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.

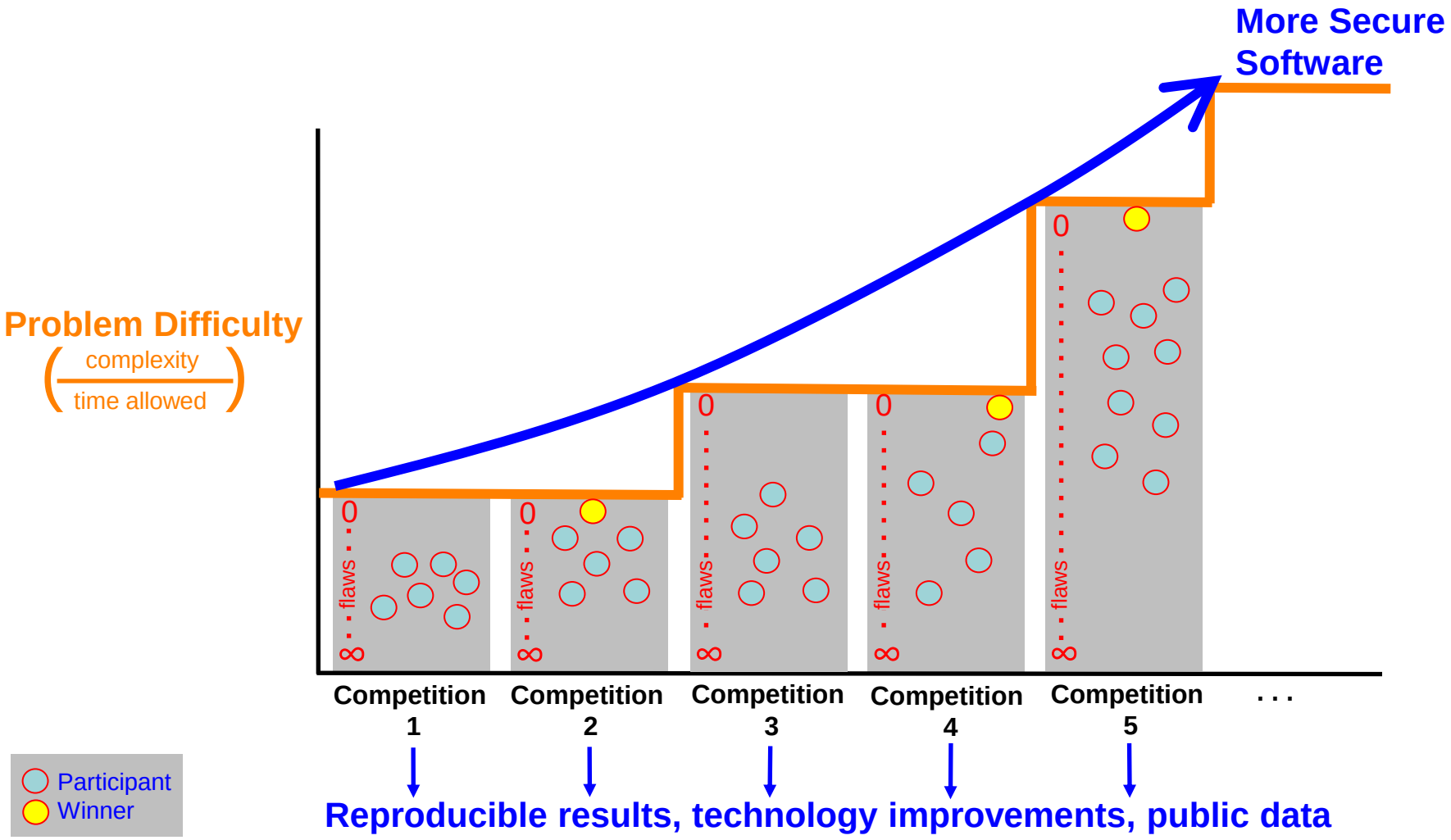
Team and Idea Provenance

NIST	Lee Badger Christopher Johnson Murugiah Souppaya	Larry Keys Michael Bartock Jeffrey Cichonski
G2, Inc.	Daniel Shiplett Scott Wilson Shawn Webb	Roger Chapple Sean McGinnis
GWU/LeMoyne College	Carl Landwehr	
Provenance	Based on an idea from Designing a Secure Systems Engineering Competition (DESSEC) workshop run by NSF in 2010: Secure Development Tool Chain	

Agenda

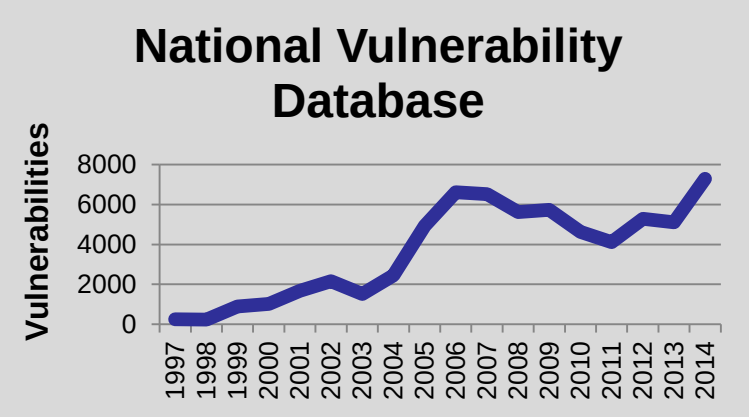
- Overview and rationale slides.
- A worked example.
- Feedback from a dry run.
- Live Demonstration.
- Status and future plans.

