

# Vulnerability Discovery

Dr. Edward J. Schwartz

*eschwartz@cert.org*

Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213



**Software Engineering Institute**

**Carnegie Mellon University**

© 2015 Carnegie Mellon University

Distribution Statement A: Approved for Public Release;  
Distribution is Unlimited

Copyright 2015 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.

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.

This material has been approved for public release and unlimited distribution except as restricted below.

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](mailto:permission@sei.cmu.edu).

Carnegie Mellon® and CERT® are registered marks of Carnegie Mellon University.

DM-0002802



# Vulnerability Discovery Project

Increase **assurance** of 1<sup>st</sup> and 3<sup>rd</sup> party  
DoD software through **enhanced**  
**vulnerability discovery techniques**



# Team



## Software Engineering Institute

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

## Collaborators

### V.Secure

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



# Agenda



**Towards vulnerability discovery as a science**

**Intelligent fusion of vulnerability discovery techniques**

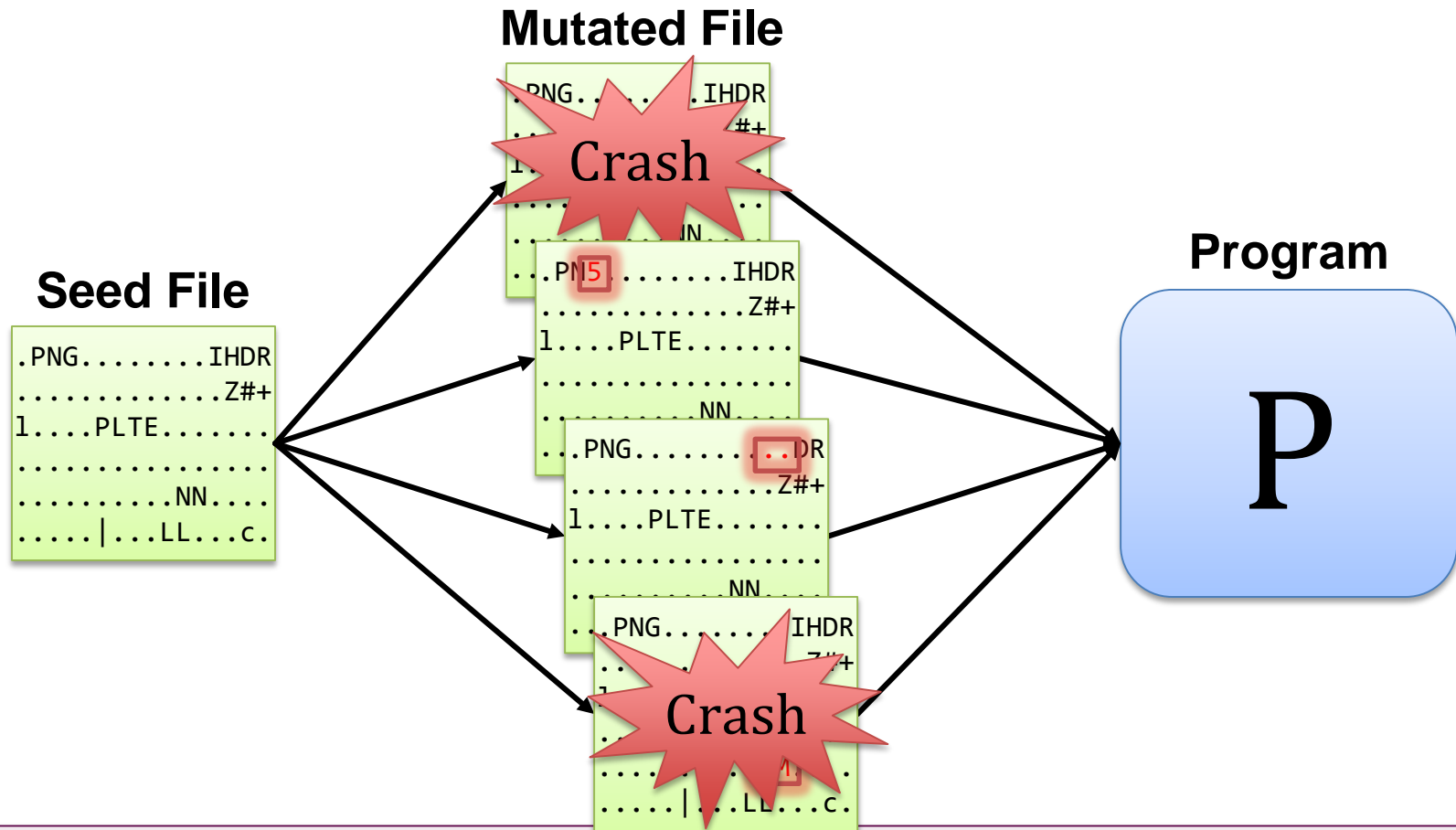




# Towards Vulnerability Discovery as a Science



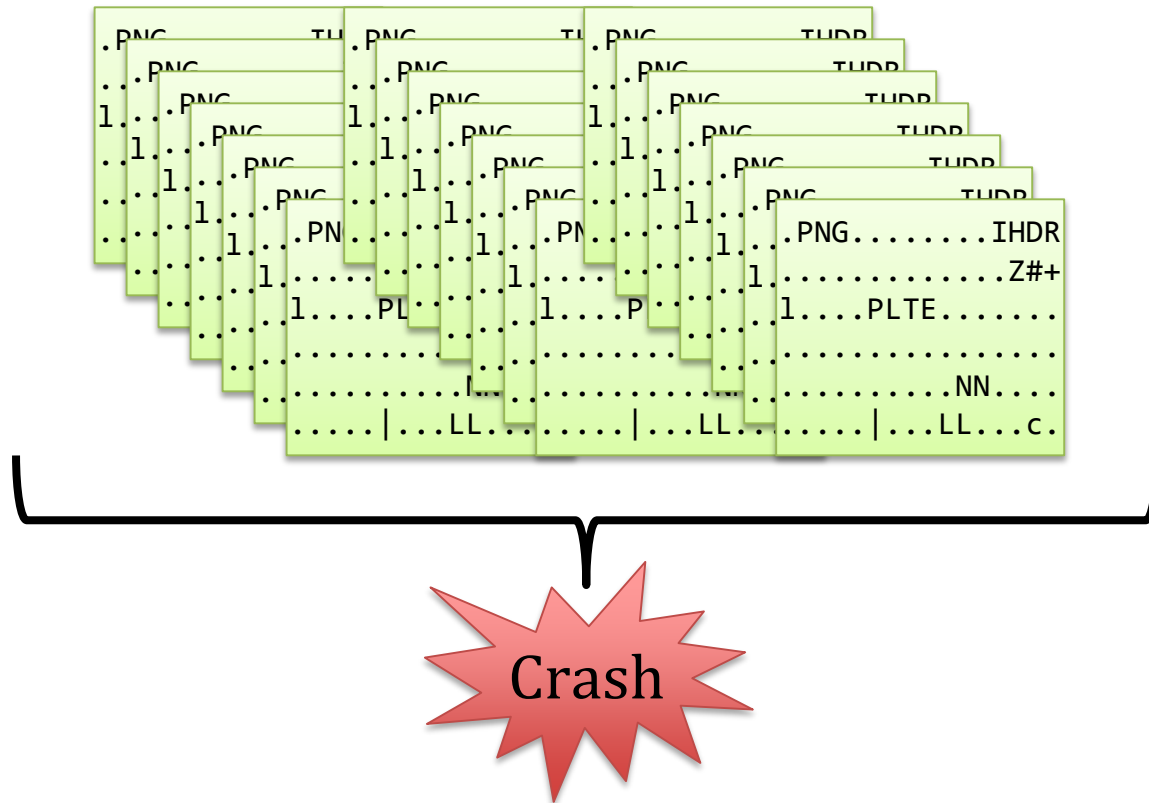
# Background: Mutational Fuzzing of Software



