A Secure Toolchain Competition

June 16, 2016

Lee Badger
Christopher Johnson

Computer Security Division
NIST

**Note:** Any mention of a vendor or product is not an endorsement or recommendation.

**Credit:** The proposed competition is based on one of the ideas developed during the Designing a Secure Systems Engineering Competition (DESSEC) workshop run by NSF in 2010: Secure Development Tool Chain.
# Team and Idea Provenance

<table>
<thead>
<tr>
<th>Team</th>
<th>Members</th>
<th>Provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NIST</strong></td>
<td>Lee Badger, Christopher Johnson, Murugiah Souppaya, Larry Keys, Michael Bartock, Jeffrey Cichonski</td>
<td>Based on an idea from Designing a Secure Systems Engineering Competition (DESSEC) workshop run by NSF in 2010: Secure Development Tool Chain</td>
</tr>
<tr>
<td><strong>G2, Inc.</strong></td>
<td>Daniel Shiplett, Scott Wilson, Shawn Webb</td>
<td></td>
</tr>
<tr>
<td><strong>GWU/LeMoyne College</strong></td>
<td>Carl Landwehr</td>
<td></td>
</tr>
<tr>
<td><strong>Provenance</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Objective: Secure Software Through Development Toolchain Competitions

Problem Difficulty
(complexity / time allowed)

More Secure Software

Reproducible results, technology improvements, public data

Participants won in Competitions 1 to 5.
The Problem

- Vulnerabilities are routinely produced by millions of software developers.
- The resulting attacks undermine US competitiveness and security.

Opportunities for Vulnerability Suppression/Mitigation

(simplified)

Software Lifecycle Phases

Design and implementation

Deployment

Operation & Maintenance

Tools

toolchains

People

developers

administrators

operators

3 million in US
(NICE securely provision
IEEE building code for building code)

- Security-focused toolchain enhancements could have large downstream benefits.
- Developer training is also important, but our focus is on the tools.
What is a Toolchain?

toolchain  A collection of software or hardware **mechanisms** that a software developer may use to produce a software entity that can execute on a specific **platform**. Our working definition. Wikipedia has one too.

Some kinds of mechanisms:

- **Build environments**
- **Libraries**
- **Version control systems**
- **Compilers**
- **Debuggers**
- **Modeling tools**
- **Languages**
- **Editors**
- **Code generation tools**
- **Interpreters**
- **Testing tools**
- **Media authoring tools**
- **Frameworks**
- **Linkers**
- **Static analyzers**
- **Integrated development environments**
- **Reverse engineering**
### Some Toolchain Platforms

<table>
<thead>
<tr>
<th>Android</th>
<th>iOS</th>
<th>Blackberry</th>
<th>MS Windows Version X</th>
<th>OS X</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris</td>
<td>Java Virtual Machines</td>
<td>MS .Net</td>
<td>Adobe Flash</td>
<td>Web Browser (e.g., ajax)</td>
<td>Arduino</td>
</tr>
<tr>
<td>Embedded</td>
<td>App X Loadable Modules</td>
<td>OS command line</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Improvements could reduce vulnerability production.
- But, **how can we incentivize security improvements?**
An Iterative Competition to Foster Improved Software Toolchains

Competition Announcement

Time to prepare (Improve Tools!)

~6 months

registration

4-person teams

Formulate lessons learned (NIST publication)

Industry Tool builders

Award Day

If objective scoring threshold achieved: award prize $$ to earliest winning submission

Game Day

Programming assignment

start!

stop!

Work

Work

Work

Work

Solutions

Competition database

Automated Scoring Day

Test a Solution

Scores

By Pearson Scott Foresman [Public domain], via Wikimedia Commons, gnome icon artists
Goal: Identify and Measure the Most Effective Kinds of Development Tools

- To discover what works well, allow nearly all possibilities:
  - Any programming language
  - Any operating system (except in cell phones)
  - Any development methodology
  - Any test/analysis approach or tools
  - Any building-block components
    - E.g., existing frameworks, libraries, custom utilities

(Implicit large submission packages)
Goal: Maximize Objectivity

• Mechanical scoring
  – All tests are formulated before game day
  – All solutions subjected to the same tests
• Public bulletin board for questions
• Scoring **infrastructure source code** published after the testing
• **Goal:** test results will be reproducible
  – (better than repeatable)
• **Requirement:** all test infrastructure software components must be free and available
A Challenge Problem (CP)

• Developed (but not disclosed) before Game Day
• Comprised of 3 parts:

1. **Functional Specification** of the program to develop.
   A white paper (<= 20 pages) with diagrams, in English (including major application states, protocol and data format descriptions).

