

Vulnerability Discovery

David Warren



Copyright 2016 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon®, CERT® and CERT Coordination Center® are registered marks of Carnegie Mellon University.

DM-0004078

Vulnerability Discovery Project



Increase **assurance** of DoD software through
enhanced vulnerability discovery techniques

Team



- Edward Schwartz, PhD, CERT
- David Warren, CERT
- Allen Householder, CERT



- David Brumley, PhD
- Thanassis Avgerinos, PhD
- Tyler Nighswander

Agenda

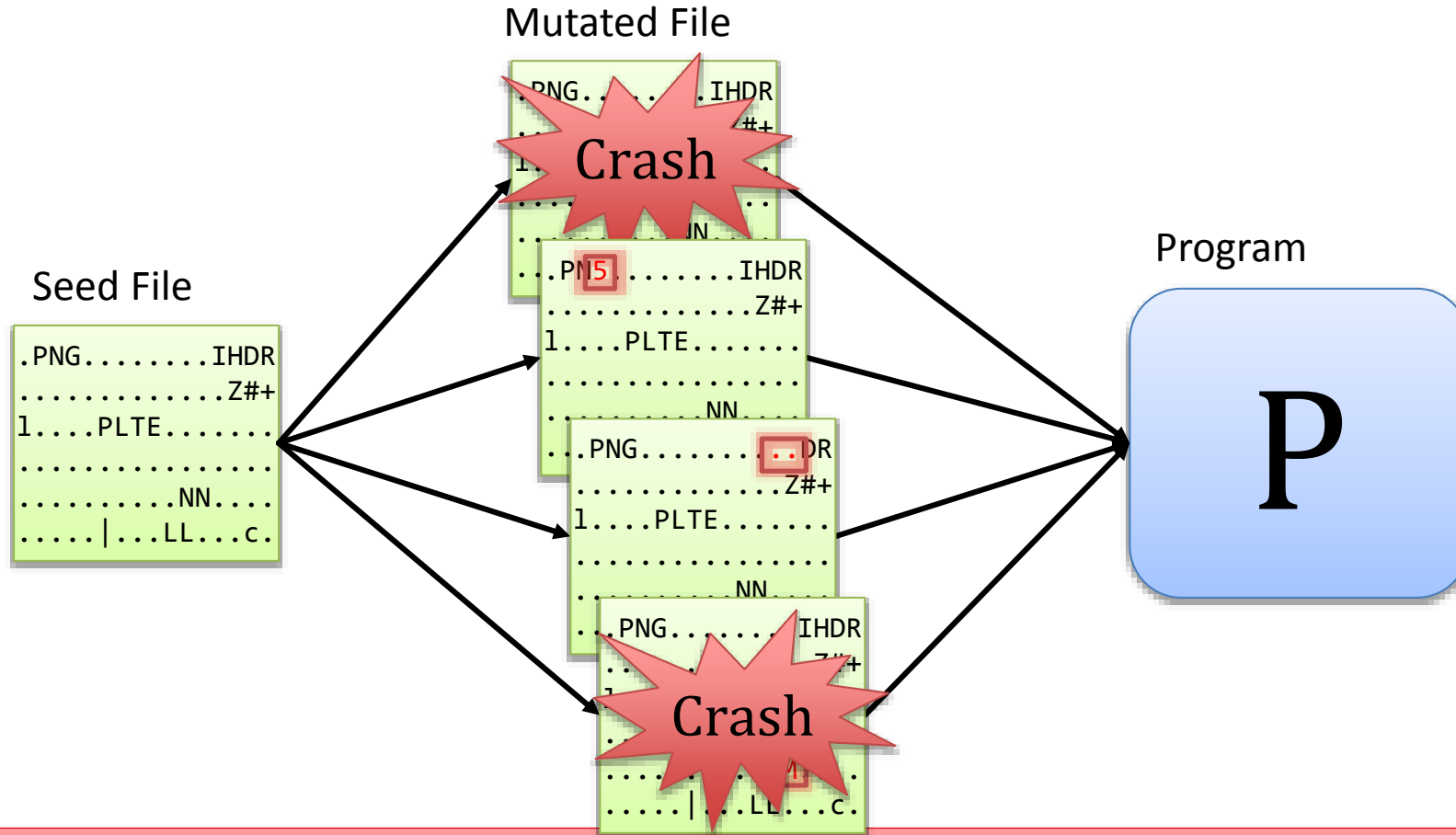


**Revisiting Widely Held Beliefs
about Black-Box Fuzzing**

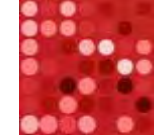
**SMART Fuzzing: How to
Intelligently Combine a Fuzzer
with a Concolic Executor**