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

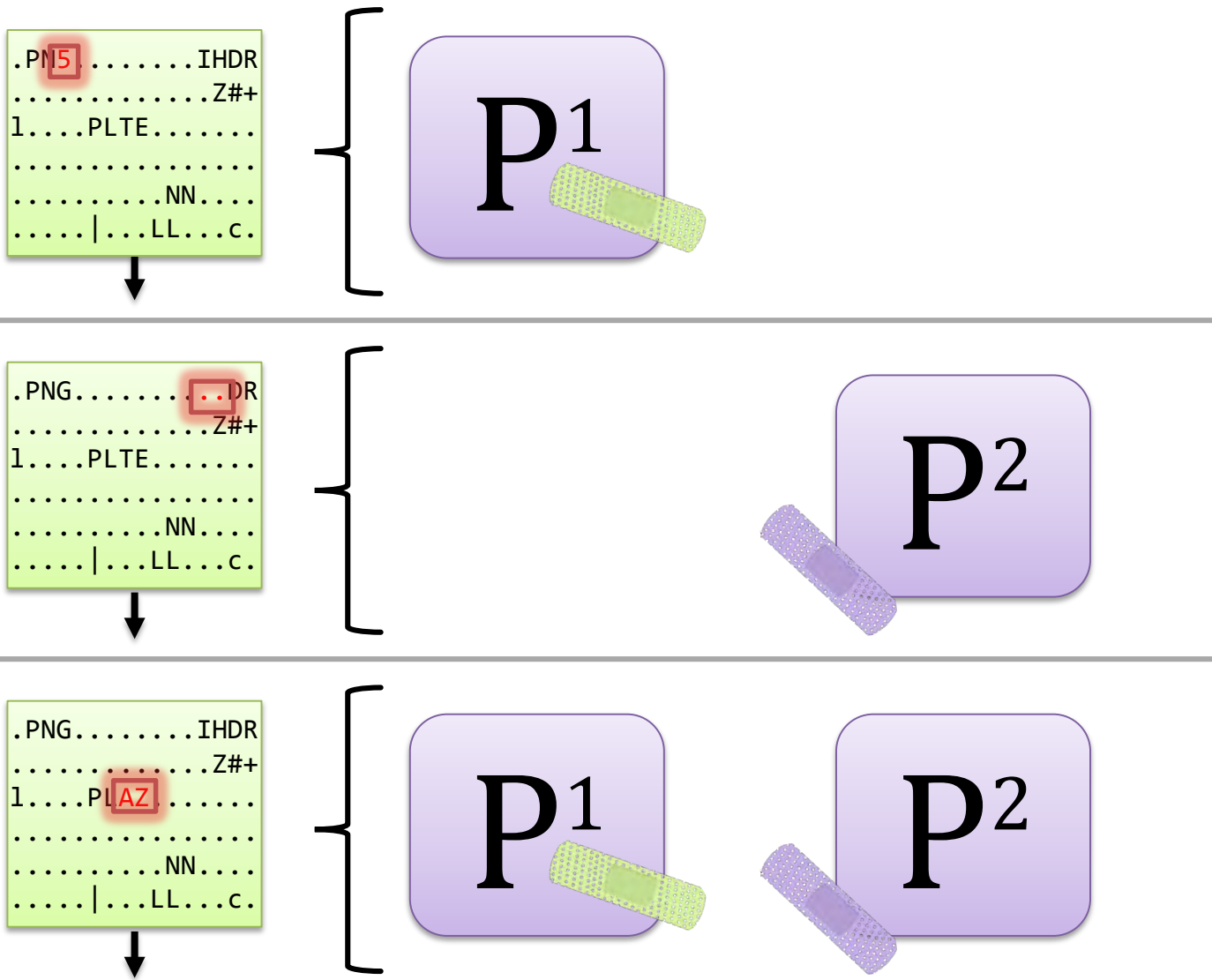


# The Idea: Patches Define Vulnerabilities



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

# Example Ground Truth



# Patching ImageMagick

## Fuzzed old ImageMagick with the CERT BFF fuzzer

- 1 week
- 130,000 crashes found

## Manually patched all vulnerabilities

- Took approximately one month
- 31 patches/vulnerabilities



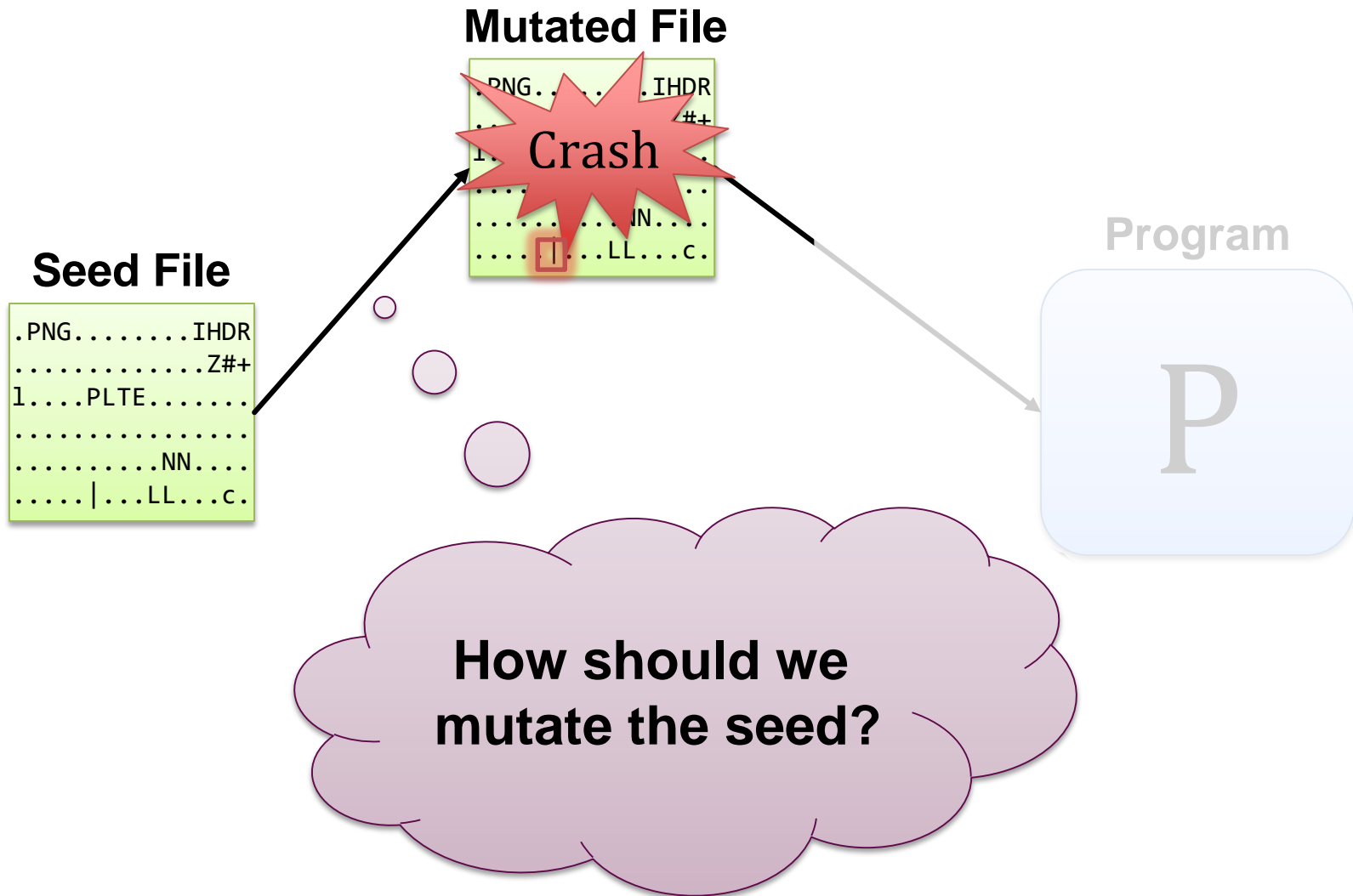
# Vulnerability Discovery Science

## Analyze fuzzing parameters

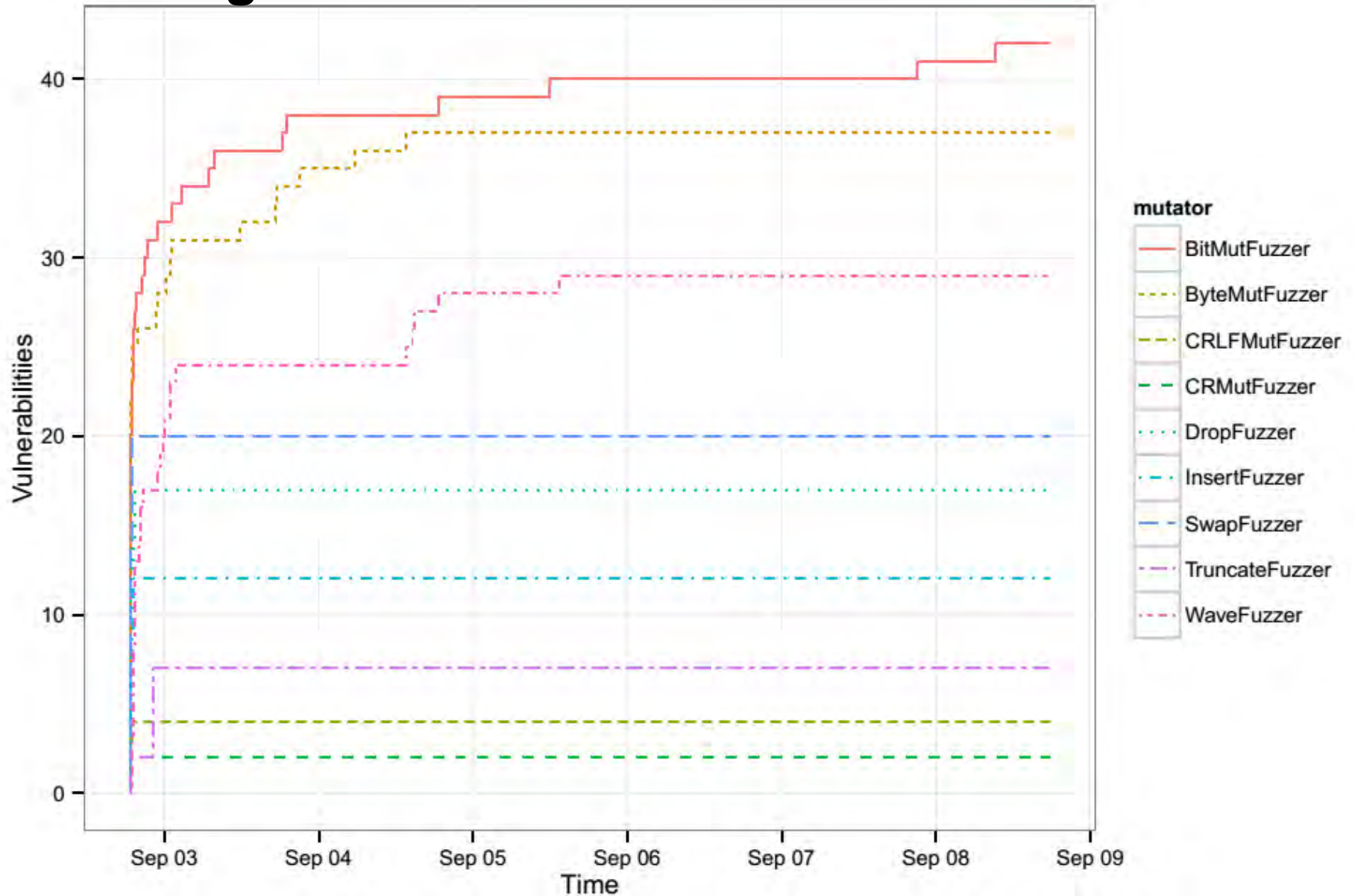
- What mutators work best?
- When should we stop fuzzing?
- What effect do compiler settings have?
- ...
- Paper submitted to NDSS 2016