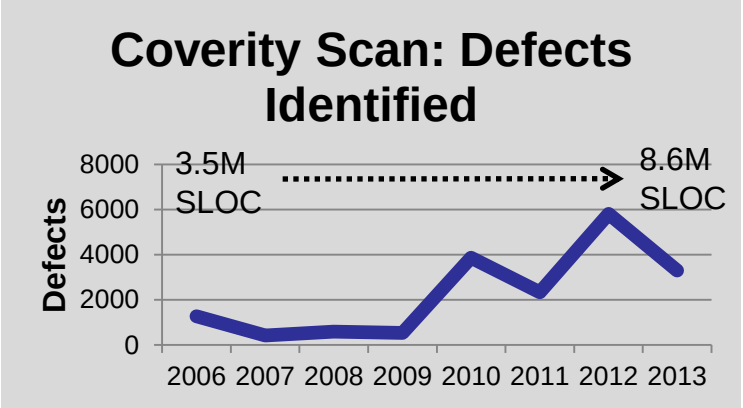
Objective: Secure Software Through Development Toolchain Competitions

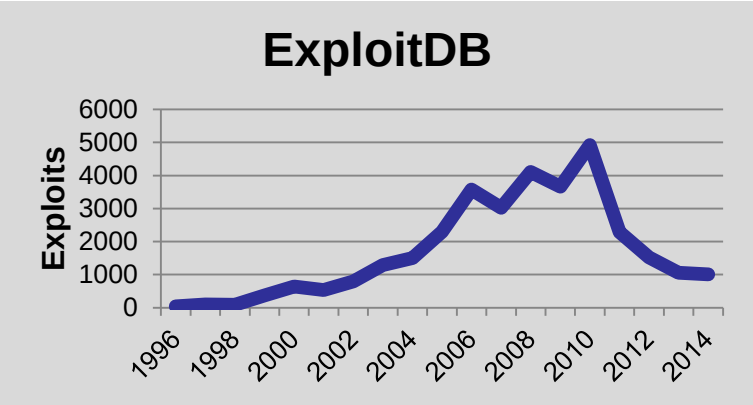


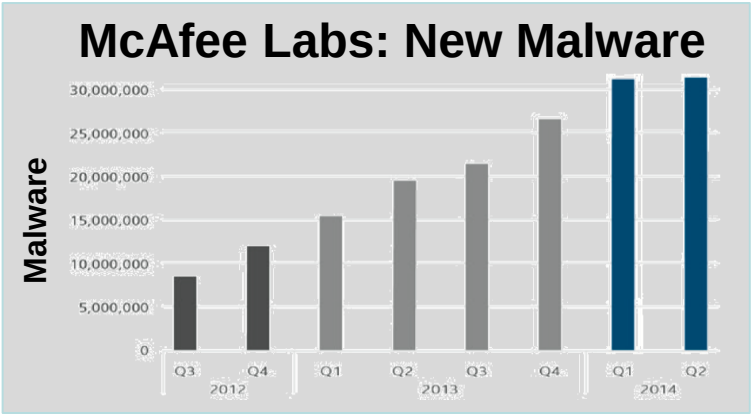
The Problem

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



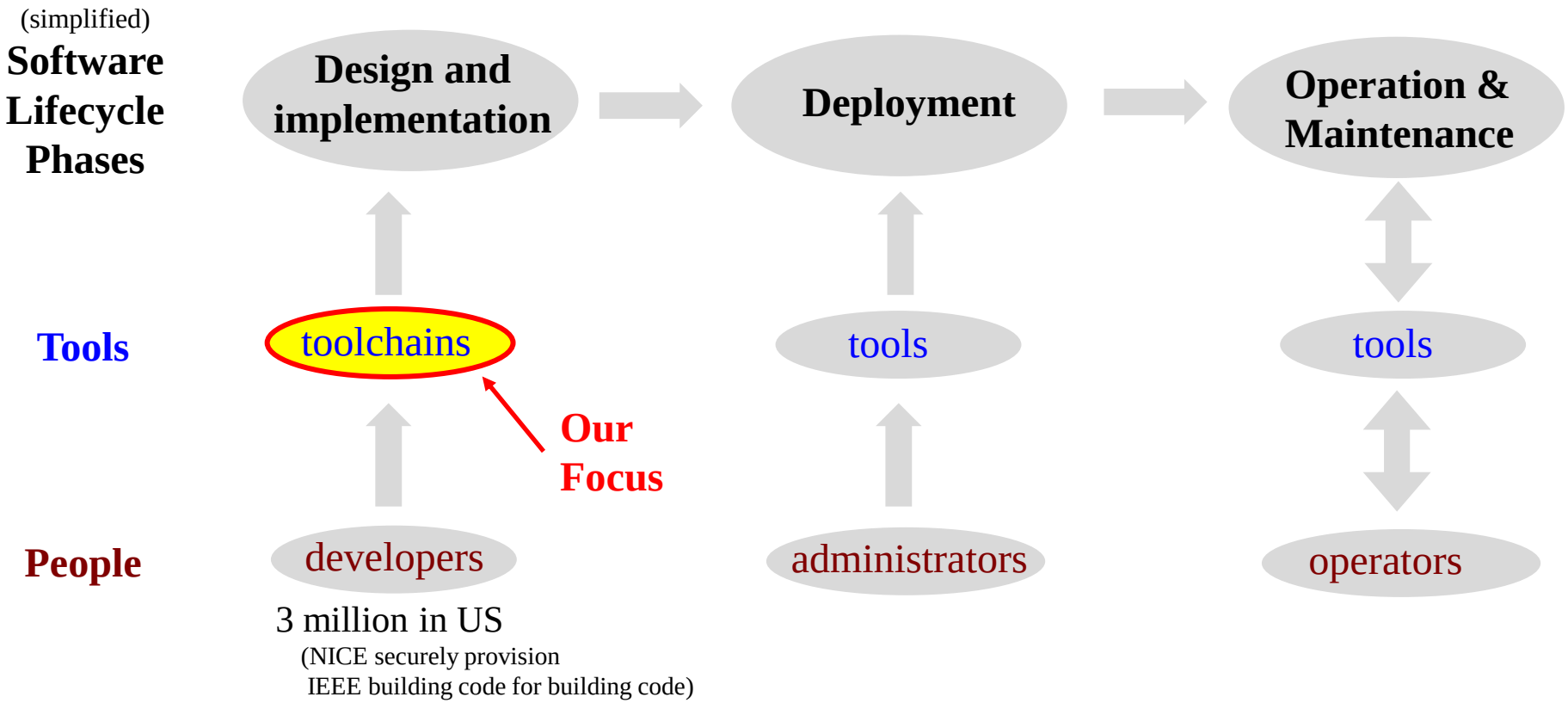






Credit: nvd.nist.gov, www.exploit-db.com, www.coverity.com, McAfee Labs, 2014.

Opportunities for Vulnerability Suppression/Mitigation



- 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**