Revisiting Widely Held Beliefs About Black-Box Fuzzing

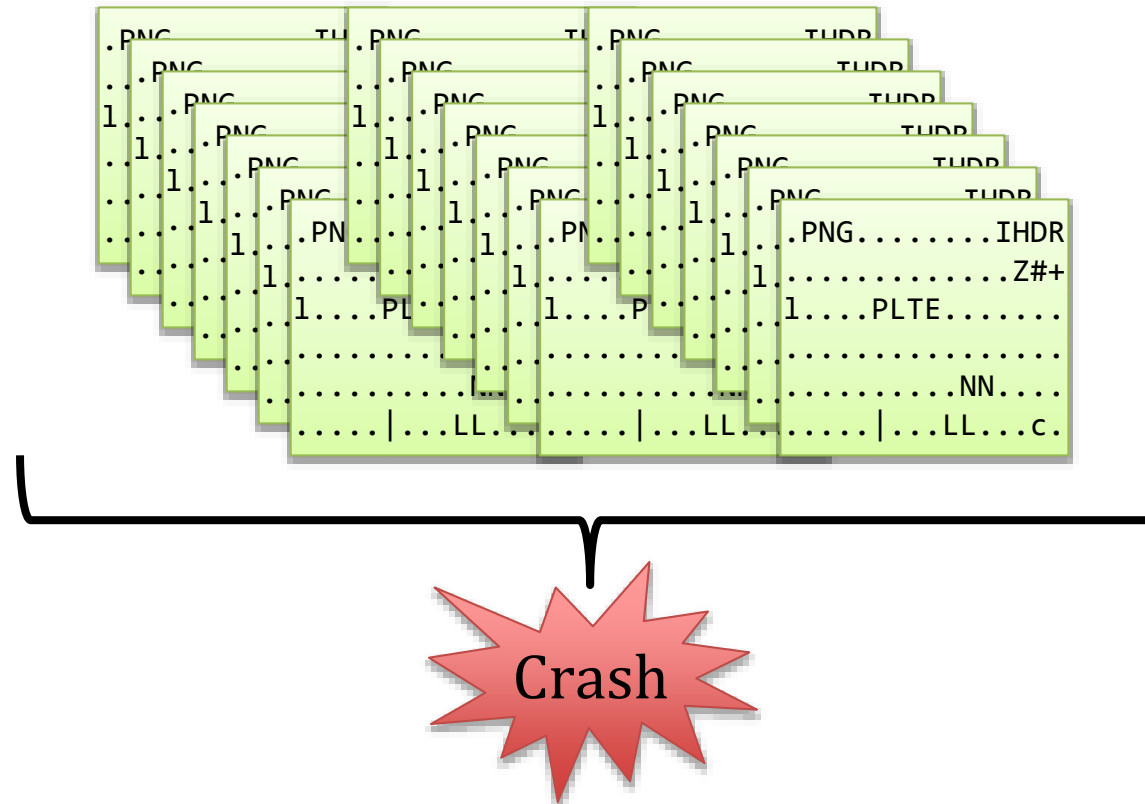
Background: Mutational Fuzzing of Software



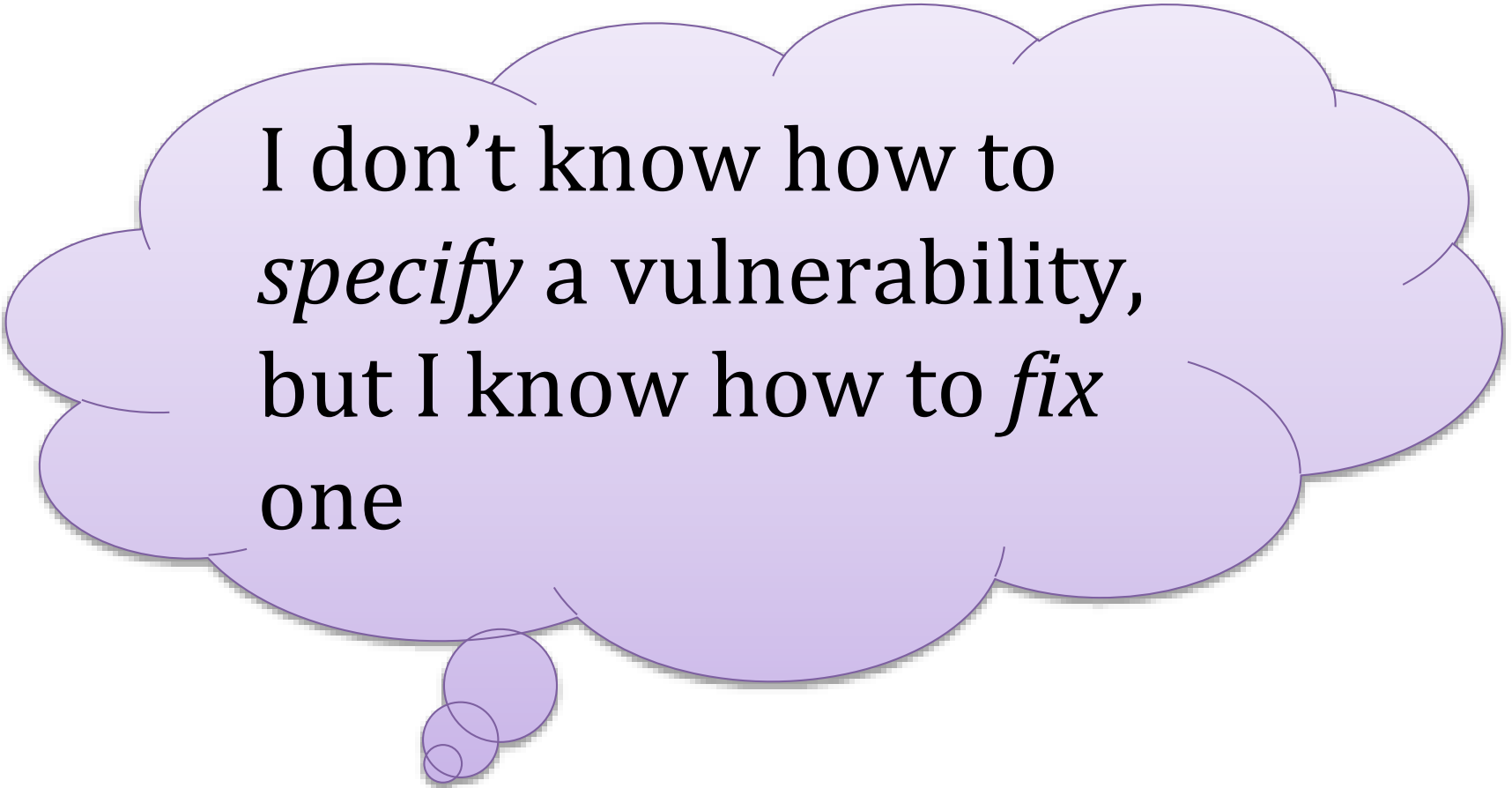
Testing of programs by randomly mutating program inputs (seeds)



Challenge: How Many Software Vulnerabilities Are There?