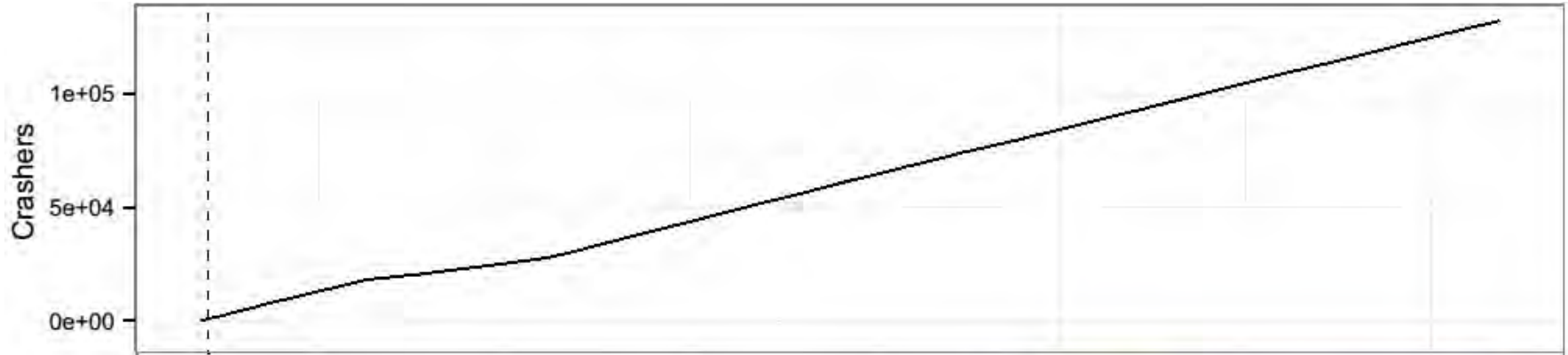
# Background: Mutational Fuzzing of Software



