A Secure Toolchain Competition

Sep. 9, 2015

Lee Badger
Christopher Johnson
Computer Security Division
NIST

Shawn Webb
G2 Inc.

Carl Landwehr
GWU/LeMoyne College

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>Team</th>
<th>Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIST</td>
<td>Lee Badger, Christopher Johnson, Murugiah Souppaya</td>
<td>G2, Inc.</td>
<td>Daniel Shiplett, Scott Wilson, Shawn Webb</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Roger Chapple, Sean McGinnis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GWU/LeMoyne College</td>
<td>Carl Landwehr</td>
</tr>
<tr>
<td>Provenance</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>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agenda

• Overview and rationale slides.
• A worked example.
• Feedback from a dry run.
• Live Demonstration.
• Status and future plans.
Objective: Secure Software Through Development Toolchain Competitions

Problem Difficulty

- Complexity
- Time allowed

More Secure Software

- Reproducible results
- Technology improvements
- Public data

Competitions:
- Competition 1: 8 flaws
- Competition 2: 0 flaws
- Competition 3: 0 flaws
- Competition 4: 0 flaws
- Competition 5: 0 flaws

Participant  Winner
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*.

Some kinds of mechanisms:

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

Our working definition. Wikipedia has one too.
### 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!

Work

Work

Work

Work

Work

Work

Work

Work

... Solutions

stop!

Competition database

Automated Scoring Day

Test a Solution

Scores

By Pearson Scott Foresman [Public domain], via Wikimedia Commons, gnome icon artists
Start Demo
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

Participant's Solution

Windows
Python
JVM
maven
C++
Linux
Legacy frameworks
OS X
Java
Model Driven Development
Formal Methods

Testing Infrastructure

narrow interface

(Impies 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

## 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

Web figure credit: GPL license from The GNOME Web Browser Developers, wikimedia commons.
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

News server
REST / JSON protocol
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

User-submit
Transmit APK
Via SSH, launch news server VM
Tell: clone the mobil-1-ping job
Tell: run the ping job
Tell: run the test job

Jenkins VM
saved
clone the mobil-1-ping job
run the mobil-1-ping 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)

Host OS
launch VM
kill the VM

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

Reference measurements
Average ~2,600 SLOC for 8 exemplar implementations (not participant submissions).
Excluding libraries and lib-generated code.

McCabe Cyclomatic complexity
Halstead complexity

Indicators on the complexity, or difficulty of the CP.

20 Pass/Fail Functional Tests
- Pass join_table
- Pass list_decks
- Pass take_deck
- Pass release_deck
- Pass shuffle_deck
- Pass start_play
- Pass start_turn
- Pass pop_deck
- Pass take_card
- Pass put_card
- Pass show_hand
- Pass show_table
- Pass save_table
- Pass multiple_players
- Pass search_player
- Pass search_deck
- Fail remove_player
- Pass multiple_decks
- Pass max_players
- Pass history

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

20 Pass/Fail Security Tests
- Pass authentication
- Pass buffer_error
- Pass code_injection
- Fail format_string
- Pass command_inject
- Pass race_condition
- Pass credential_fail
- Pass input_validation
- Pass numeric_error
- Fail privilege_error
- Pass path_traversal
- Pass link_following
- Pass info_leak
- Pass access_control
- Pass out_of_turn_play
- Pass join_order_used
- Pass invalid_deck_use
- Fail deck_ownership
- Pass card_visibility
- Pass random_order

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

Fuzz testing
- N cpu hours
- C crashes
- H hangs

Submission time
<= 10 hours (break ties)

Fuzz testing applied uniformly across submissions.