Problem: Distinguishing One Vulnerability From Another



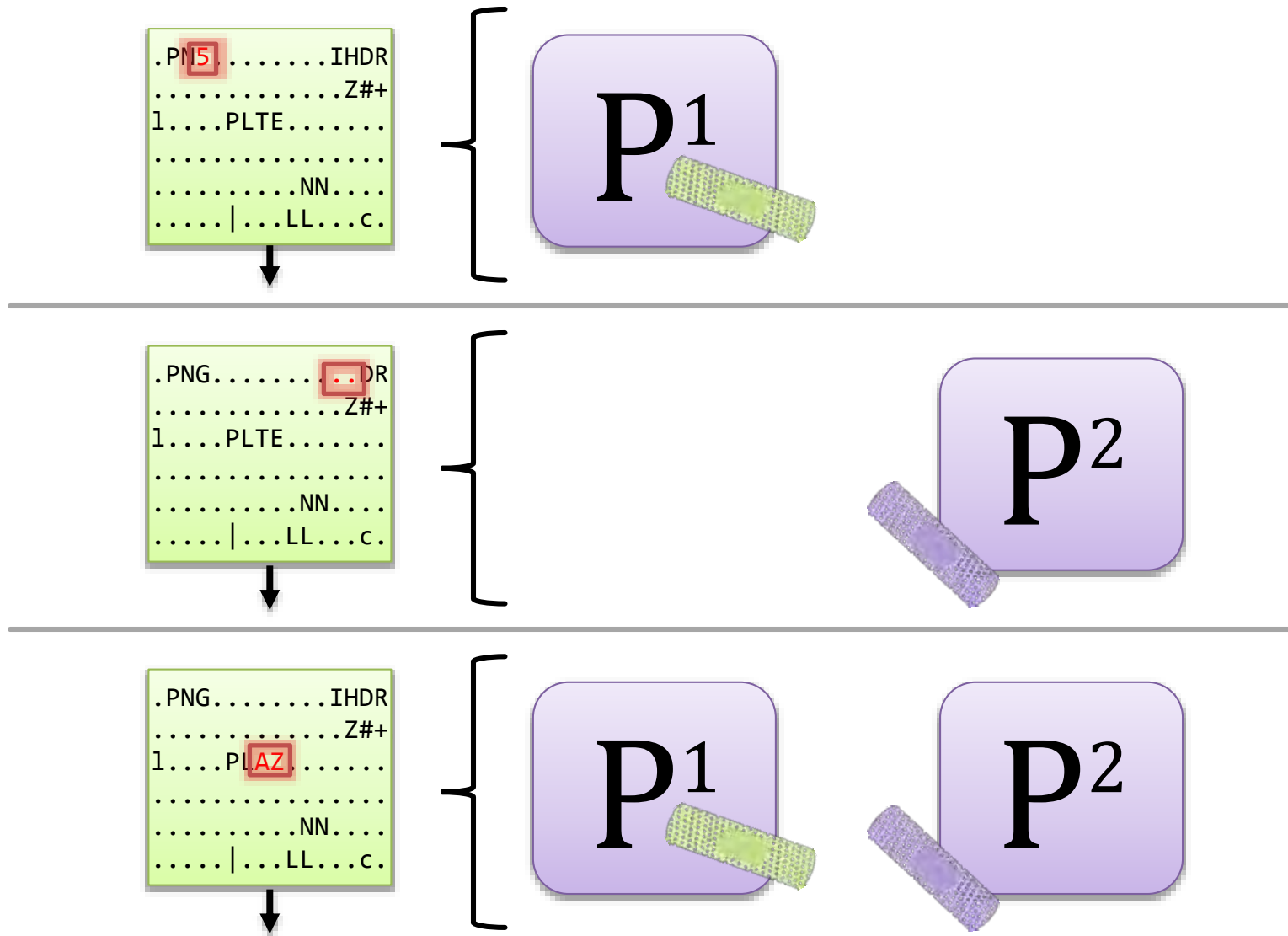
I don't know how to
specify a vulnerability,
but I know how to *fix*
one

The Idea: Patches Define Vulnerabilities



Any crash that is fixed by the patch is also affected by vulnerability V

Example Ground Truth



Patching software

- Fuzzed each program for 1 week with BFF fuzzer
- Manually patched all vulnerabilities to collect ground truth

	Flasm	ImageMagick	Jasper	OpenJpeg
Crashes (BFF)	253	64	93	145
Vulnerabilities	6	31	12	36

Revisiting Common Beliefs about Black-Box Fuzzing

Belief 1: Stack backtrace hashing always accurately counts vulnerabilities.

- Used by BFF and other fuzzers.

Belief 2: Sanitization never harms fuzzing performance.

- Detects vulnerabilities that do not cause a crash.

Belief 3: The AFL fuzzer always finds more vulnerabilities than non-guided fuzzers.

- Newer is better.