- **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**

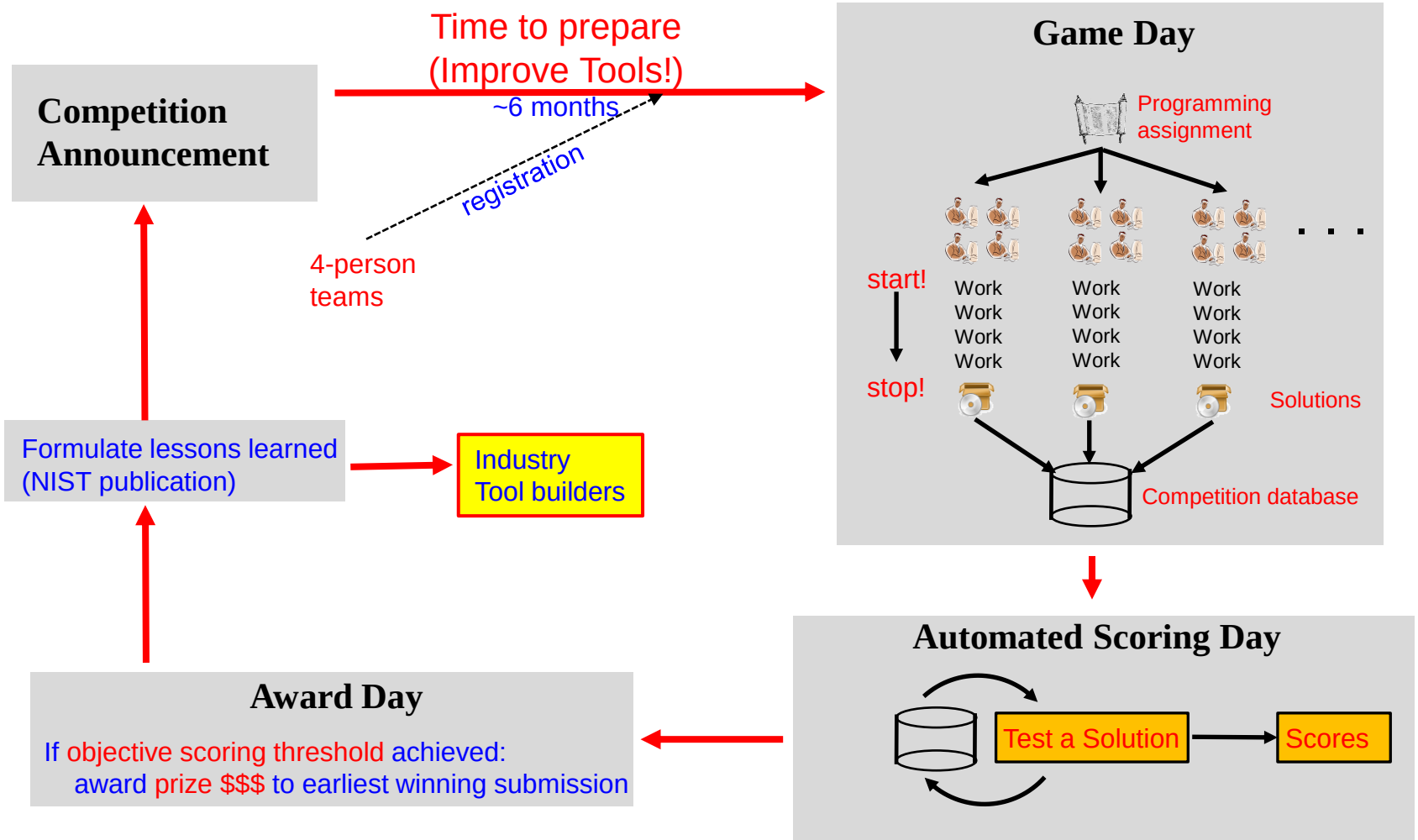
Some Toolchain Platforms

Android	**iOS**	**Blackberry**	**MS Windows Version X**	**OS X**	**Linux**
Solaris	**Java Virtual Machines**	**MS .Net**	**Adobe Flash**	**Web Browser (e.g., ajax)**	**Arduino**
Embeded	**App X Loadable Modules**	**OS command line**			

and many more …

- Improvements could reduce vulnerability production.
- But, **how can we incentivize security improvements?**