### Actual Measurement Results: Functional Tests

**OVERVIEW**

- **Challenge Name:** ANDROID-01 - News App
- **Participant:** User
- **Submission File:** mobile-01.apk
- **Submission MDSSUM:** 81fb1dbf1f51772b45ea3
- **Submission Size:** 2,484,174 bytes
- **Test Start:** 08/31/2015 14:55:38
- **Test Duration:** 34 minutes and 34 seconds
- **Test Score:** 40/40
- **Functional Test Score:** 20/20
- **Security Test Score:** 20/20

| Feature | Scenarios | Steps |  |  |  |  |  |  |  |
|---------|-----------|-------|---|---|---|---|---|---|
|         | Total     | Passed | Failed | Total | Passed | Failed | Skipped | Pending | Duration | Status |
| Test 1 - Login view is presented | 1 | 1 | 0 | 2 | 2 | 0 | 0 | 0 | 5 secs and 923 ms | passed |
| Test 2 - Add an account as John | 1 | 1 | 0 | 8 | 8 | 0 | 0 | 0 | 21 secs and 316 ms | passed |
| Test 3 - Log in as guest | 1 | 1 | 0 | 3 | 3 | 0 | 0 | 0 | 19 secs and 486 ms | passed |
| Test 4 - Test the Save credentials | 1 | 1 | 0 | 7 | 7 | 0 | 0 | 0 | 1 min and 870 ms | passed |
| Test 5 - Log in as John | 1 | 1 | 0 | 5 | 5 | 0 | 0 | 0 | 22 secs and 849 ms | passed |
| Test 6 - Check for proper titles for guest user | 1 | 1 | 0 | 5 | 5 | 0 | 0 | 0 | 33 secs and 78 ms | passed |
| Test 7 - Check for proper titles for authenticated user | 1 | 1 | 0 | 7 | 7 | 0 | 0 | 0 | 37 secs and 222 ms | passed |
| Test 8 - News stories are properly presented | 1 | 1 | 0 | 3 | 3 | 0 | 0 | 0 | 16 secs and 6 ms | passed |
| Test 9 - Test the Refresh Item | 1 | 1 | 0 | 3 | 3 | 0 | 0 | 0 | 14 secs and 179 ms | passed |
| Test 10 - Test the Keyword Item | 1 | 1 | 0 | 5 | 5 | 0 | 0 | 0 | 20 secs and 642 ms | passed |
| Test 11 - Test the Cancel of Keyword Item | 1 | 1 | 0 | 4 | 4 | 0 | 0 | 0 | 18 secs and 8 ms | passed |
| Test 12 - Test the Story Count Item | 1 | 1 | 0 | 9 | 9 | 0 | 0 | 0 | 52 secs and 722 ms | passed |
| Test 13 - Test the Change Location Item | 1 | 1 | 0 | 6 | 6 | 0 | 0 | 0 | 35 secs and 586 ms | passed |
| Test 14 - Test the Log Out Item | 1 | 1 | 0 | 3 | 3 | 0 | 0 | 0 | 29 secs and 394 ms | passed |
| Test 15 - Test story content | 1 | 1 | 0 | 7 | 7 | 0 | 0 | 0 | 27 secs and 974 ms | passed |
| Test 16 - Test story hyperlinks | 1 | 1 | 0 | 3 | 3 | 0 | 0 | 0 | 15 secs and 993 ms | passed |
| Test 17 - Test the Save Story to Internal Memory | 1 | 1 | 0 | 4 | 4 | 0 | 0 | 0 | 14 secs and 854 ms | passed |
| Test 18 - Test the Save Story to External Memory | 1 | 1 | 0 | 4 | 4 | 0 | 0 | 0 | 14 secs and 672 ms | passed |
| Test 19 - Test the Share Story | 1 | 1 | 0 | 4 | 4 | 0 | 0 | 0 | 30 secs and 411 ms | passed |
| Test 20 - Test for Unresponsive Backend | 1 | 1 | 0 | 5 | 5 | 0 | 0 | 0 | 42 secs and 819 ms | passed |