# Fuzzing Mutators

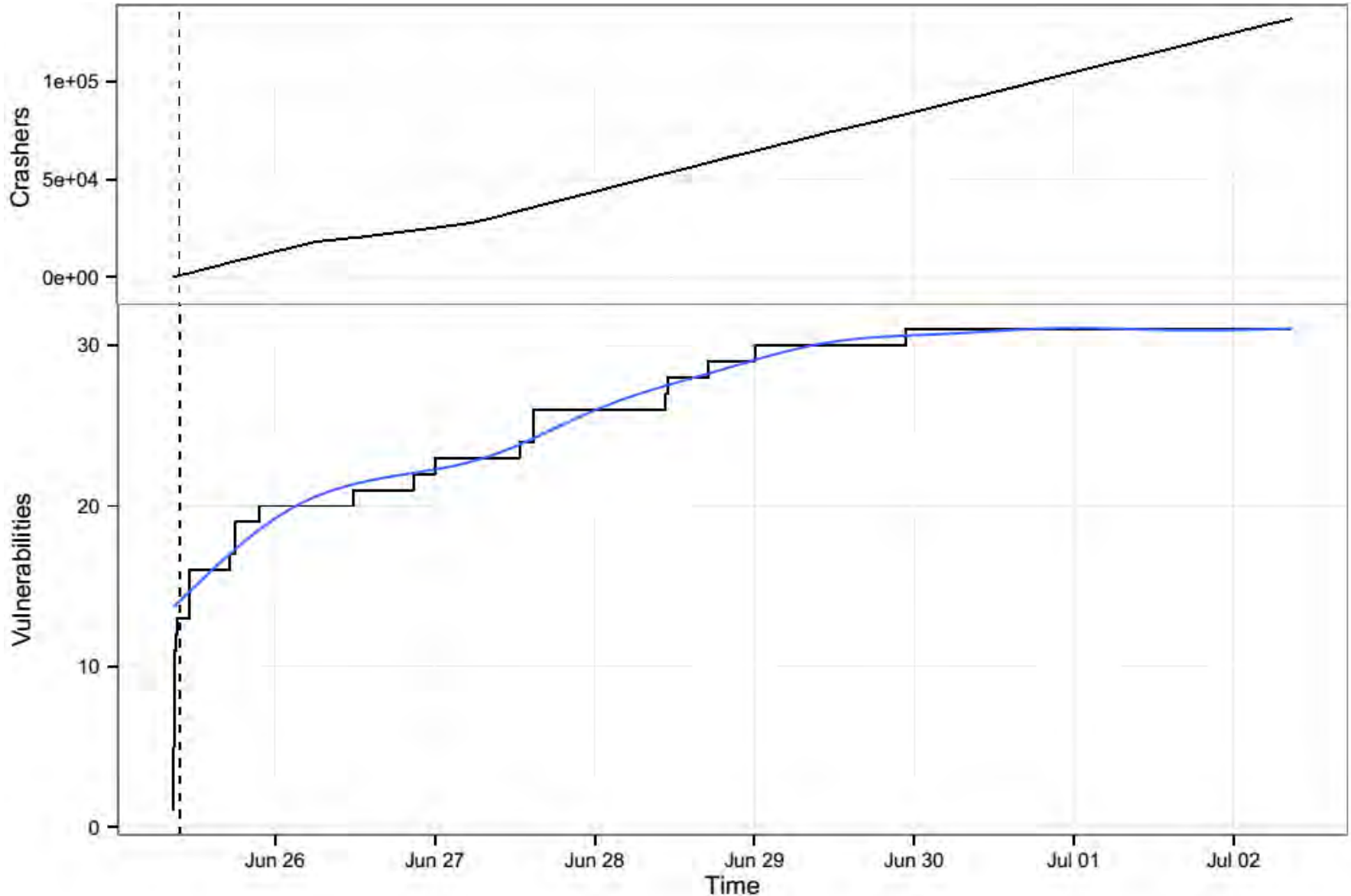


# When to Stop Fuzzing?





# When to Stop Fuzzing?





# Collaboration with ForAllSecure



## CMU Spinoff of David Brumley's research group



**Dr. David Brumley**



**Dr. Thanassis Avgerinos**



**Alex Rebert**

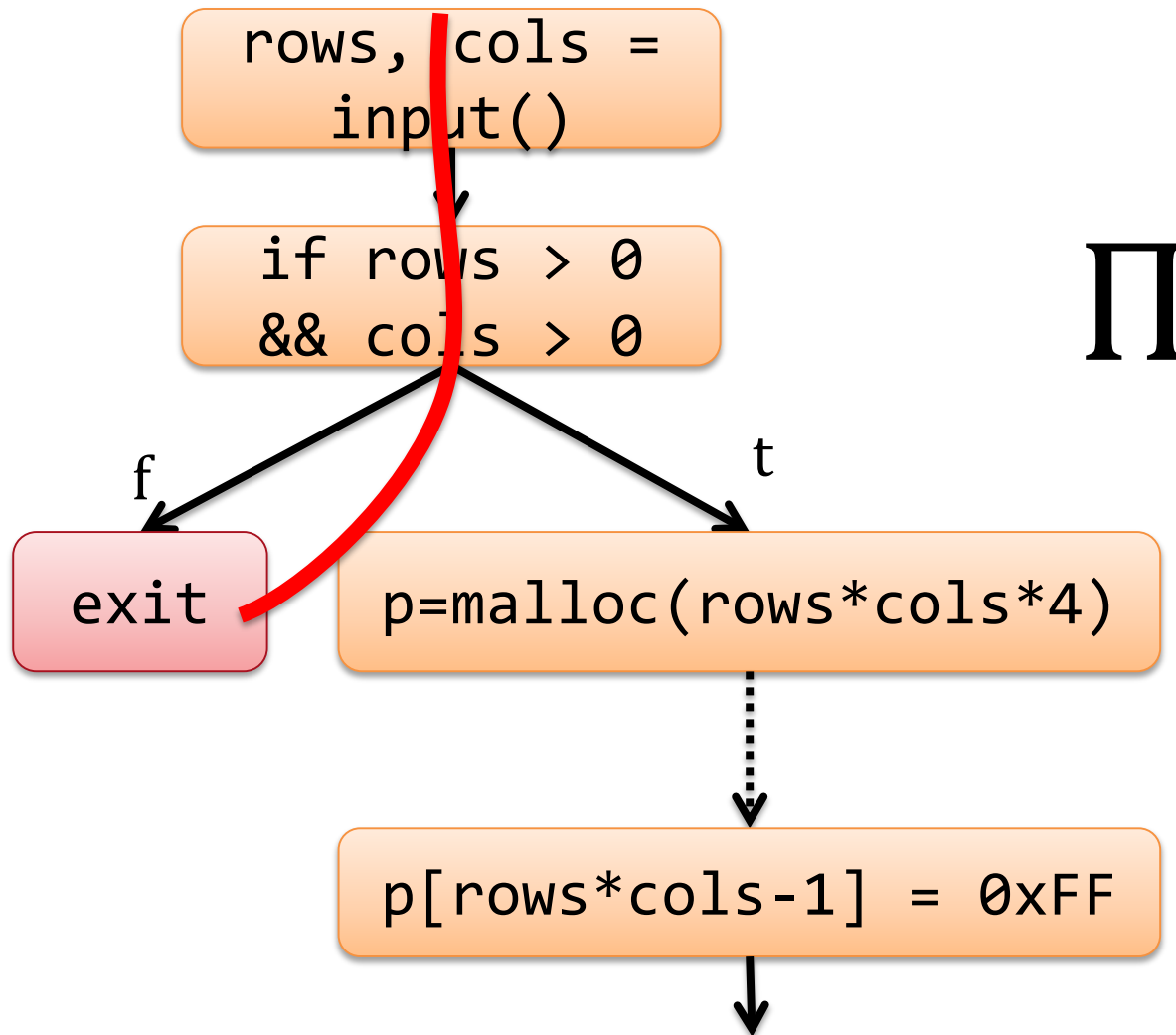
## Expertise

- Concolic execution
- Automatic exploitation
- Binary analysis
- Complements SEI's expertise in fuzzing

## Previous collaboration

- With same group at CMU

# Background: Concolic Execution

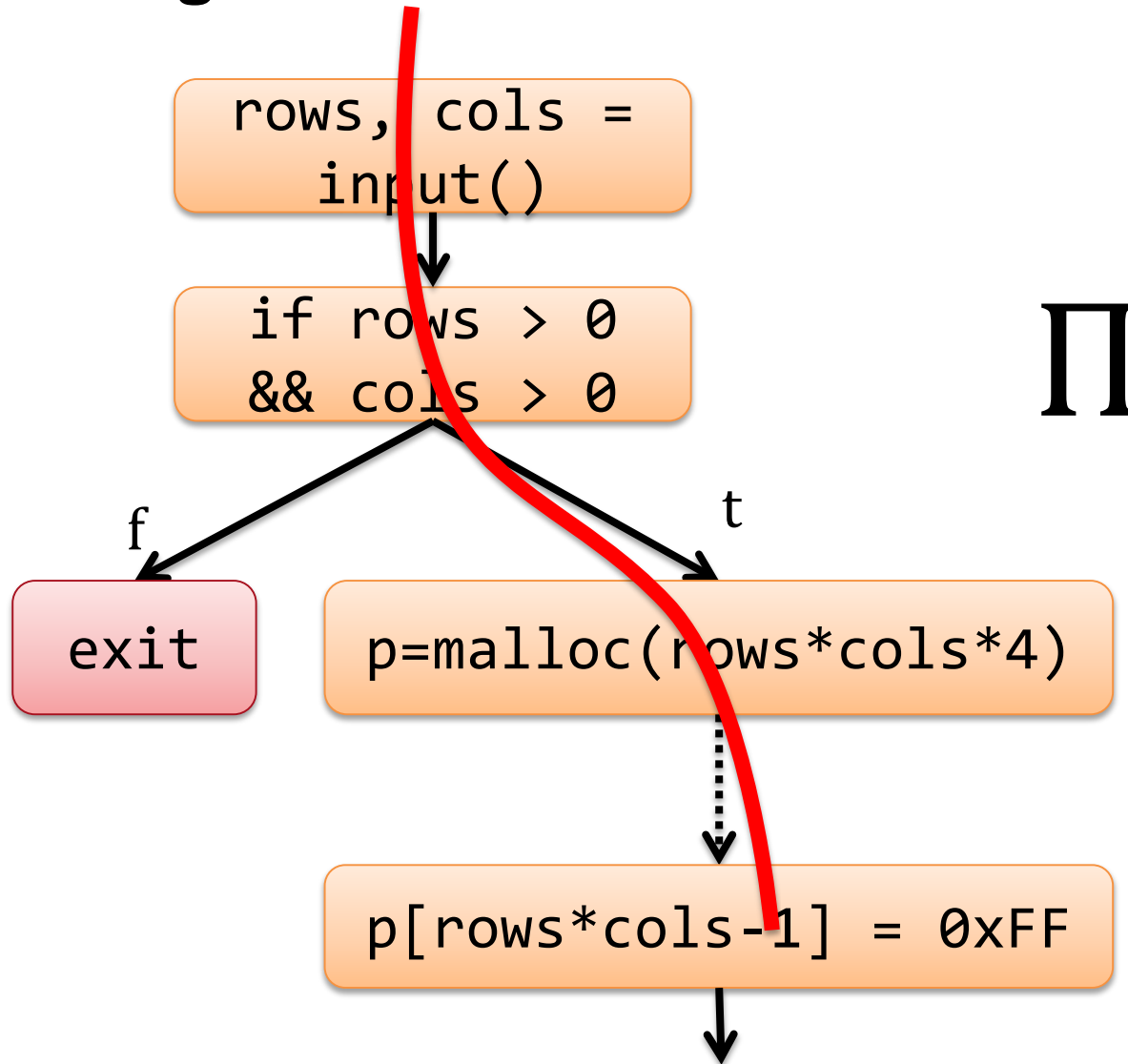


rows=0  
cols=0

✘ (rows > 0  
&& cols > 0)

rows=5  
cols=10

# Background: Concolic Execution



$\Pi$

rows=0  
cols=0

✘ (rows > 0  
&& cols > 0)