An Iterative Competition to Foster Improved Software Toolchains

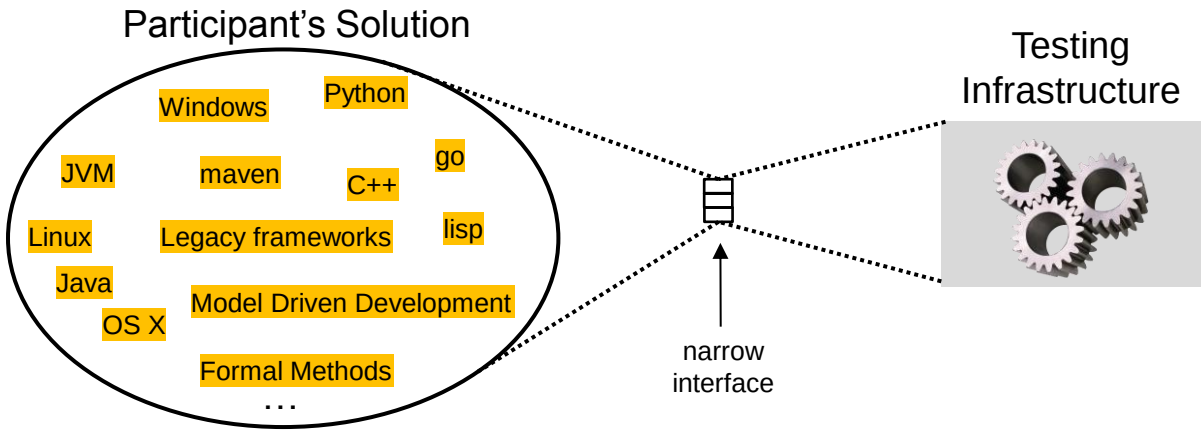


By Pearson Scott Foresman [Public domain], via Wikimedia Commons, gnome icon artists

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



(Implies 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)

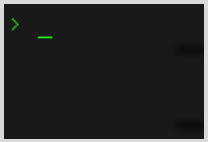
   20 fully-automated application-specific pass/fail functional tests.

   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



(2)

- **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)

(3)

- **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

(3)

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

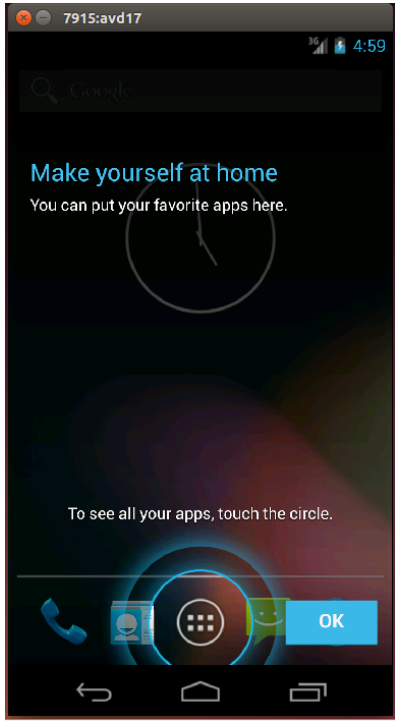
- Provided XML views
- Account creation on server
- Persistence; password masking