Belief 1: Stack Backtrace Hashing Always Accurately Counts Vulnerabilities

Undercount (UC)

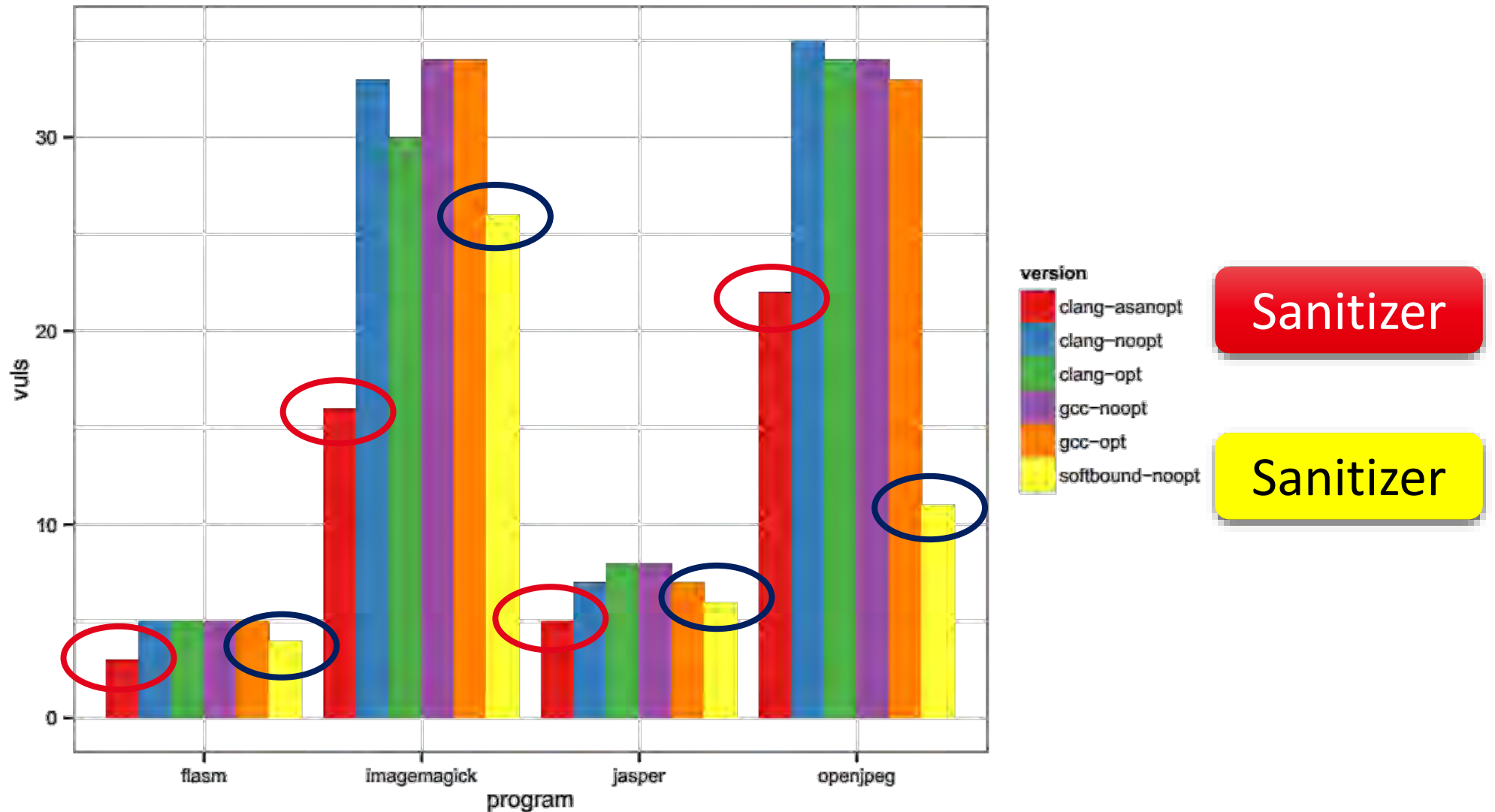
- # vulnerabilities missed by stack backtrace hashing on average
- We never see these vulnerabilities

Overcount (OC)