2. Required **Security Policy**.
   Confidentiality and integrity requirements, function availability requirements, authentication and access control requirements, in English. **Rules of Engagement** specifying permitted/prohibited actions.

3. Problem-specific **Test Suite** (revealed after Game Day)
   20 fully-automated application-specific pass/fail functional tests.
   20 fully-automated application-specific pass/fail security tests.
   **Fuzz tester** configured for the required external interfaces/features.
### Initial Challenge Problem Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Command Line Interface (CLI)** | - Standalone program, launched from an interactive session  
- Can receive file, network, and user keyboard input  
- Perform arbitrary functions; generate any data or protocol  
- Few restrictions on implementing technologies |
| **Mobile**    | - Android application, launched from Android home screen  
- Can receive file, network, Android user interface input  
- Perform arbitrary functions; generate any data or protocol  
- Constrained to Android package format (.apk) |
| **Web**       | - Web application, listens to port 80  
- Can receive file, network, browser user interface input  
- Perform arbitrary functions; generate any data or protocol  
- Constrained to support HTML5 web browsers |
Command-Line Interface (CLI) CPs

- **Participant provides:**
  - Deployable virtual machine (VM) image
    - SSH Daemon with user “testuser” and password “TestPass1!1”
    - Program “do-it” on the testuser’s PATH
    - Any in-VM services needed by do-it already running

- **Test Infrastructure provides:**
  - Configuration files
  - Network-accessible hosts and protocol definition
  - Behavioral specifications (to implement)
  - Sample terminal logs
  - Security properties (to provide)
  - Rules of Engagement
    - Actions that a participant must not take
    - Actions that the test infrastructure will not take

- **Known-answer and fuzz tests are run and scored automatically**
# Mobile App Challenge Problems

- **Participant provides:**
  - An Android Package file (.apk)
  - Specified SDK level

- **Test Infrastructure provides:**
  - GUI components, layout, menu XML files (required)
  - Connected devices
  - Network-accessible hosts and protocol definitions
  - Behavioral specifications (to implement)
  - Security properties (to provide)
  - Rules of Engagement
    - Actions that a participant must not take
    - Actions that the test infrastructure will not take

- **Known-answer and fuzz tests are run and scored automatically**
Web App Challenge Problems

• **Participant provides:**
  – A Deployable virtual machine (VM) image
  – The web app must automatically launch when the VM boots, and host on port 80.
  – The web app must support HTML5 web clients, including Chrome and Firefox.

• **Test Infrastructure provides:**
  – Image and icon files and HTML templates including ID attributes.
  – Network-accessible hosts and protocol definitions
  – Behavioral specifications (to implement)
  – Wire frame mockups of the intended interface
  – Security properties (to provide)
  – Rules of Engagement
    • Actions that a participant must not take.
    • Actions that the test infrastructure will not take.

• **Known-answer and fuzz tests are run and scored automatically**
Sample Mobile Challenge: News App

Security Policy
- Protected preferences
- Responsiveness
- Inter-user access control, etc.

Attack Vectors
- Malicious user GUI input
- Malicious/invalid input from News server
- Malicious/invalid input from other apps

Unauthenticated state
- Provided XML views
- Account creation on server
- Persistence; password masking

Authenticated state
- Authentication timeout
- File (story) saving, SD card or internal
- Story sharing, story filtering
- Toast message confirmations

Either state
- Toast error messages

- Participants to create an Android-based mobile news application
- 17-page informal specification
Sample Mobile Challenge: News App

- XML UI files determine the layout of graphical elements
- Multiple storage locations for persistent data
- Server interaction
User Interface Behavior
Testing a Mobile App

TCUI VM  Jenkins VM  Host OS

User-submit  Transmit APK  saved
Via SSH, launch news server VM  launch VM
Tell: clone the mobil-1-ping job  clone the mobil-1-ping job
Tell: run the ping job  run the mobil-1-ping job
Tell: run the test job  Run the test job
Run the test job
- checkout the src from gitlab
- compile (java) using maven
- start Android emulator
  (uses Android plugin)
- copy /etc/host into the emulator
- invoke maven to run tests
  (generates raw reports)