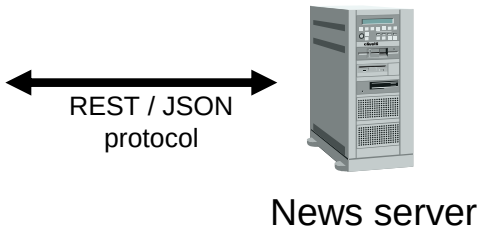
Authenticated state

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

Either state

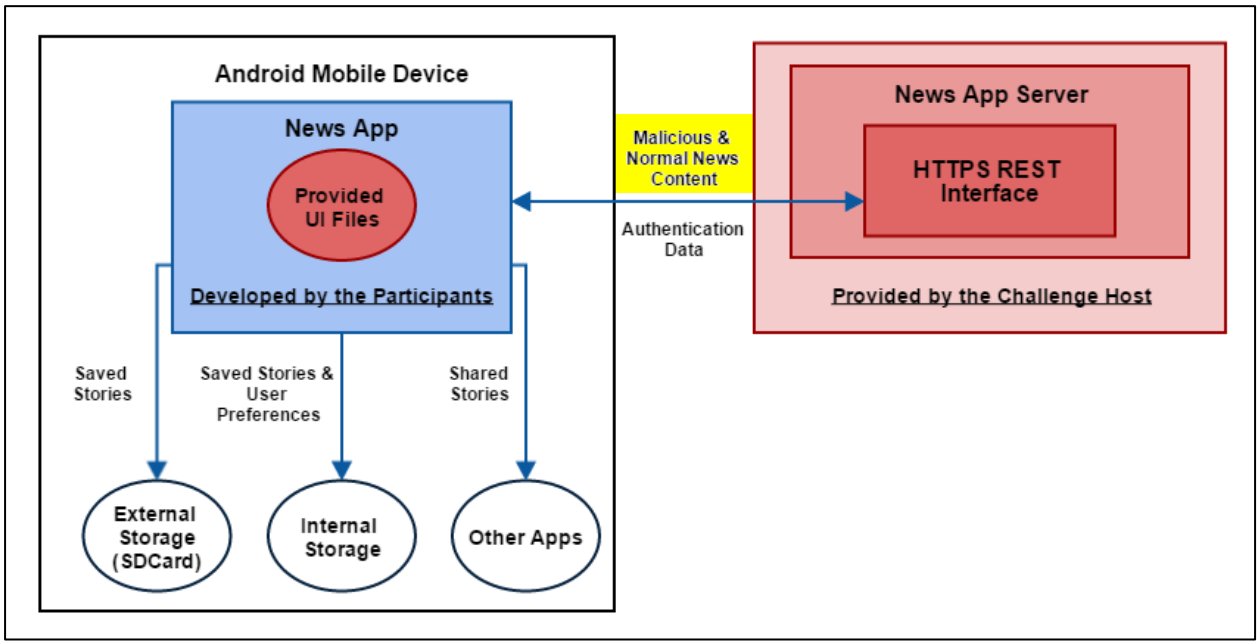
- Toast error messages





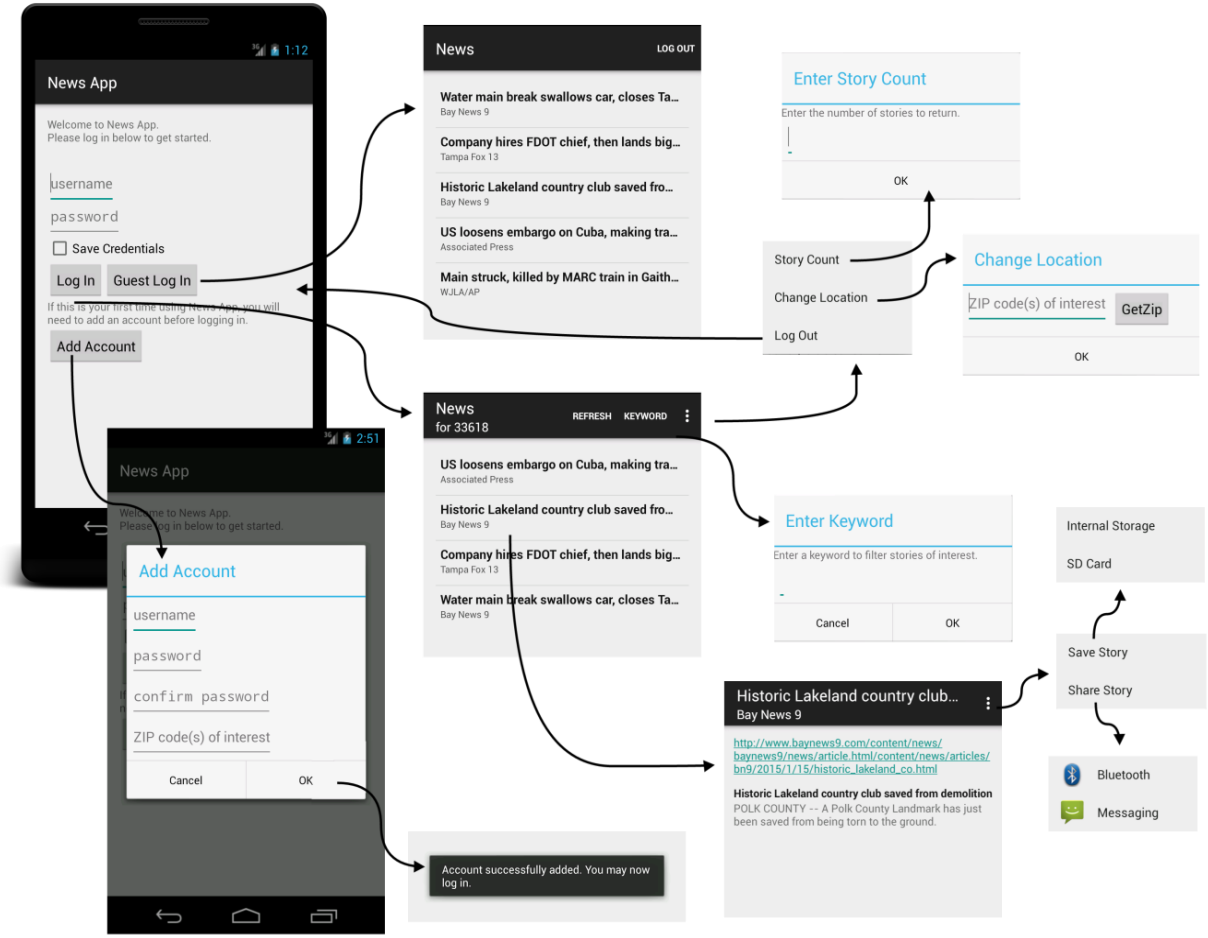
- Participants to create an Android-based mobile news application
- 17-page informal specification

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
- 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

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	numberic_error
Fail	privilage_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.

Credit: http://cwe.mitre.org/data/slices/2000.html

Fuzz testing

N cpu hours
C crashes
H hangs

Fuzz testing applied uniformly across submissions.