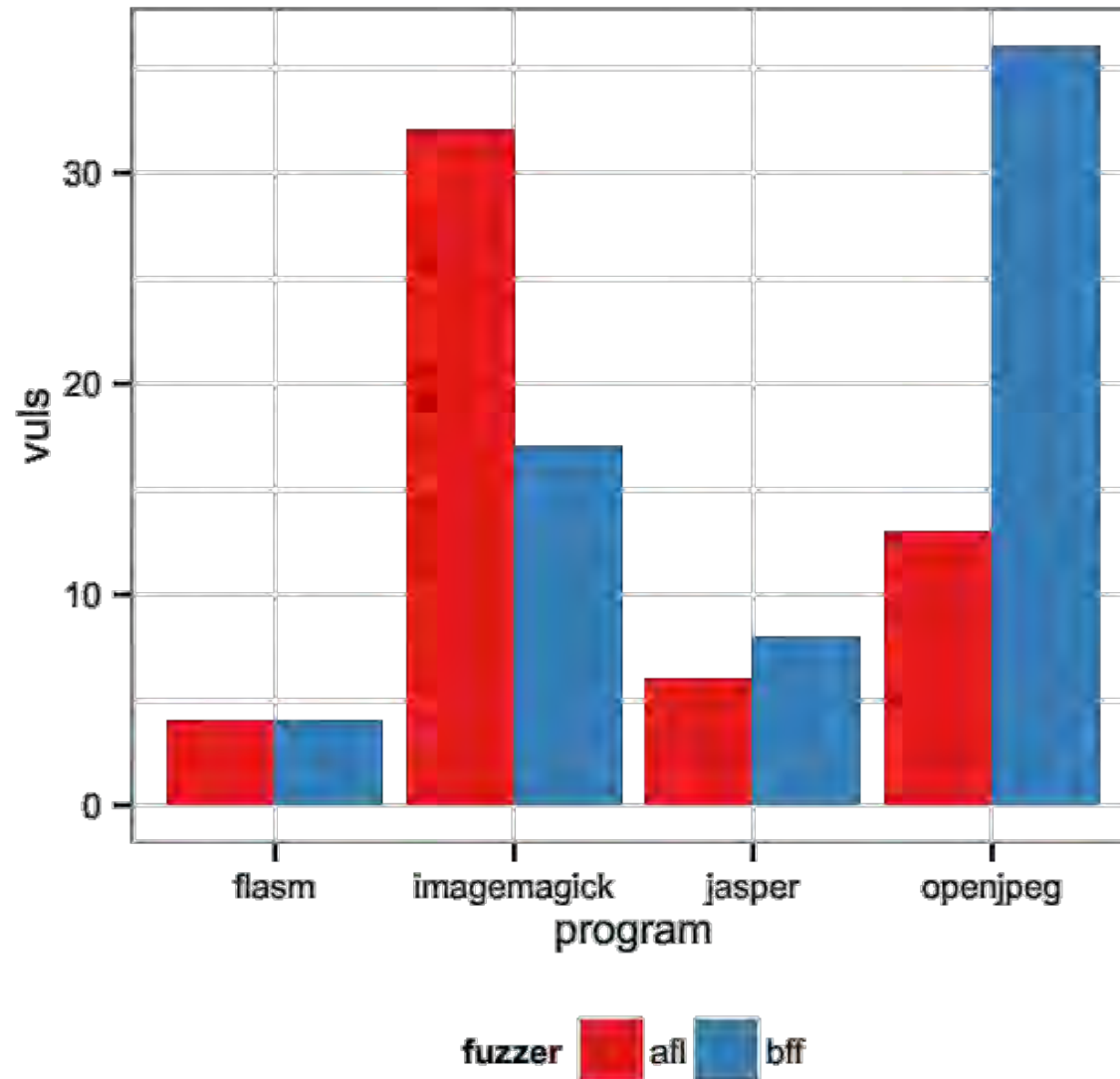
- # times a vulnerability is included more than once by stack backtrace hashing on average
- We see these vulnerabilities more than once

Program	# Vuls	UC	%	OC	%
Flasm	6	1.8	29%	410.9	6,848%
ImageMagick	31	1.9	6%	67.9	219%
Jasper	12	0.0	0%	226.4	1,887%
OpenJpeg	36	0.1	0%	267.5	743%

Belief 2: Sanitization Never Harms Fuzzing Performance

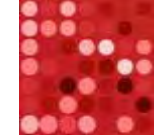


Belief 3: The AFL Fuzzer Always Finds More Vulnerabilities Than Non-Guided Fuzzers



Collaboration with ForAllSecure





DEFENSE ADVANCED
RESEARCH PROJECTS AGENCY

ABOUT US / OUR RESEARCH / NEWS / EVENTS / WORK WITH US /

EXPLORE BY TAG

Defense Advanced Research Projects Agency > News And Events > [DARPA Celebrates Cyber Grand Challenge Winners](#)

DARPA Celebrates Cyber Grand Challenge Winners

Top-scoring cyber reasoning system becomes first machine to be invited to participate in DEF CON Capture the Flag tournament