Via SSH, kill the news server VM  kill the VM
Retrieve the raw report  Read/send
Modify report for presentation;
generate scores

Computer Security Division
Information Technology Laboratory
# Abstract Measurement Results

<table>
<thead>
<tr>
<th>Reference measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average ~2,600 SLOC for 8 exemplar implementations (not participant submissions).</td>
</tr>
<tr>
<td>Excluding libraries and lib-generated code.</td>
</tr>
</tbody>
</table>

## McCabe Cyclomatic complexity

## Halstead complexity

<table>
<thead>
<tr>
<th>20 Pass/Fail Functional Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass join_table</td>
</tr>
<tr>
<td>Pass list_decks</td>
</tr>
<tr>
<td>Pass take_deck</td>
</tr>
<tr>
<td>Pass release_deck</td>
</tr>
<tr>
<td>Pass shuffle_deck</td>
</tr>
<tr>
<td>Pass start_play</td>
</tr>
<tr>
<td>Pass start_turn</td>
</tr>
<tr>
<td>Pass pop_deck</td>
</tr>
<tr>
<td>Pass take_card</td>
</tr>
<tr>
<td>Pass put_card</td>
</tr>
<tr>
<td>Pass show_hand</td>
</tr>
<tr>
<td>Pass show_table</td>
</tr>
<tr>
<td>Pass save_table</td>
</tr>
<tr>
<td>Pass multiple_players</td>
</tr>
<tr>
<td>Pass search_player</td>
</tr>
<tr>
<td>Pass search_deck</td>
</tr>
<tr>
<td>Fail remove_player</td>
</tr>
<tr>
<td>Pass multiple_decks</td>
</tr>
<tr>
<td>Pass max_players</td>
</tr>
<tr>
<td>Pass history</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>20 Pass/Fail Security Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass authentication</td>
</tr>
<tr>
<td>Pass buffer_error</td>
</tr>
<tr>
<td>Pass code_injection</td>
</tr>
<tr>
<td>Fail format_string</td>
</tr>
<tr>
<td>Pass command_inject</td>
</tr>
<tr>
<td>Pass race_condition</td>
</tr>
<tr>
<td>Pass credential_fail</td>
</tr>
<tr>
<td>Pass input_validation</td>
</tr>
<tr>
<td>Pass numeric_error</td>
</tr>
<tr>
<td>Fail privilege_error</td>
</tr>
<tr>
<td>Pass path_traversal</td>
</tr>
<tr>
<td>Pass link_following</td>
</tr>
<tr>
<td>Pass info_leak</td>
</tr>
<tr>
<td>Pass access_control</td>
</tr>
<tr>
<td>Pass out_of_turn_play</td>
</tr>
<tr>
<td>Pass join_order_used</td>
</tr>
<tr>
<td>Pass invalid_deck_use</td>
</tr>
<tr>
<td>Fail deck_ownership</td>
</tr>
<tr>
<td>Pass card_visibility</td>
</tr>
<tr>
<td>Pass random_order</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fuzz testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>N cpu hours</td>
</tr>
<tr>
<td>C crashes</td>
</tr>
<tr>
<td>H hangs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Submission time</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 10 hours (break ties)</td>
</tr>
</tbody>
</table>

Fuzz testing applied uniformly across submissions.

Indicators on the complexity, or difficulty of the CP.

CP-specific functional tests (score displayed is notional).

Application-specific security tests, categorized when possible using the MITRE Common Weaknesses and Vulnerabilities types.