Submission time

<= 10 hours

(break ties)

Actual Measurement Results: Functional Tests

OVERVIEW

Challenge Name:ANDROID-01 - News App
Participant: User
Submission File:mobile-01.apk
Submission MD5SUM: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

	Scenarios			Steps						
Feature	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 488 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
	20	**20**	**0**	**97**	**97**	**0**	**0**	**0**		

Actual Measurement Results: Security Tests

	Scenarios			Steps						
Feature	Total	Passed	Failed	Total	Passed	Failed	Skipped	Pending	Duration	Status
Test 21 - Check the android security permissions	1	1	0	2	2	0	0	0	5 secs and 523 ms	passed
Test 22 - Attempt to add a new account with invalid username	1	1	0	8	8	0	0	0	58 secs and 582 ms	passed
Test 23 - Attempt to add a new account with invalid password	1	1	0	8	8	0	0	0	57 secs and 963 ms	passed
Test 24 - Attempt to add a new account with duplicate user	1	1	0	8	8	0	0	0	21 secs and 387 ms	passed
Test 25 - Attempt to login with invalid account	1	1	0	5	5	0	0	0	12 secs and 934 ms	passed
Test 26 - Test handling of invalid add account data from server	1	1	0	8	8	0	0	0	22 secs and 686 ms	passed
Test 27 - Test handling of invalid login data from server	1	1	0	5	5	0	0	0	13 secs and 595 ms	passed
Test 28 - Add a valid account2	1	1	0	8	8	0	0	0	21 secs and 224 ms	passed
Test 29 - Log in as user2	1	1	0	5	5	0	0	0	23 secs and 621 ms	passed
Test 30 - Test handling of invalid story data from server	1	1	0	12	12	0	0	0	45 secs and 161 ms	passed
Test 31 - Test session expiration	1	1	0	4	4	0	0	0	2 mins and 19 secs and 975 ms	passed
Test 32 - Attempt to enter an invalid keyword	1	1	0	9	9	0	0	0	34 secs and 587 ms	passed
Test 33 - Attempt to enter an invalid story count	1	1	0	5	5	0	0	0	34 secs and 634 ms	passed
Test 34 - Attempt to enter an invalid zip code	1	1	0	5	5	0	0	0	35 secs and 684 ms	passed
Test 35 - Test the session close	1	1	0	3	3	0	0	0	26 secs and 407 ms	passed
Test 36 - Test the persistence of account settings for user bob	1	1	0	18	18	0	0	0	1 min and 58 secs and 482 ms	passed
Test 37 - Test the persistence of account settings for user john	1	1	0	18	18	0	0	0	1 min and 56 secs and 623 ms	passed
Test 38 - Attempt username fuzzing	1	1	0	4	4	0	0	0	2 mins and 708 ms	passed
Test 39 - Attempt keyword fuzzing	1	1	0	18	18	0	0	0	4 mins and 49 secs and 739 ms	passed
Test 40 - Attempt GUI fuzzing	1	1	0	3	3	0	0	0	1 min and 25 secs and 525 ms	passed
	20	20	0	156	156	0	0	0		

Known-answer testing

Fuzz testing

Actual Measurement Results: Detailed View

Cucumber scenarios

	Scenarios			Steps						
Feature	Total	Passed	Failed	Total	Passed	Failed	Skipped	Pending	Duration	Status
@test22	1	1	0	8	8	0	0	0	58 secs and 582 ms	passed

Invalid Input

View Feature File

@test22

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

	Scenarios			Steps						
Feature	Total	Passed	Failed	Total	Passed	Failed	Skipped	Pending	Duration	Status
@test40	1	1	0	3	3	0	0	0	1 min and 25 secs and 525 ms	passed