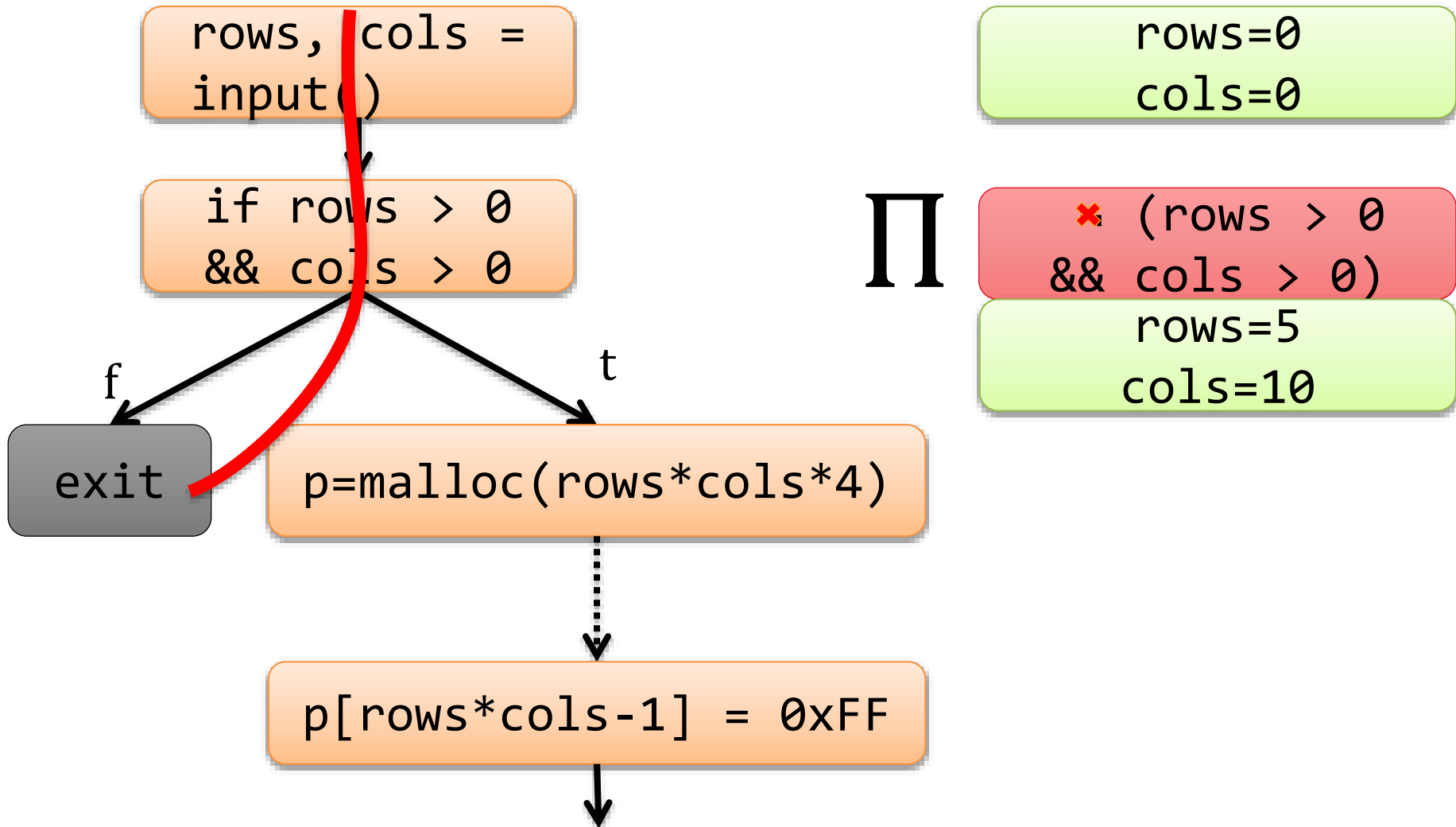
OUTREACH@DARPA.MIL
8/5/2016



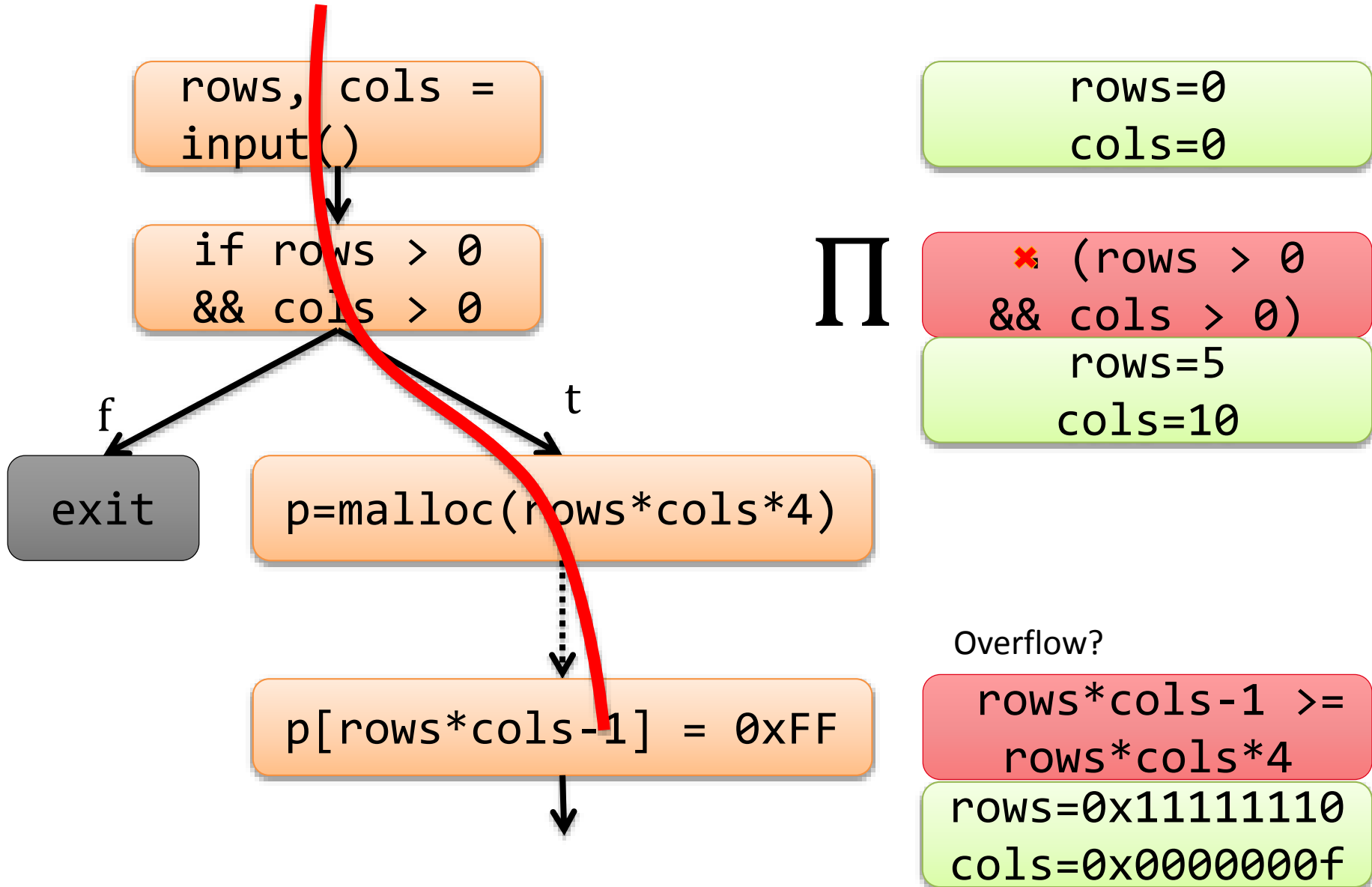
(Updated August 7, 2016)

DARPA officials this morning released partial final, audited results of yesterday's all-day [Cyber Grand Challenge \(CGC\) Final Event](#)—the world's first all-machine cyber hacking tournament—and confirmed that the top-scoring machine was Mayhem, developed by team ForAllSecure of Pittsburgh.

