T&E Team

nicholas.hwang@ll.mit.edu
spar-lincoln@ll.mit.edu

October 7, 2014
How Should We Assess Software?

Software Assessment Framework

Assessment Report

START  STOP

Software Under Test

Baseline Software
Software T&E Mantras

- Black box treatment of systems
- End-to-end automation
- Test-time agility
- Real-time status reports
- Minimal system overhead
- Repeatability
- Rapid environment reconfiguration
- Extensibility
- Comprehensive data capture
- Reusability

ITEA Symposium 2014 - 3
Hwang 10/7/2014
Incorporating Software T&E Mantras

- Real-time status reports
- Rapid environment reconfiguration
- End-to-end automation
- Black box treatment of systems

Software Under Test
Baseline Software

Test Status
Test Parameters
Assessment Report
Our Contributions

Mantras for **Effectively** Performing **Effective** Software Assessments

An Exemplar Software Assessment Framework

https://github.com/mitll-csa/sparta
T&E Case Study: SPAR

- Security and Privacy Assurance Research (SPAR)
- IARPA-funded program, managed by W. Konrad Vesey
- Nine research teams in three Technical Areas (TAs)

<table>
<thead>
<tr>
<th>Phase 1 (18 Months)</th>
<th>Phase 2 (15 Months)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TA1:</strong> Practical Security and Privacy Assurance For Database Access</td>
<td>Stealth</td>
</tr>
<tr>
<td><strong>TA2:</strong> Homomorphic Encryption for Evaluation of Assured Data Retrieval Functions</td>
<td>Stealth</td>
</tr>
<tr>
<td><strong>TA3:</strong> Privacy Protection in Publish/Subscribe Systems</td>
<td></td>
</tr>
</tbody>
</table>

Independent Test & Evaluation

- Q1 FY12
- Q3 FY13
- Q3 FY14
Traditional Database Systems

Client learns:
- Access policies

Cloud learns:
- Access policies
- Queries
- Query results
- Query access patterns

Owner learns:
- Queries
- Query results
- Query access patterns

Security and Privacy Guarantees
Operation Speed

COTS products optimized for speed, not security or privacy.
Theoretical SPIR Database Systems

Client learns:
- Access policies
Cloud learns:
- Access policies
- Query
- Query results
- Query access patterns
Owner learns:
- Query
- Query results
- Query access patterns

SPIR achieves security and privacy, but is impractical and slow.
SPAR Database System Goals

Client learns:
- Access policies

Cloud learns:
- Access policies
- Query
  - Query structure
  - Query results
  - Result set size
  - Query access patterns

Owner learns:
- Query
- Query results
- Query access patterns

SPAR systems achieve a practical balance between security, privacy, and performance.
Lincoln’s T&E Role

Assess SPAR research prototypes for correctness, functionality, and performance.
Software Assessment Framework

- Test Preparation Tools
- Test Execution Tools
- Test Reporting Tools

Cloud Storage

- Encrypted DB

Data Owner

- Plaintext Test DB

Owner Node

Cloud Storage

- Encrypted Test DB

Cloud Node

Client

Client Node
Common Test Preparation Challenges

How can we generate good tests? Can we determine the correct answers beforehand?

How can we generate realistic, synthetic test data?
Our Test Preparation Tools

Our Test Preparation Tools include:

- Data & Query Generator
- Test Script Generator
- Results Database
- Test Scripts
- Test Logs
- Test Harness
- Test Orchestrator
- Test Logs
- Test Results
- System Monitors

SPAR Challenges:

- Generate 10TB databases within 12 hours
- Generate “interesting” queries

NOTE: Nodes can represent either performer or baseline software.
<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eq</td>
<td>Single-field equality</td>
<td>fname = ‘Homer’</td>
</tr>
<tr>
<td>P1</td>
<td>Boolean operations</td>
<td>Iname = ‘Simpson’ AND city = ‘Springfield’</td>
</tr>
<tr>
<td>P2</td>
<td>Numeric ranges</td>
<td>age BETWEEN 38 and 40</td>
</tr>
<tr>
<td>P3</td>
<td>Free-text keyword</td>
<td>CONTAINED_IN(notes1, ‘donut’)</td>
</tr>
<tr>
<td>P4</td>
<td>Stemming</td>
<td>CONTAINS_STEM(notes2, ‘work’)</td>
</tr>
<tr>
<td>P6</td>
<td>Wildcard</td>
<td>notes3 LIKE ‘%oo %oo!’</td>
</tr>
<tr>
<td>P7</td>
<td>Substring</td>
<td>notes4 LIKE ‘%mmm%’</td>
</tr>
<tr>
<td>P8</td>
<td>Threshold</td>
<td>M_OF_N(2, 3, income &gt; 40000, citizenship = ‘Yes_Born_In_US’, marital_status = ‘Married’)</td>
</tr>
<tr>
<td>P9</td>
<td>Ranking</td>
<td>M_OF_N(2, 3, income &gt; 40000, citizenship = ‘Yes_Born_In_US’, marital_status = ‘Married’) ORDER BY RANK</td>
</tr>
<tr>
<td>P11</td>
<td>Searching on XML</td>
<td>xml XPATH xml//company = ‘Springfield Nuclear Power Plant’</td>
</tr>
</tbody>
</table>
Data & Query Generators

Data Generator

- Training Data
- Learner
- Row Generator
- Test Database

Query Generator

- Query Schema
- Query Creator
- Query Refiner
- Queries

Critical Achievements

- Repeatability
- Extensibility
- Reusability
Common Test Execution Challenges

How can we configure the test environment with minimal time and overhead?

How can we actuate performer systems with minimal overhead?

NOTE: Nodes can represent either performer or baseline software.
Our Test Execution Tools

**SPAR Challenges**

- Handle unbounded result set sizes on Client Node
- Simultaneously orchestrate tests on 30+ nodes

NOTE: Nodes can represent either performer or baseline software
Test Harness Protocol Handler

<table>
<thead>
<tr>
<th>Operation</th>
<th>string</th>
<th>rope</th>
<th>knot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>O(1)</td>
<td>O(1) [lazy]</td>
<td>O(1)</td>
</tr>
<tr>
<td>Concatenate</td>
<td>O(n)</td>
<td>O(1) [lazy]</td>
<td>O(1)</td>
</tr>
<tr>
<td>Delete</td>
<td>O(n)</td>
<td>O(1) [left/right]</td>
<td>O(1)</td>
</tr>
<tr>
<td>Substring</td>
<td>O(n)</td>
<td>O(1) [left/right]</td>
<td>O(1)</td>
</tr>
<tr>
<td>Iteration</td>
<td>O(n)</td>
<td>O(1) [lazy]</td>
<td>O(1)</td>
</tr>
</tbody>
</table>

Average case run-time

Critical Achievements

- Minimal system overhead
- Reusability
config.py

c0["host"] = "owner"
c0["executable"] = "test-harness"
c0["args"] = "-p owner-sut"

cl["host"] = "client"
c1["executable"] = "test-harness"
c1["args"] = "-p client-sut"

Critical Achievements

- Minimal system overhead
- Comprehensive data capture
- Real-time status reports
- End-to-end automation
- Rapid environment reconfiguration
- Test-time agility
- Reusability
Common Test Reporting Challenges

How can we perform accurate forensics on our test data, possibly months in the future?

How much of the assessment report can we automatically compose?
Our Test Reporting Tools

SPAR Challenge

Sponsor-ready draft report one day after test completion
Critical Achievements

- Comprehensive data capture
- Extensibility
- End-to-end automation

**Figure 4:** Automatically-generated graph displaying one performer's query latency as a function of the number of matching records. Master's thesis, Massachusetts Institute of Technology, 2013.

<table>
<thead>
<tr>
<th>b=1</th>
<th>b=2</th>
<th>b=3</th>
<th>b=4</th>
<th>b=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=0</td>
<td>0</td>
<td>13</td>
<td>74</td>
<td>87</td>
</tr>
<tr>
<td>a=1</td>
<td>86</td>
<td>89</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>a=2</td>
<td>86</td>
<td>90</td>
<td>96</td>
<td>99</td>
</tr>
<tr>
<td>a=3</td>
<td>88</td>
<td>90</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>a=4</td>
<td>89</td>
<td>94</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

% queries where performer ≤ a + b · baseline
Software T&E Mantras

- Black box treatment of systems
- End-to-end automation
- Test-time agility
- Real-time status reports
- Minimal system overhead
- Repeatability
- Rapid environment reconfiguration
- Extensibility
- Comprehensive data capture
- Reusability
https://github.com/mitll-csa/sparta
Questions?

nicholas.hwang@ll.mit.edu
spar-lincoln@ll.mit.edu