Fuzzing

View Feature File

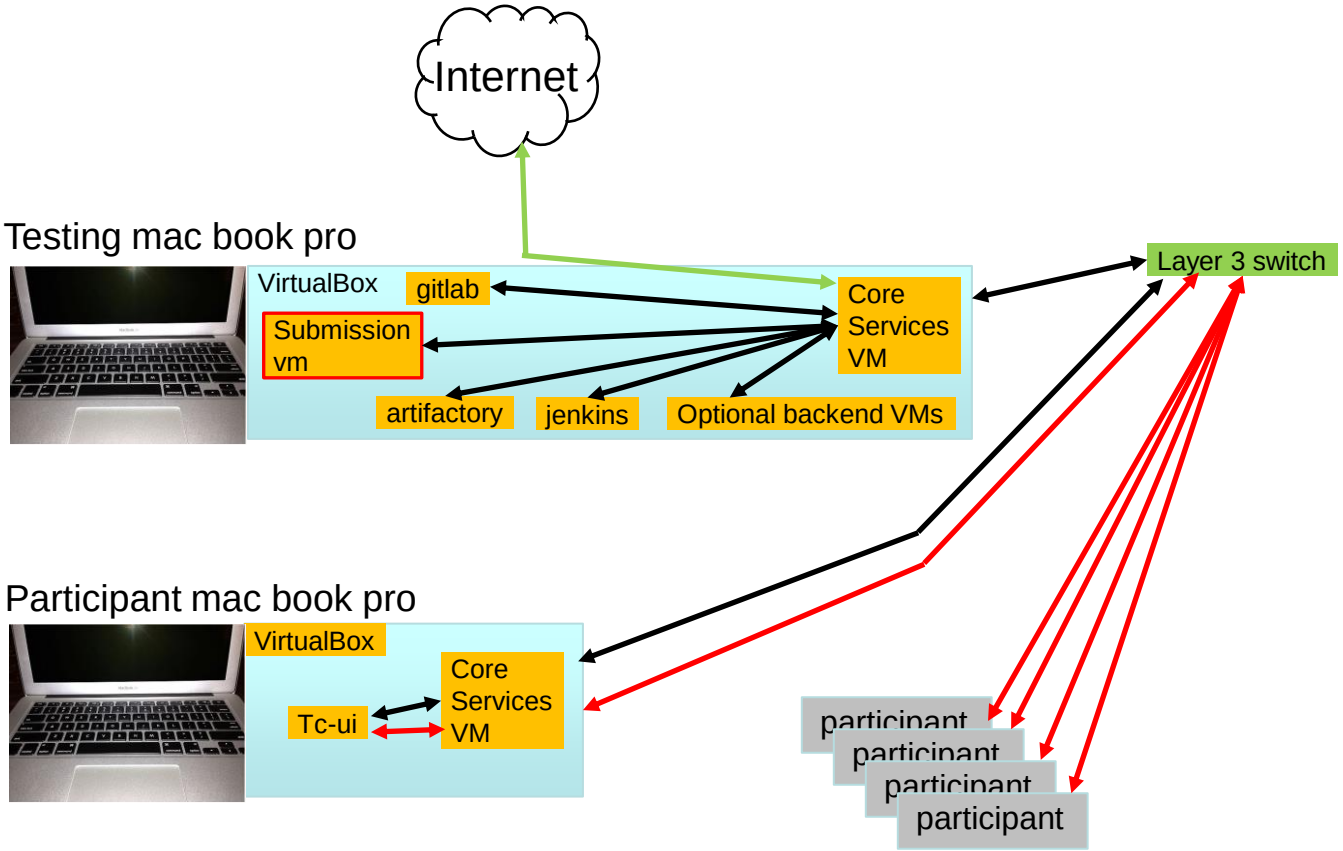
@test40

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

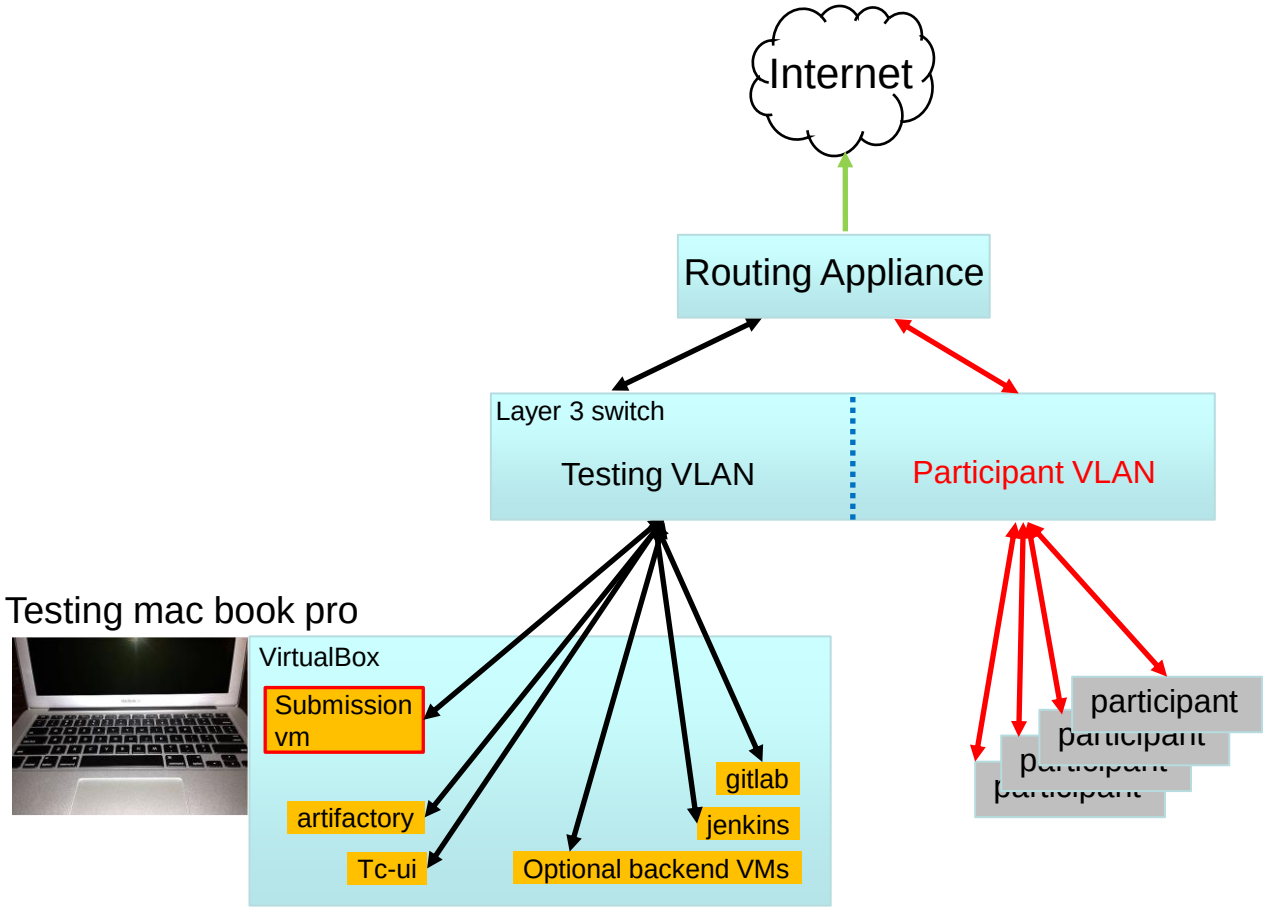
- Concurrent clients
- Protected scoring
- Mobility