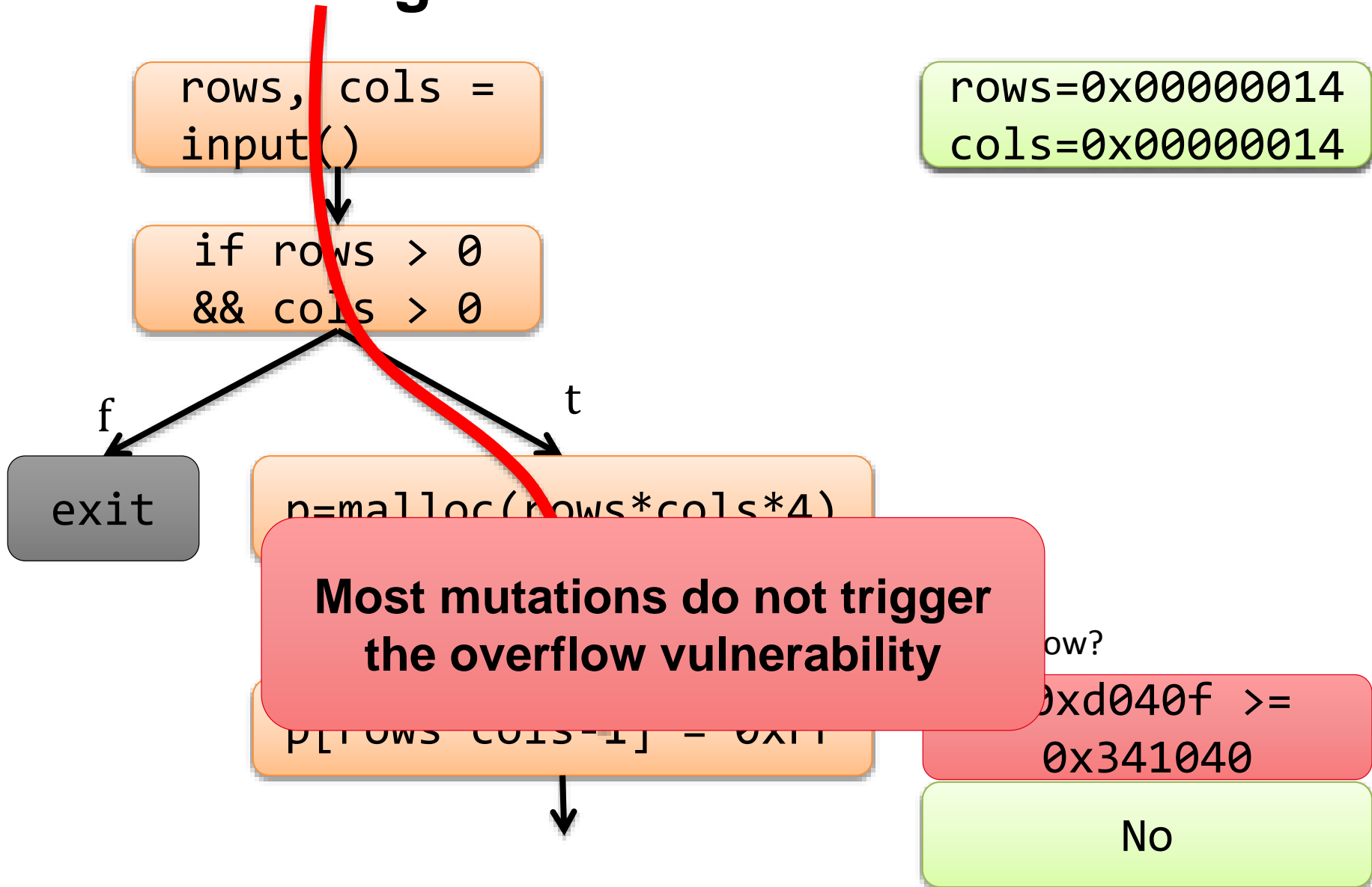
Background: Concolic Execution



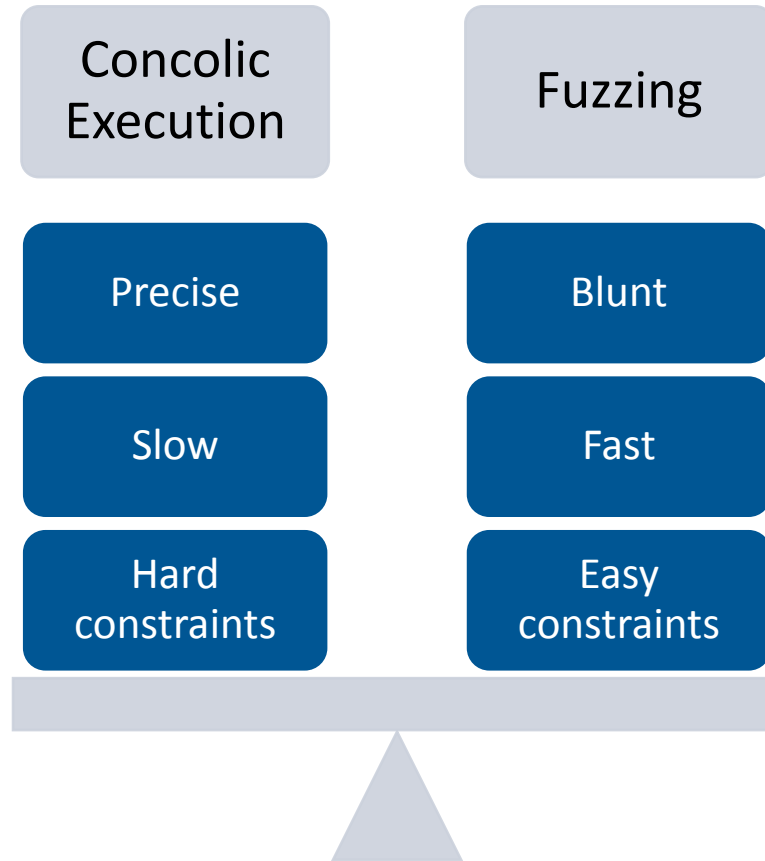
Background: Concolic Execution



Background: Fuzzing



Concolic Execution vs. Fuzzing



SMART



The Synergistic Mayhem AFL Research Tool

- Concolic execution: Mayhem (ForAllSecure+SEI)
- Fuzzing: AFL
- Periodically synchronize seed files between them

Challenges

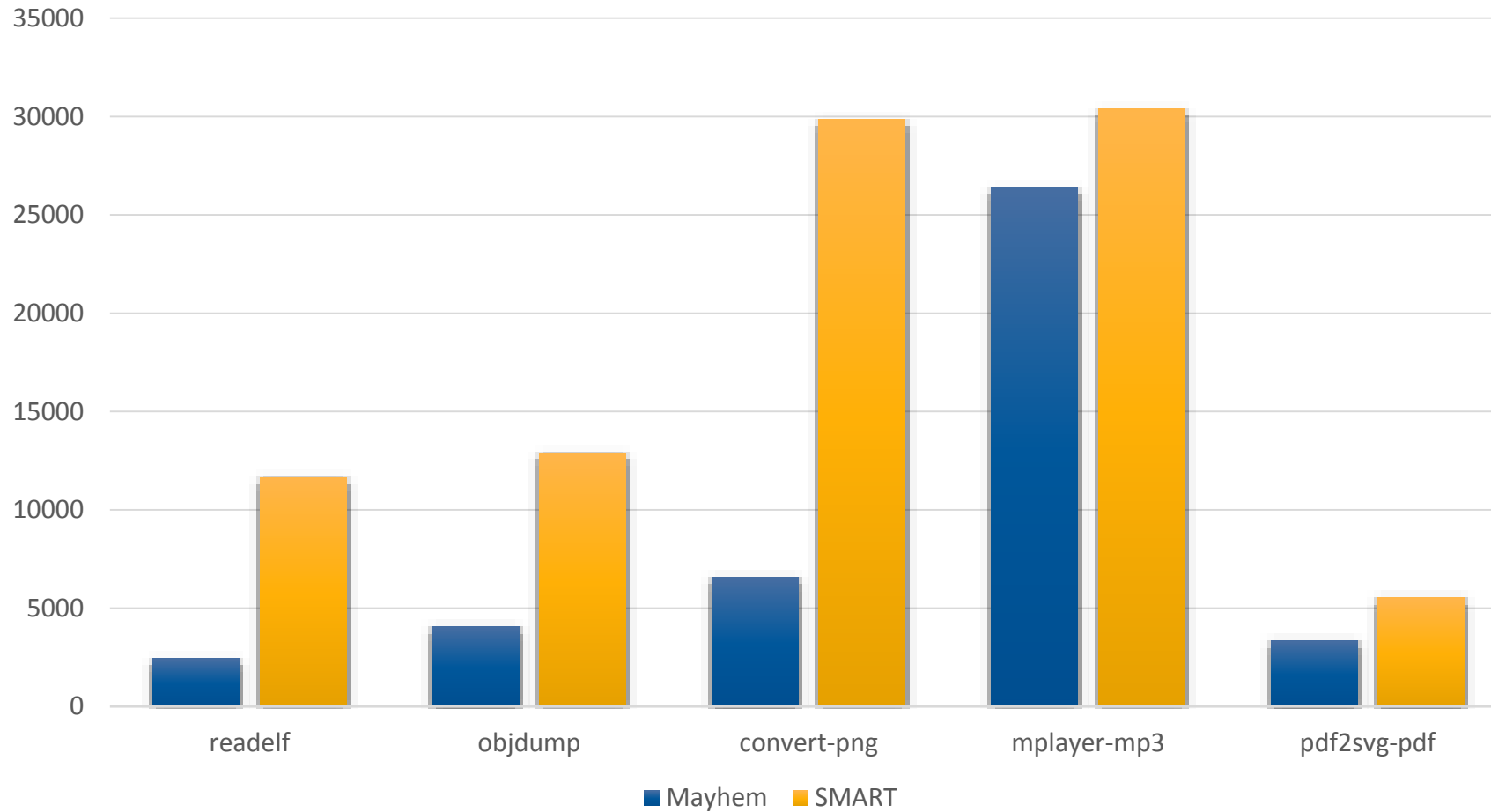
How much should we use concolic execution?

- $\sim 10^4$ times slower than fuzzing
- Brute force vs. high cost

SMART evaluation



Edge Coverage After Two Days with Blank Seeds



Summary

- Developing new techniques for discovering and mitigating vulnerabilities in the DoD
- Developed vulnerability uniqueness model and used ground truth to explore common fuzzing (mis-)beliefs
- ForAllSecure: Hybrid fuzzing and concolic tester

Team Members

- Edward Schwartz, PhD, CERT
- David Warren, CERT
- Allen Householder, CERT

ForAllSecure Inc.:

- David Brumley, PhD
- Thanassis Avgerinos, PhD
- Tyler Nighswander