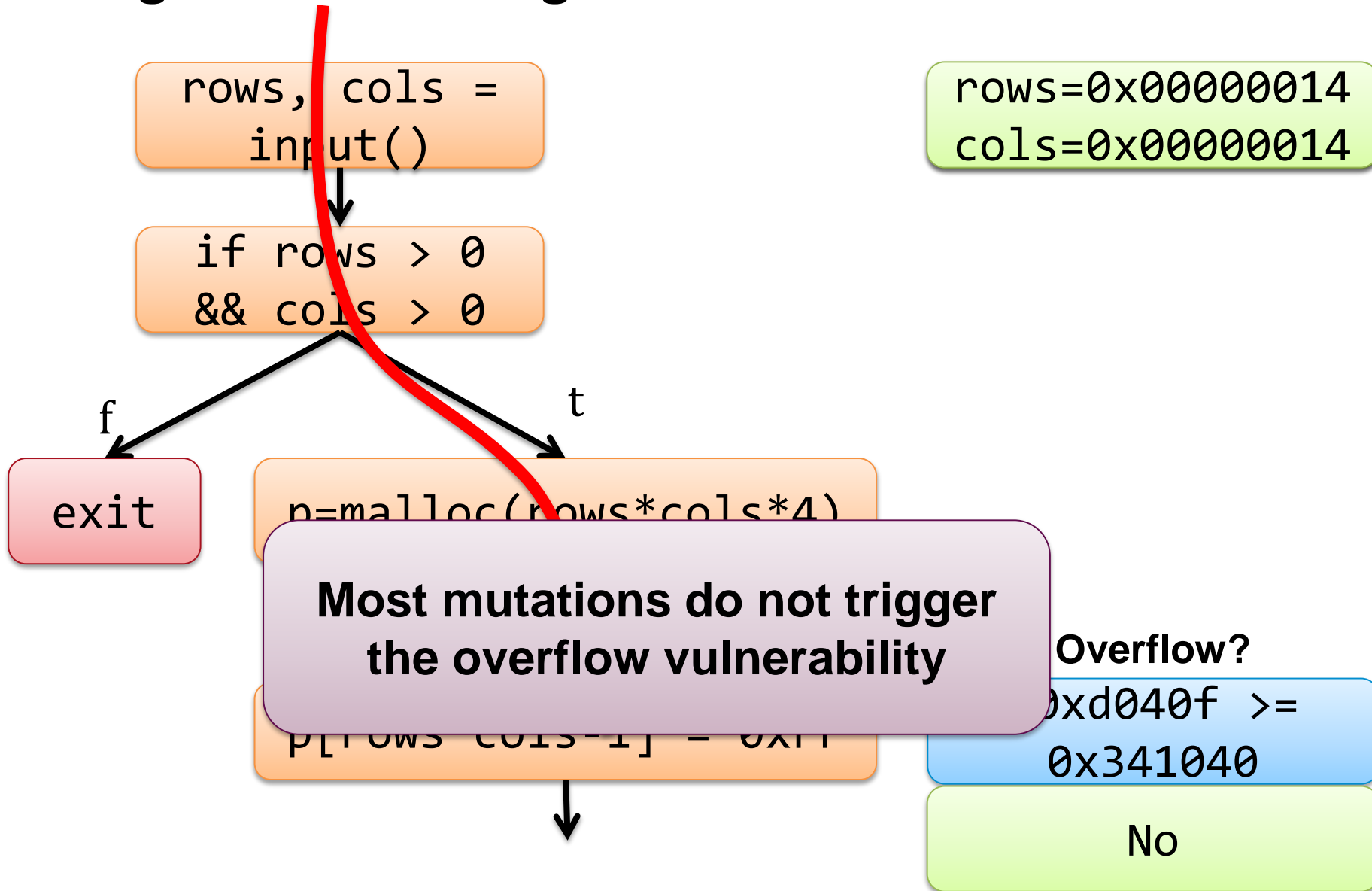
rows=5  
cols=10

**Overflow?**

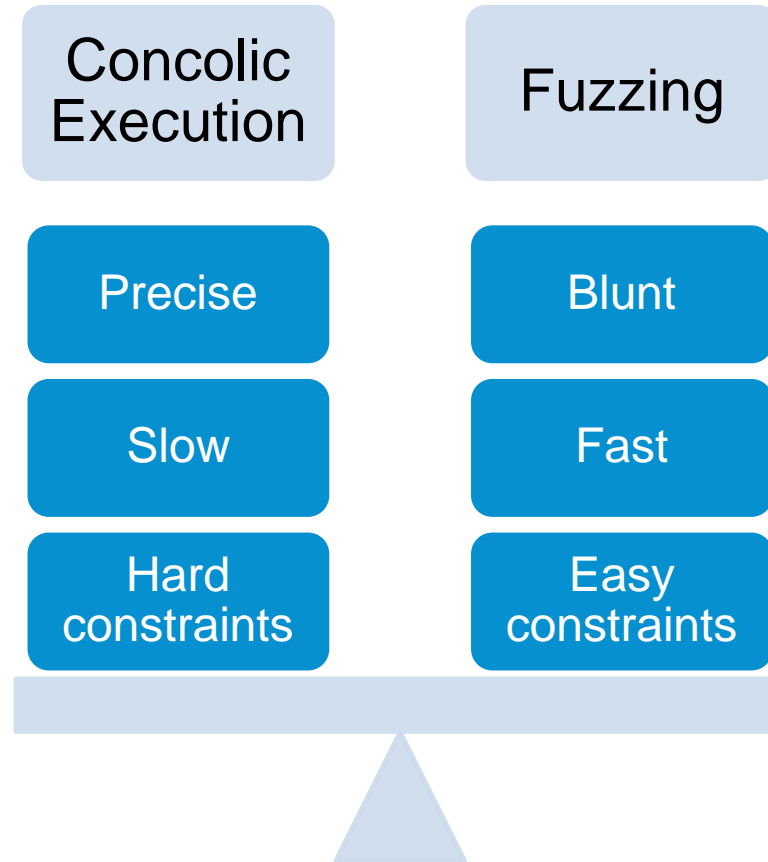
rows \* cols - 1 >= rows \* cols \* 4

rows=0x11111112  
cols=0x0000000f

# 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

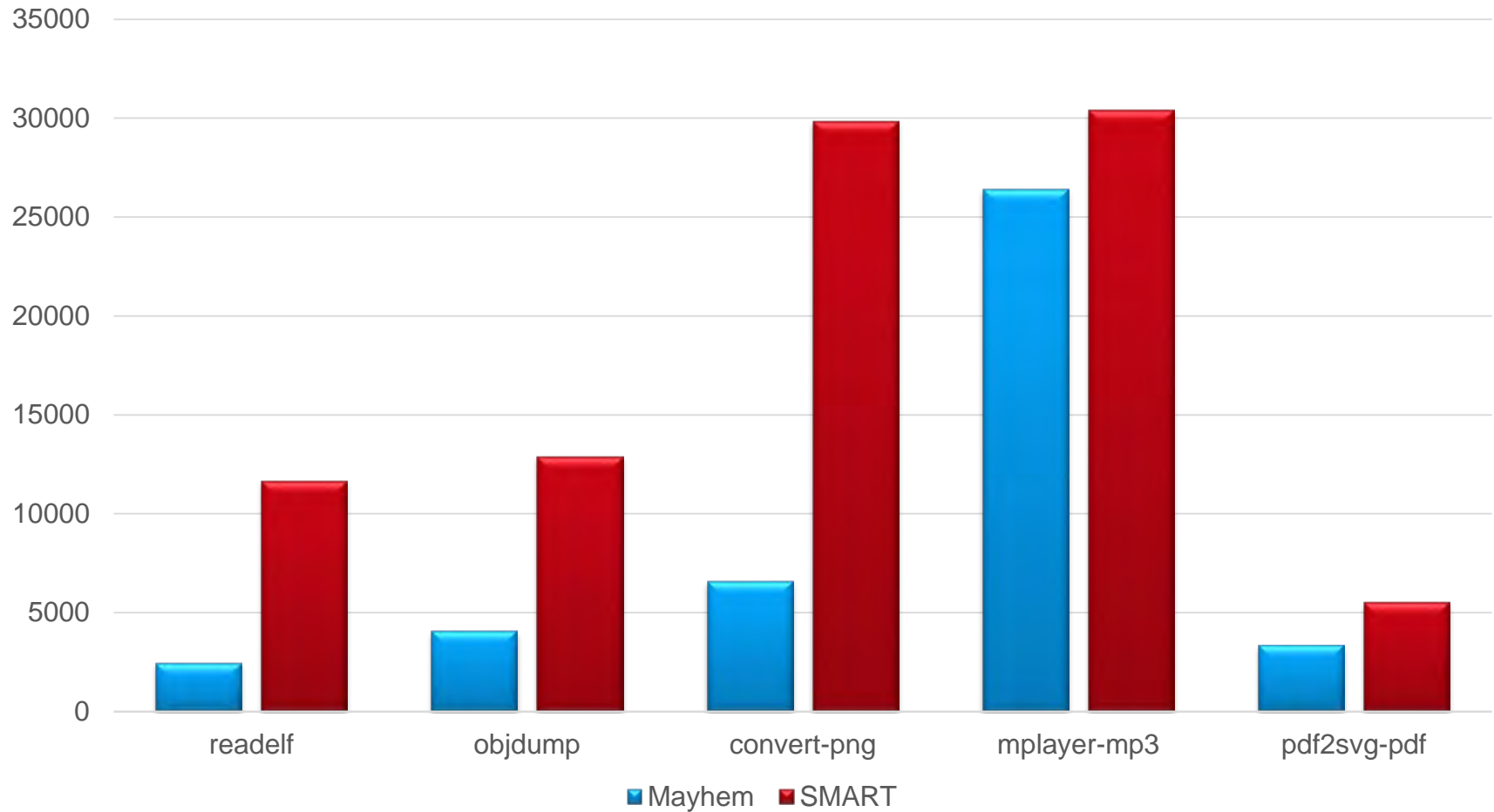
## Challenges

1. Where to go?
  - We don't know the location of vulnerabilities
2. 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 the effect of fuzzing parameters
- ForAllSecure: Hybrid fuzzing and concolic tester