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

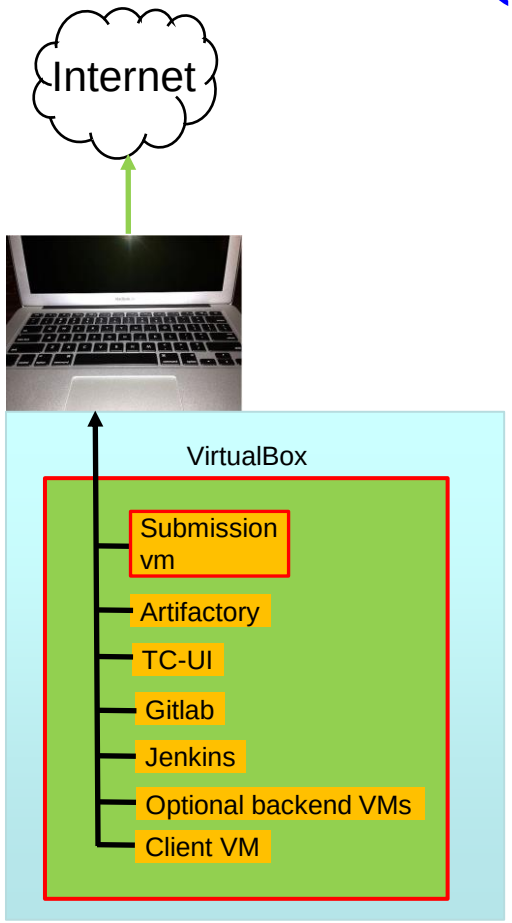
Improved Testing Architecture



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

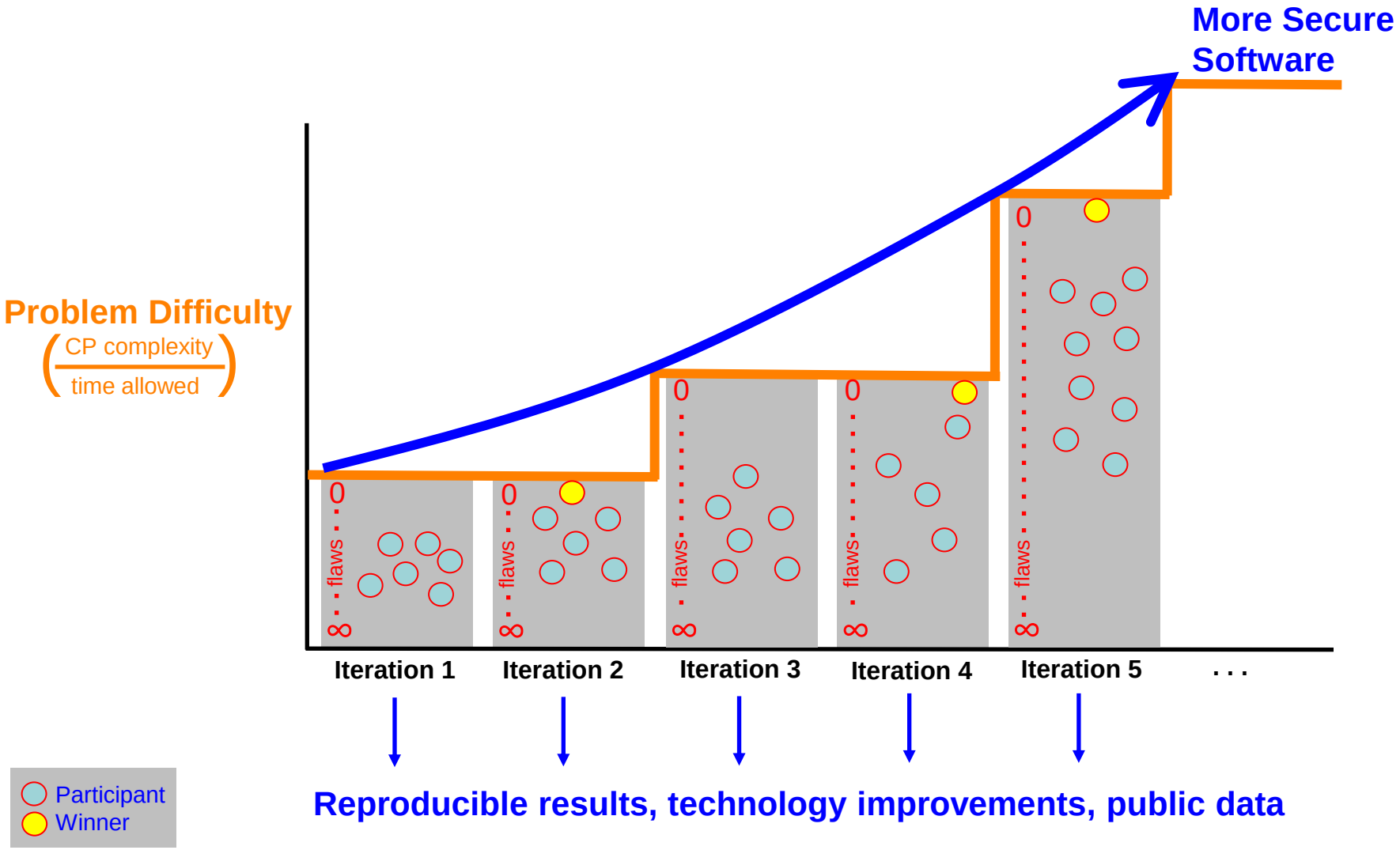
Dry Run Synopsis

- 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



Status

Preparation Phase

Oct. 1 2014 — Sep. 30 2015

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

Iteration 1 Competition

Oct. 1 2015 — Sep. 30 2016

- **Re-engineer competition testing infrastructure**
- **Second competition simulation**
- **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 9 (“an iterative competition...”)**
- **Plan iteration 2 competition.**

Thank You