### Actual Measurement Results: Functional Tests

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Total</th>
<th>Passed</th>
<th>Failed</th>
<th>Total</th>
<th>Passed</th>
<th>Failed</th>
<th>Skipped</th>
<th>Pending</th>
<th>Duration</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1 - Login view is presented</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 secs and 923 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 2 - Add an account as John</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>21 secs and 316 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 3 - Log in as guest</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>19 secs and 468 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 4 - Test the Save credentials</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 min and 870 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 5 - Log in as John</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22 secs and 849 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 6 - Check for proper titles for guest user</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33 secs and 78 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 7 - Check for proper titles for authenticated user</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>37 secs and 222 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 8 - News stories are properly presented</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16 secs and 6 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 9 - Test the Refresh Item</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14 secs and 179 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 10 - Test the Keyword Item</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20 secs and 642 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 11 - Test the Cancel of Keyword Item</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18 secs and 8 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 12 - Test the Story Count Item</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>52 secs and 722 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 13 - Test the Change Location Item</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>35 secs and 586 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 14 - Test the Log Out Item</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20 secs and 394 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 15 - Test story content</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27 secs and 974 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 16 - Test story hyperlinks</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15 secs and 993 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 17 - Test the Save Story to Internal Memory</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14 secs and 854 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 18 - Test the Save Story to External Memory</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14 secs and 672 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 19 - Test the Share Story</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30 secs and 411 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 20 - Test for Unresponsive Backend</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>42 secs and 819 ms</td>
<td>passed</td>
</tr>
</tbody>
</table>

Total: 20 20 0 97 97 0 0 0
### Actual Measurement Results: Security Tests

<table>
<thead>
<tr>
<th>Feature</th>
<th>Scenarios</th>
<th>Steps</th>
<th>Duration</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 21 - Check the android security permissions</td>
<td>1</td>
<td>1</td>
<td>5 secs and 523 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 22 - Attempt to add a new account with invalid username</td>
<td>1</td>
<td>1</td>
<td>58 secs and 582 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 23 - Attempt to add a new account with invalid password</td>
<td>1</td>
<td>1</td>
<td>57 secs and 963 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 24 - Attempt to add a new account with duplicate user</td>
<td>1</td>
<td>1</td>
<td>21 secs and 367 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 25 - Attempt to login with invalid account</td>
<td>1</td>
<td>1</td>
<td>12 secs and 934 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 26 - Test handling of invalid add account data from server</td>
<td>1</td>
<td>1</td>
<td>22 secs and 886 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 27 - Test handling of invalid login data from server</td>
<td>1</td>
<td>1</td>
<td>13 secs and 956 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 28 - Add a valid account2</td>
<td>1</td>
<td>1</td>
<td>21 secs and 224 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 29 - Log in as user2</td>
<td>1</td>
<td>1</td>
<td>23 secs and 623 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 30 - Test handling of invalid story data from server</td>
<td>1</td>
<td>1</td>
<td>45 secs and 161 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 31 - Test session expiration</td>
<td>1</td>
<td>1</td>
<td>2 mins and 19 secs and 975 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 32 - Attempt to enter an invalid keyword</td>
<td>1</td>
<td>1</td>
<td>34 secs and 587 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 33 - Attempt to enter an invalid story count</td>
<td>1</td>
<td>1</td>
<td>34 secs and 634 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 34 - Attempt to enter an invalid zip code</td>
<td>1</td>
<td>1</td>
<td>35 secs and 684 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 35 - Test the session close</td>
<td>1</td>
<td>1</td>
<td>26 secs and 407 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 36 - Test the persistence of account settings for user bob</td>
<td>1</td>
<td>1</td>
<td>1 min and 58 secs and 482 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 37 - Test the persistence of account settings for user john</td>
<td>1</td>
<td>1</td>
<td>1 min and 56 secs and 623 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 38 - Attempt username fuzzing</td>
<td>1</td>
<td>1</td>
<td>2 mins and 708 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 39 - Attempt keyword fuzzing</td>
<td>1</td>
<td>1</td>
<td>4 mins and 49 secs and 759 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 40 - Attempt GUI fuzzing</td>
<td>1</td>
<td>1</td>
<td>1 min and 25 secs and 525 ms</td>
<td>passed</td>
</tr>
</tbody>
</table>

**Known-answer testing**

**Fuzz testing**
Actual Measurement Results: Detailed View

### Invalid Input

<table>
<thead>
<tr>
<th>Feature</th>
<th>Scenarios</th>
<th>Steps</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>@test122</td>
<td>Total: 1, Passed: 1, Failed: 0</td>
<td>Passed: 8, Failed: 8, Skipped: 0</td>
<td>58 secs and 582 ms, passed</td>
</tr>
</tbody>
</table>