**Summary**

- **Total Scenarios:** 20
- **Total Passed:** 20
- **Total Failed:** 0
- **Total Skipped:** 0
- **Total Duration:** 97 secs
- **Total Status:** passed
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>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>2 secs 523 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 22 - Attempt to add a new account with invalid username</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>58 secs 582 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 23 - Attempt to add a new account with invalid password</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>57 secs 963 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 24 - Attempt to add a new account with duplicate user</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>21 secs 387 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 25 - Attempt to login with invalid account</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>12 secs 934 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 26 - Test handling of invalid add account data from server</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>22 secs 686 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 27 - Test handling of invalid login data from server</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>13 secs 595 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 28 - Add a valid account2</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>21 secs 224 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 29 - Log in as user2</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>23 secs 621 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 30 - Test handling of invalid story data from server</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>45 secs 161 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 31 - Test session expiration</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>2 mins 19 secs and 975 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 32 - Attempt to enter an invalid keyword</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>34 secs 587 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 33 - Attempt to enter an invalid story count</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>34 secs 634 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 34 - Attempt to enter an invalid zip code</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>35 secs 684 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 35 - Test the session close</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>26 secs 407 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 36 - Test the persistence of account settings for user bob</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>1 min 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>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>1 min 56 secs and 623 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 38 - Attempt username fuzzing</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>2 mins 708 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 39 - Attempt keyword fuzzing</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</td>
<td>4 mins 49 secs and 739 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Test 40 - Attempt GUI fuzzing</td>
<td>Total: 1</td>
<td>Passed: 1, Failed: 0</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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Scenarios</th>
<th>Steps</th>
<th>Duration</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Passed</td>
<td>Failed</td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

Invalid Input

View Feature File

@test:22

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$$% 3 secs and 278 ms
- And I enter a value in add_user_password of password 2 secs and 417 ms
- And I enter a value in add_user_password_confirm of password 2 secs and 414 ms
- And I enter a value in add_user_zipcode of 33618 1 sec and 839 ms
- And I click the OK button 3 secs and 33 ms
Then user creation failed with username john$$% and password password 36 secs and 990 ms

Cucumber scenarios

Fuzzing

View Feature File

@test:40

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
Improved Testing Architecture

Locking issues for NICs avoided, but memory pressure still an issue.

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 (here at the CIF)

- 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
Resume Demo
Dry Run Synopsis

- 8 tests
- 12 developers total
- Experience ranging from 2 years to 32 years
- Test1: no working submission made; networking issue
- Test2: incomplete submission; networking issues
- Test3: incomplete submission; networking issues worse
- Test4: incomplete submission; network functional
- Test5: submission did not pass tests
- Test6: no submission (one requirement judged too hard)
- Test7: more features; Jenkins job misconfiguration
- Test8: 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
Anticipated Impact of Competition

Problem Difficulty
\( \frac{CP \text{ complexity}}{time \text{ allowed}} \)

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Participant Flaws</th>
<th>Winner Flaws</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Iteration 5</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

Reproducible results, technology improvements, public data

More Secure Software
### Status

#### Preparation Phase

<table>
<thead>
<tr>
<th>Oct. 1 2014</th>
<th>Sep. 30 2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Formulate 8</strong> preliminary Challenge Problems</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Document 8</strong> preliminary Challenge Problems</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Implement 8</strong> solutions for Challenge Problems (includes test suites)</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
| **Simulate competition** At NIST for the 8 challenge Problems.  
  • Calibrate CP size/difficulty  
  • Confirm scoring approach. | |

#### Iteration 1 Competition

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Re-engineer competition testing infrastructure</strong></td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Second competition simulation</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Confirm participation of NSA, DHS, DARPA.</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Choose and refine first CP.</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Choose venue for competition.</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Procure contractor support for competition event.</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Perform steps of slide 9 (&quot;an iterative competition...&quot;)</td>
<td></td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Plan iteration 2 competition.</td>
<td></td>
</tr>
</tbody>
</table>
Thank You