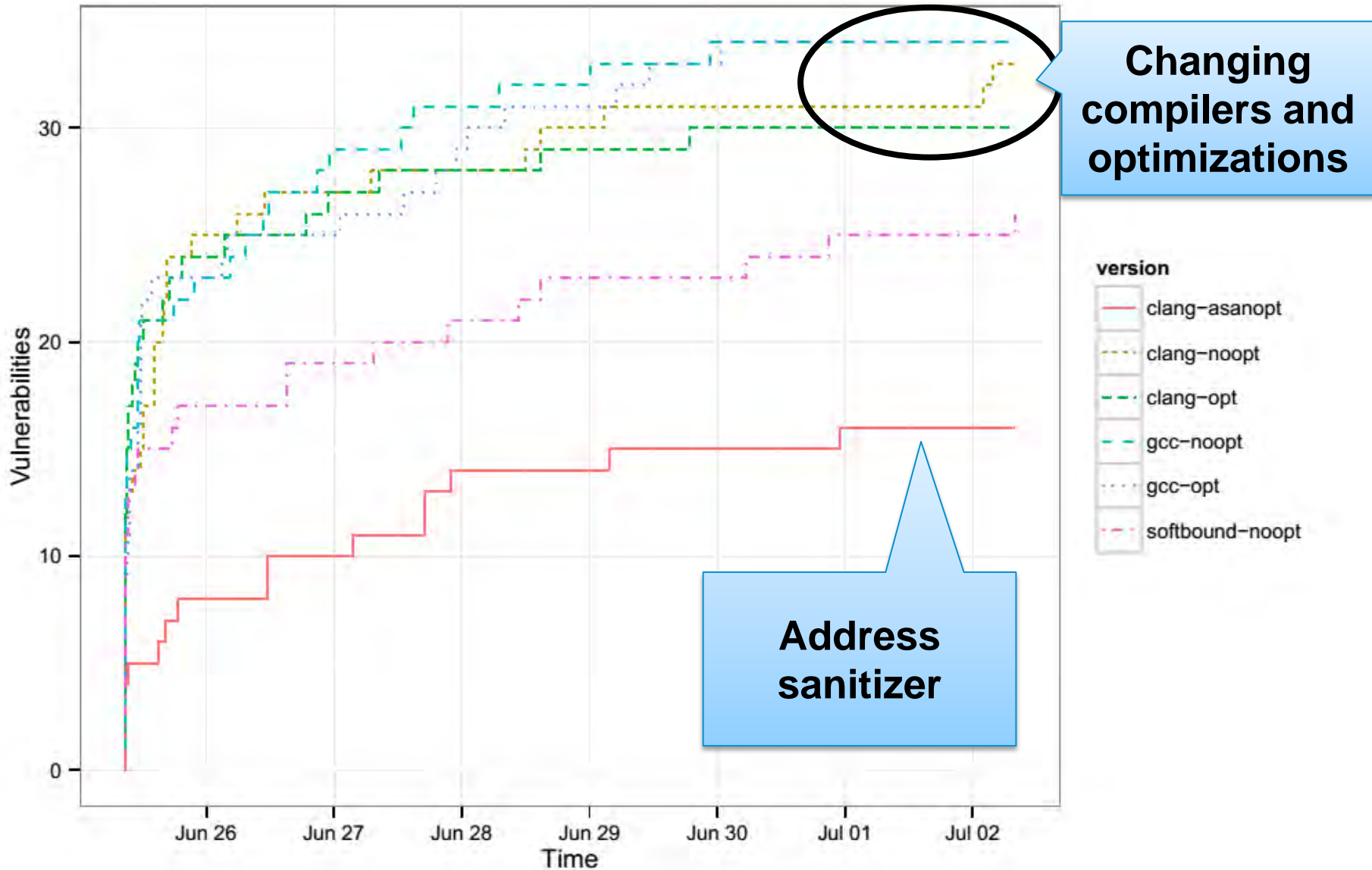
## Team Members

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

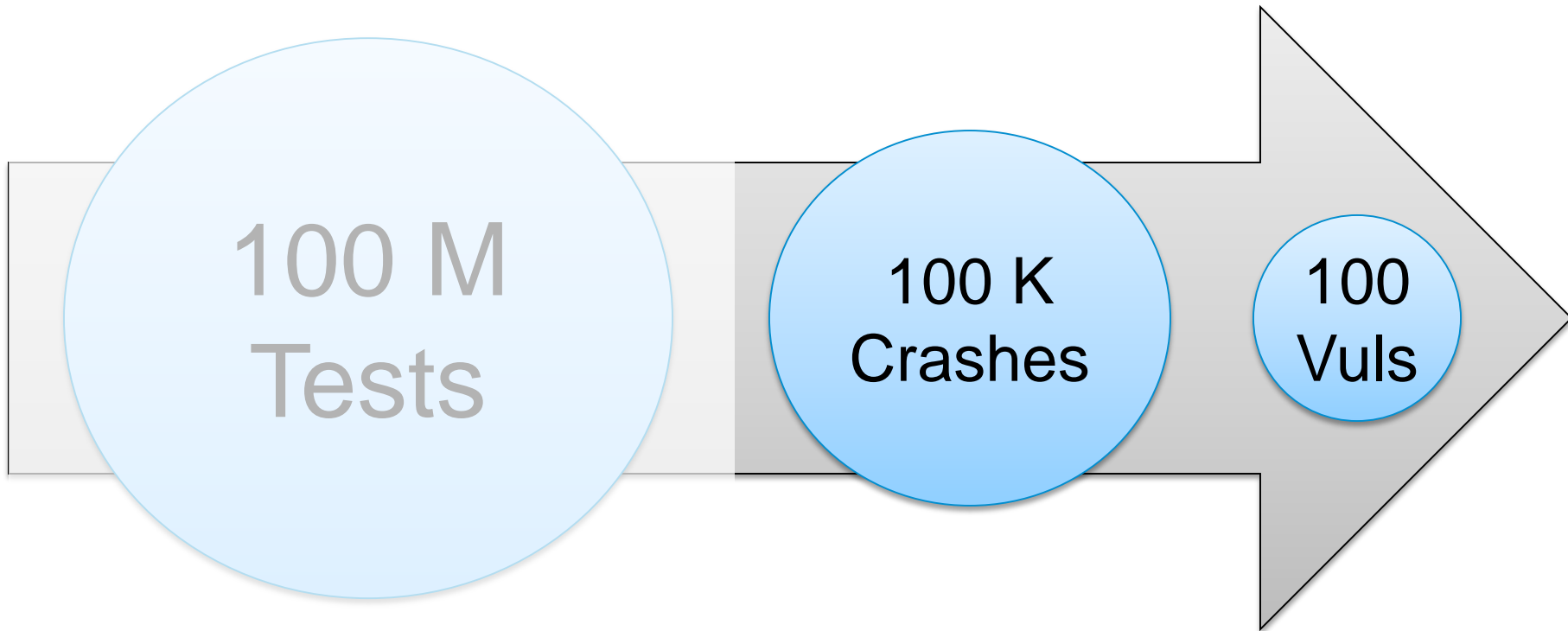
## *ForAllSecure, Inc.:*

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

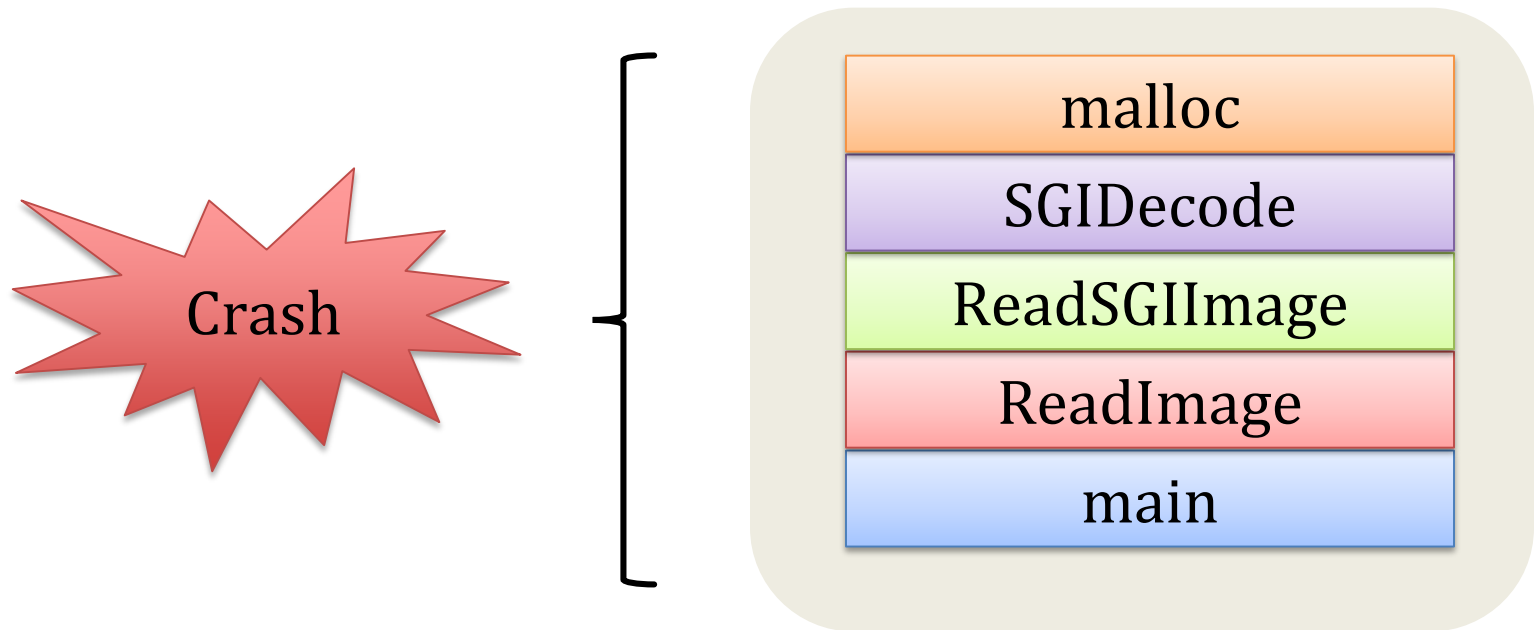
# Compiler Flags and Settings



# The Crash Uniqueness Problem



# The State of the Art: Stack Hashing

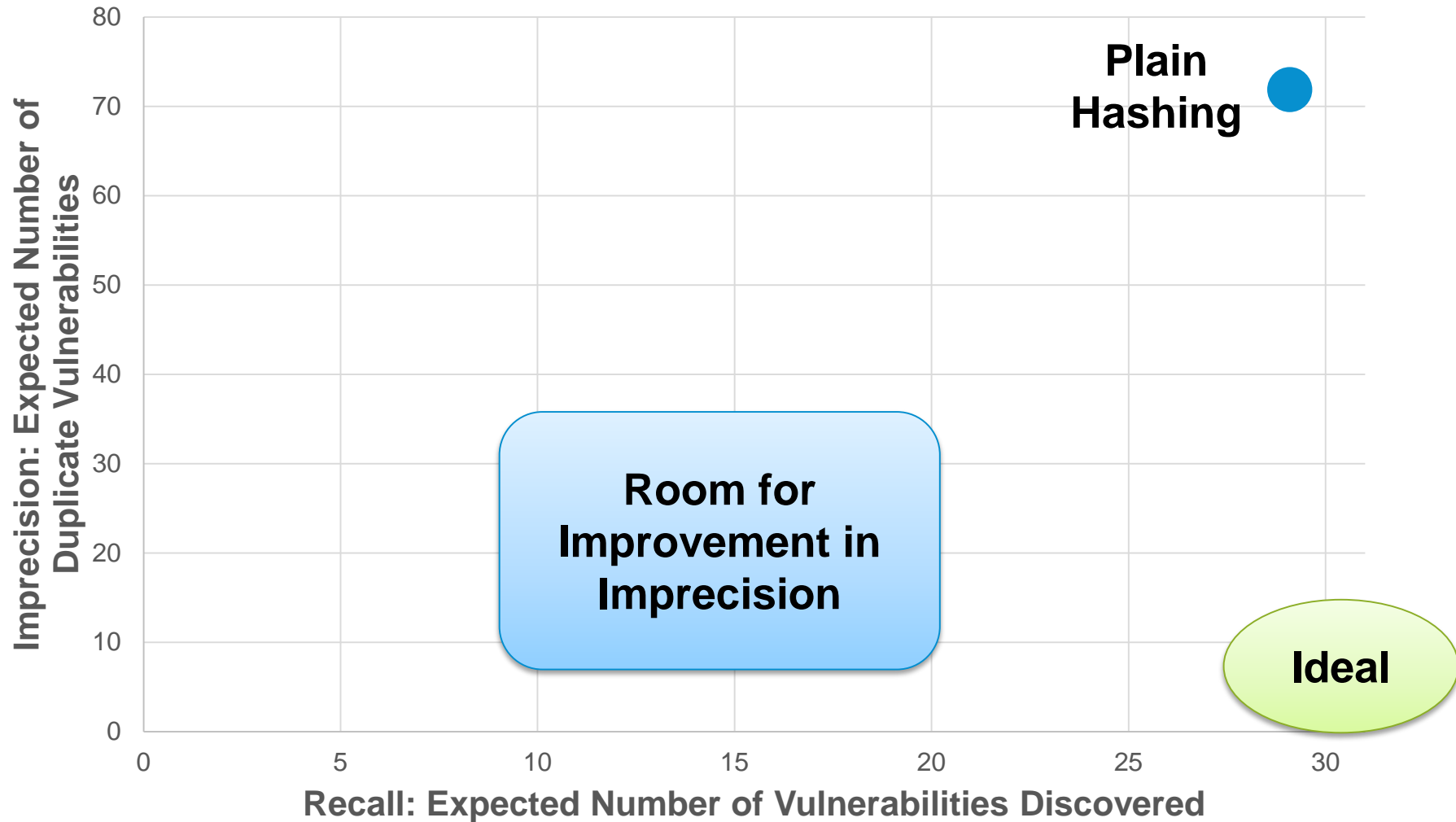


$$\text{Hash}(\text{[blue][red][green][purple][orange]}) = \begin{matrix} \text{b1946ac9} \\ \text{2492d234} \\ \text{7c6235b4} \\ \text{d2611184} \end{matrix}$$