View Feature File

@test122

**Scenario:** Attempt to add a new account with invalid username

- *Given* I am on the login screen 5 secs and 663 ms
- And I click the add_account button 2 secs and 946 ms
- And I enter a value in add_user_username of john$3% 3 secs and 278 ms
- And I enter a value in add_user_password of password 2 secs and 417 ms
- And I enter a value in add_user_password_confirm of password 2 secs and 414 ms
- And I enter a value in add_user_zipcode of 33618 1 sec and 639 ms
- And I click the OK button 3 secs and 33 ms
- Then user creation failed with username john$3% and password password 36 secs and 990 ms

### Fuzzing

<table>
<thead>
<tr>
<th>Feature</th>
<th>Scenarios</th>
<th>Steps</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>@test40</td>
<td>Total: 1, Passed: 1, Failed: 0</td>
<td>Passed: 3, Failed: 3, Skipped: 0</td>
<td>1 min and 25 secs and 525 ms, passed</td>
</tr>
</tbody>
</table>

View Feature File

@test40

**Scenario:** Attempt GUI fuzzing

- *Given* I am on the login screen 5 secs and 373 ms
- And I run the google exerciser monkey with 500 events and seed 103 19 secs and 909 ms
- Then the app is responding properly after GUI fuzzing 1 min and 242 ms
Testing Architecture for Dry Run

Design Goals
- Concurrent clients
- Protected scoring
- Mobility

Note: NICs can be bottlenecks due to large submission size (2.5GB for VMs)

Credit: Pic by User:jpp44345 (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons
Virtualized Demo Architecture

- Injected /System/etc/hosts file for Android
  - No Internet dependency
- Stack of interpreters:
  - Java bytecodes
  - MIPS instructions (QEMU emulator)
  - Guest virtual machine
  - Intel OS X base

Credit: Pic by User:jpp44345 (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons
Dry Run Synopsis

- 8 tests
- 12 developers total
- Experience ranging from 2 years to 32 years
- Test 1: no working submission made; networking issue
- Test 2: incomplete submission; networking issues
- Test 3: incomplete submission; networking issues worse
- Test 4: incomplete submission; network functional
- Test 5: submission did not pass tests
- Test 6: no submission (one requirement judged too hard)
- Test 7: more features; Jenkins job misconfiguration
- Test 8: produced deliverable; test suite failure
Lessons Learned

• It is important for teams to be warmed up.
  – Teams should choose languages, frameworks ahead of time
  – Teams should choose revision control systems ahead of time

• Prepared teams are a precondition for measuring toolchain differences.

• Provide more context prior to the testing
  – As much detail as possible without “spilling the beans”

• Provide revision control software/systems

• Provide a trial-run submission portal.

• Stress test the infrastructure prior to a competition.
## Status

### Preparation Phase

<table>
<thead>
<tr>
<th>Start Date</th>
<th>End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct. 1 2014</td>
<td>Sep. 30 2015</td>
</tr>
</tbody>
</table>

- ✔ **Formulate 8 preliminary Challenge Problems**
- ✔ **Document 8 preliminary Challenge Problems**
- ✔ **Implement 8 solutions for Challenge Problems (includes test suites)**
- ✔ **Dry Run competition**
  At NIST for the 8 challenge Problems.
  - Calibrate CP size/difficulty
  - Confirm scoring approach.

### Redesign

<table>
<thead>
<tr>
<th>Start Date</th>
<th>End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug. 1 2016</td>
<td>Dec. 31 2016</td>
</tr>
</tbody>
</table>

- ✔ **Redesign competition testing infrastructure**

### Iteration 1 Competition

<table>
<thead>
<tr>
<th>Start Date</th>
<th>End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan. 1 2017</td>
<td>Sep. 30 2017</td>
</tr>
</tbody>
</table>

- **Dry Run competition (again).**
- **Confirm participation of NSA, DHS, DARPA.**
- **Choose and refine first CP.**
- **Choose venue for competition.**
- **Procure contractor support for competition event.**
- **Perform steps of slide 8 (“an iterative competition...”)**
- **Plan next iteration.**

### Debugging and Documenting

<table>
<thead>
<tr>
<th>Start Date</th>
<th>End Date</th>
</tr>
</thead>
</table>

- In Progress by G2
  - **Debug Competition Infrastructure**
  - **Deliver Updated Documentation**
  - **Deliver Updated Source Code**
Reminder: Objective of Competition

Problem Difficulty

More Secure Software

Reproducible results, technology improvements, public data

Computer Security Division
Information Technology Laboratory
Thank You