# Does Stack Hashing Work?

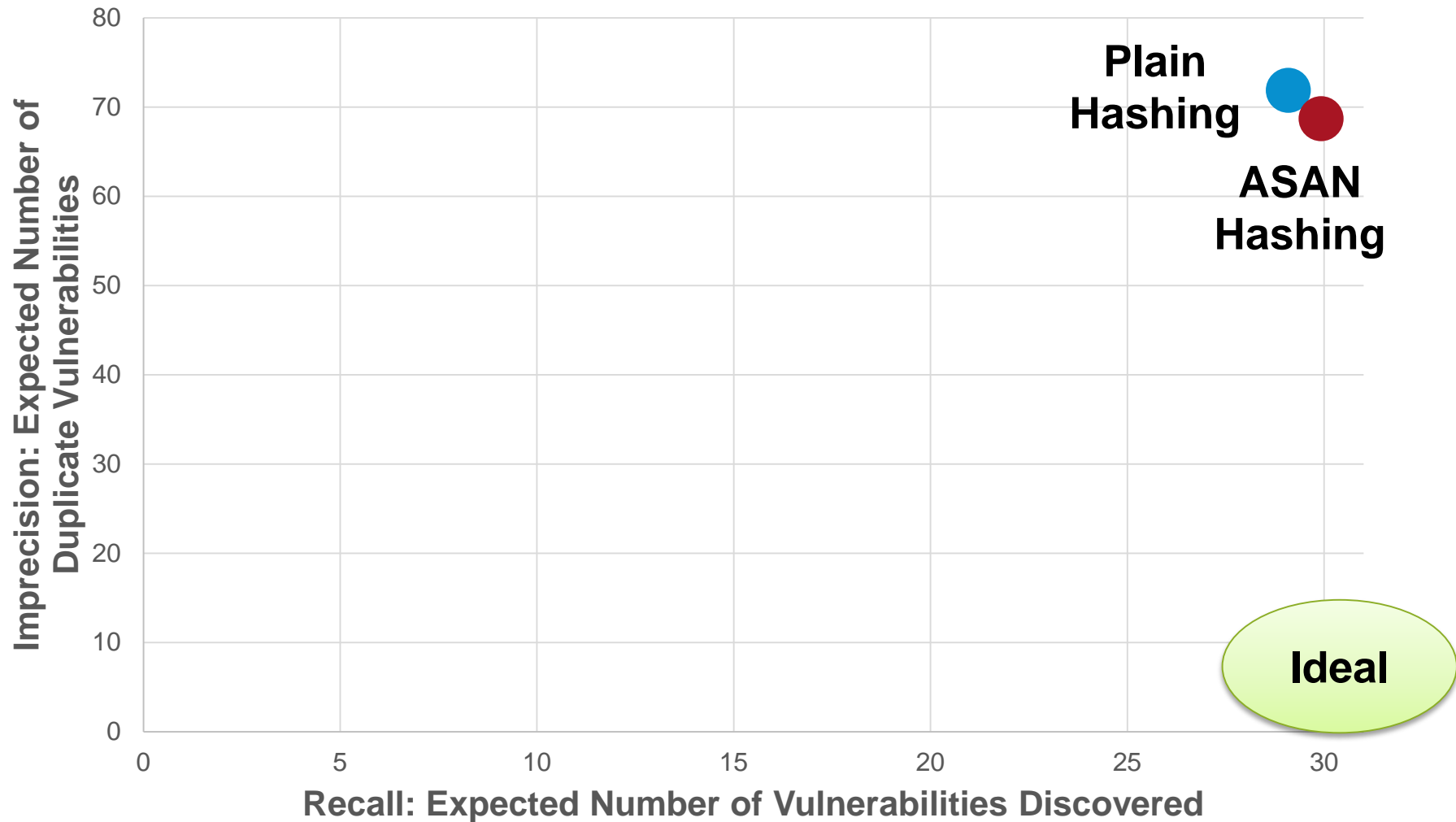


# Does Stack Hashing Work?



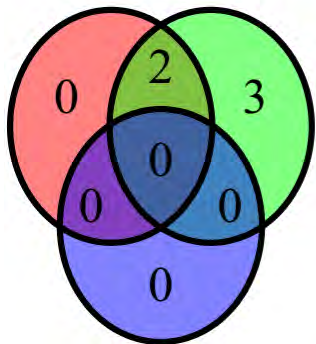


# Does Stack Hashing Work?

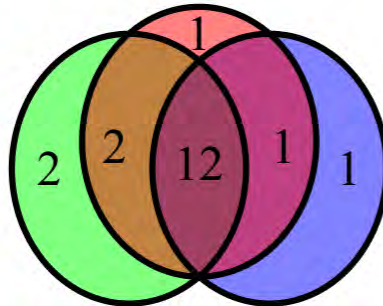


# Importance of Seed Selection

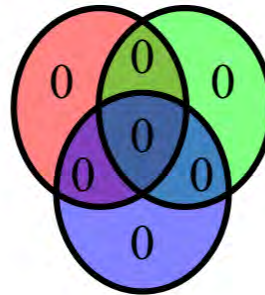
gif



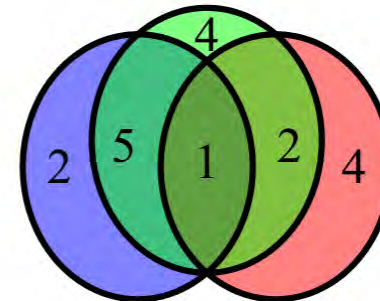
jpg



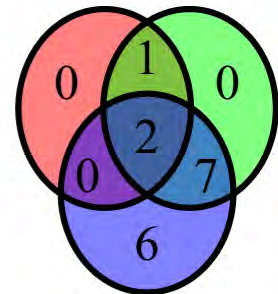
pdf



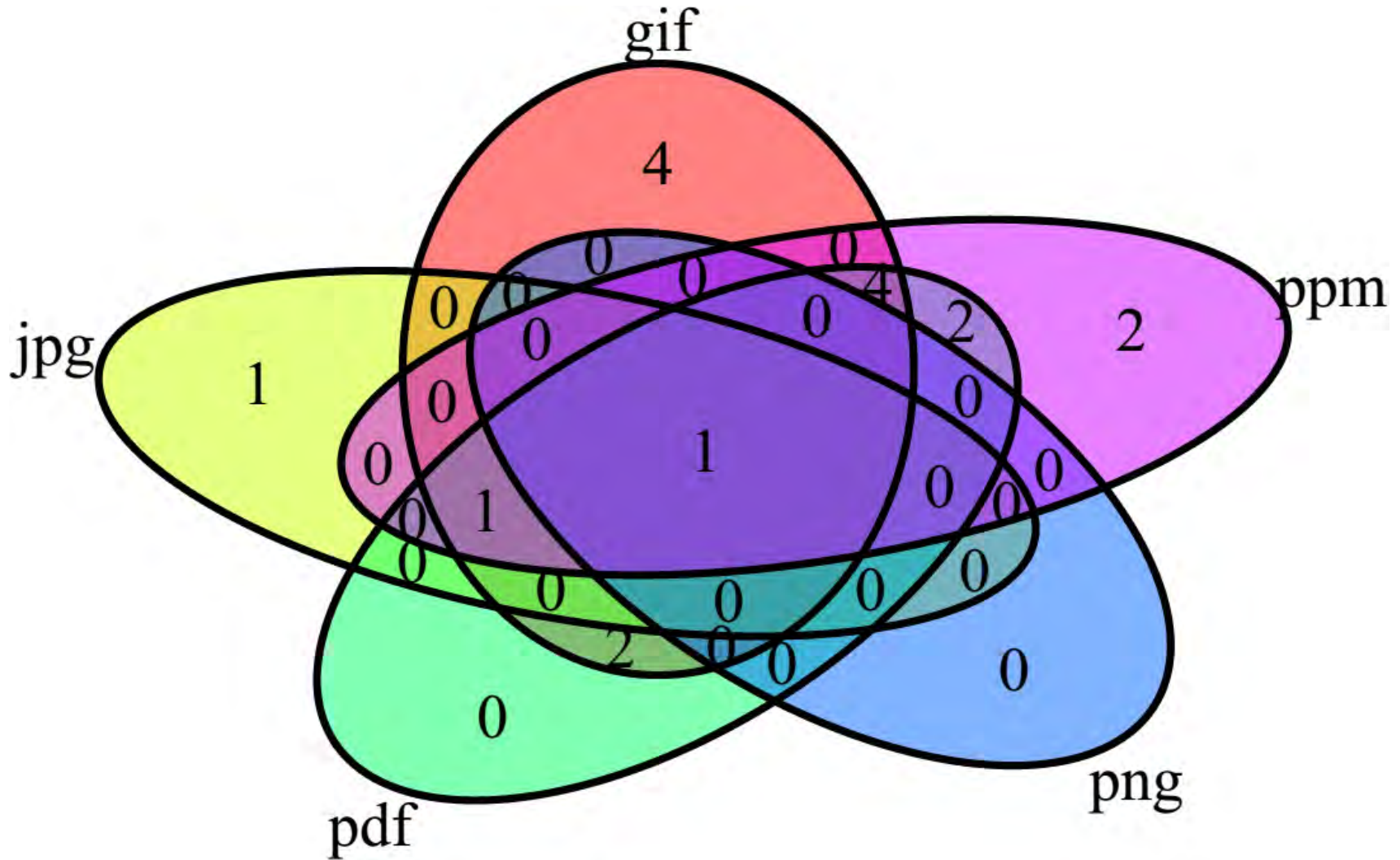
png



ppm



# Importance of Seed Selection



# Challenge: Multiple Vulnerabilities

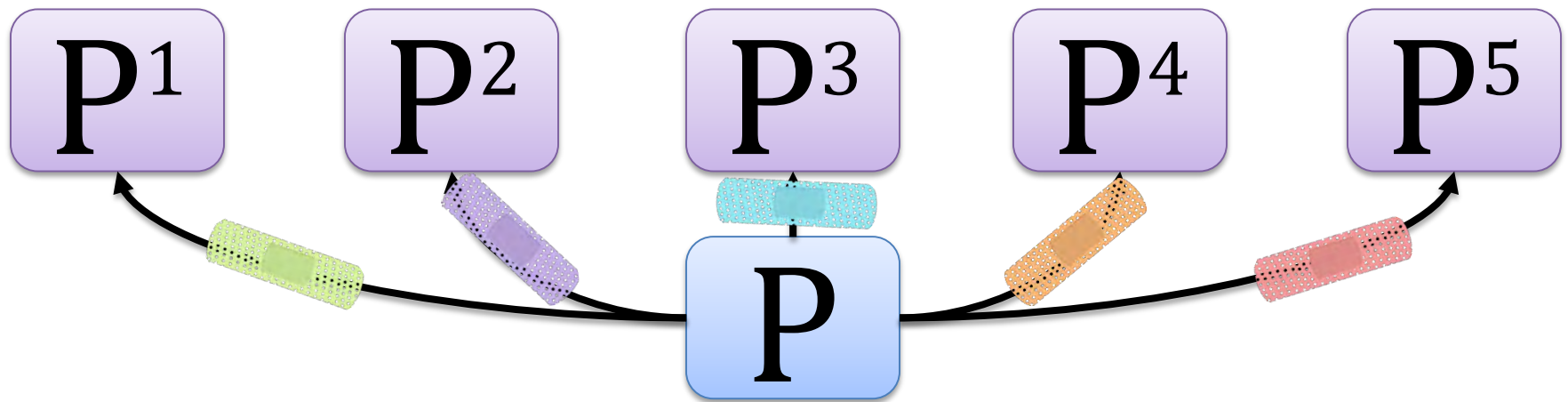
```
int main(int argc, char* argv[]) {  
    int x = atoi(argv[1]);  
    if (x&1) vulA(1);  
    if (x&2) vulB(1);  
}
```

Which vulnerability causes `main(3)`?

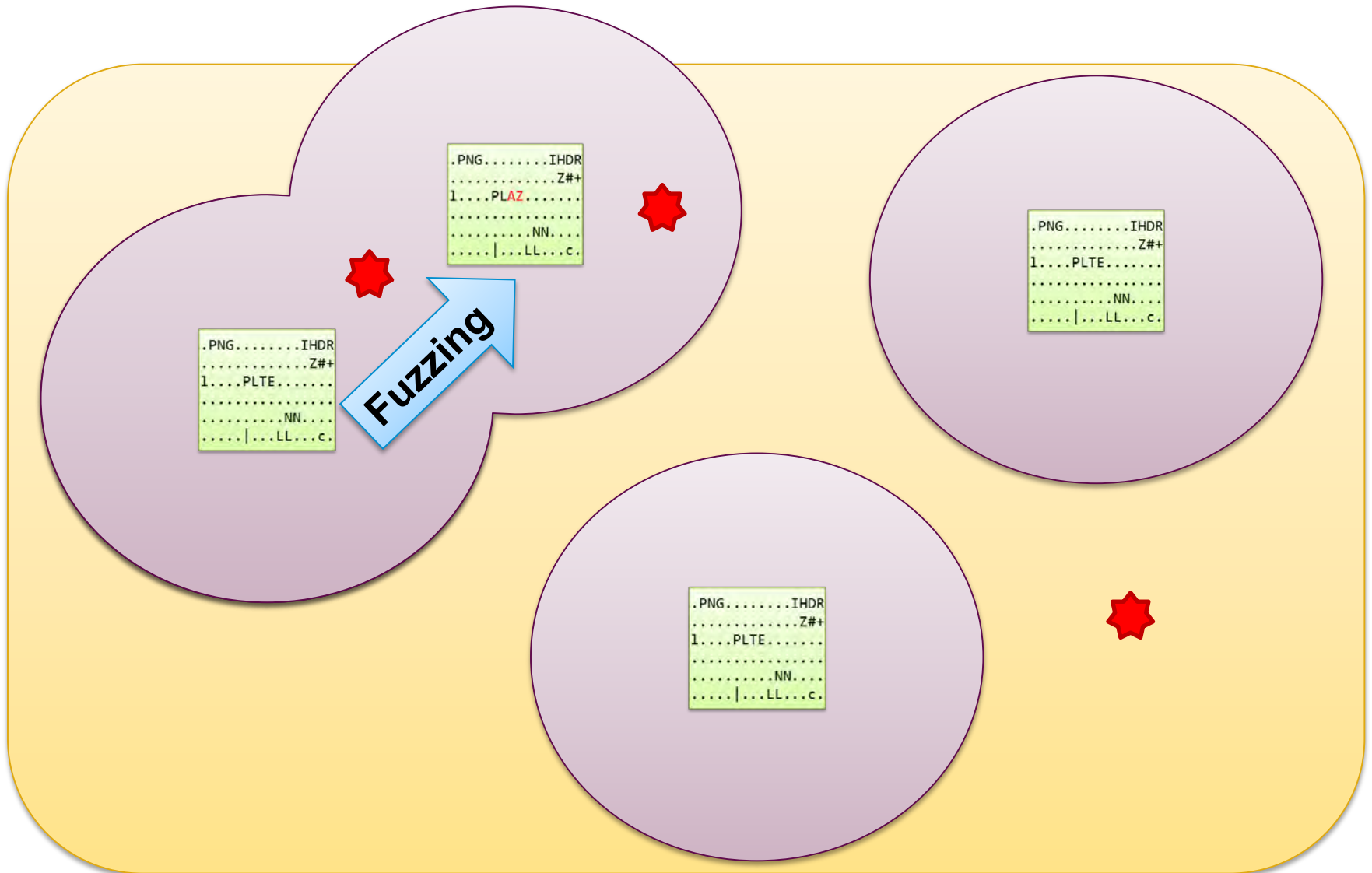
Vuls	1	2	3	4	5
Crashes	45859	<b>79626</b>	6860	21	1



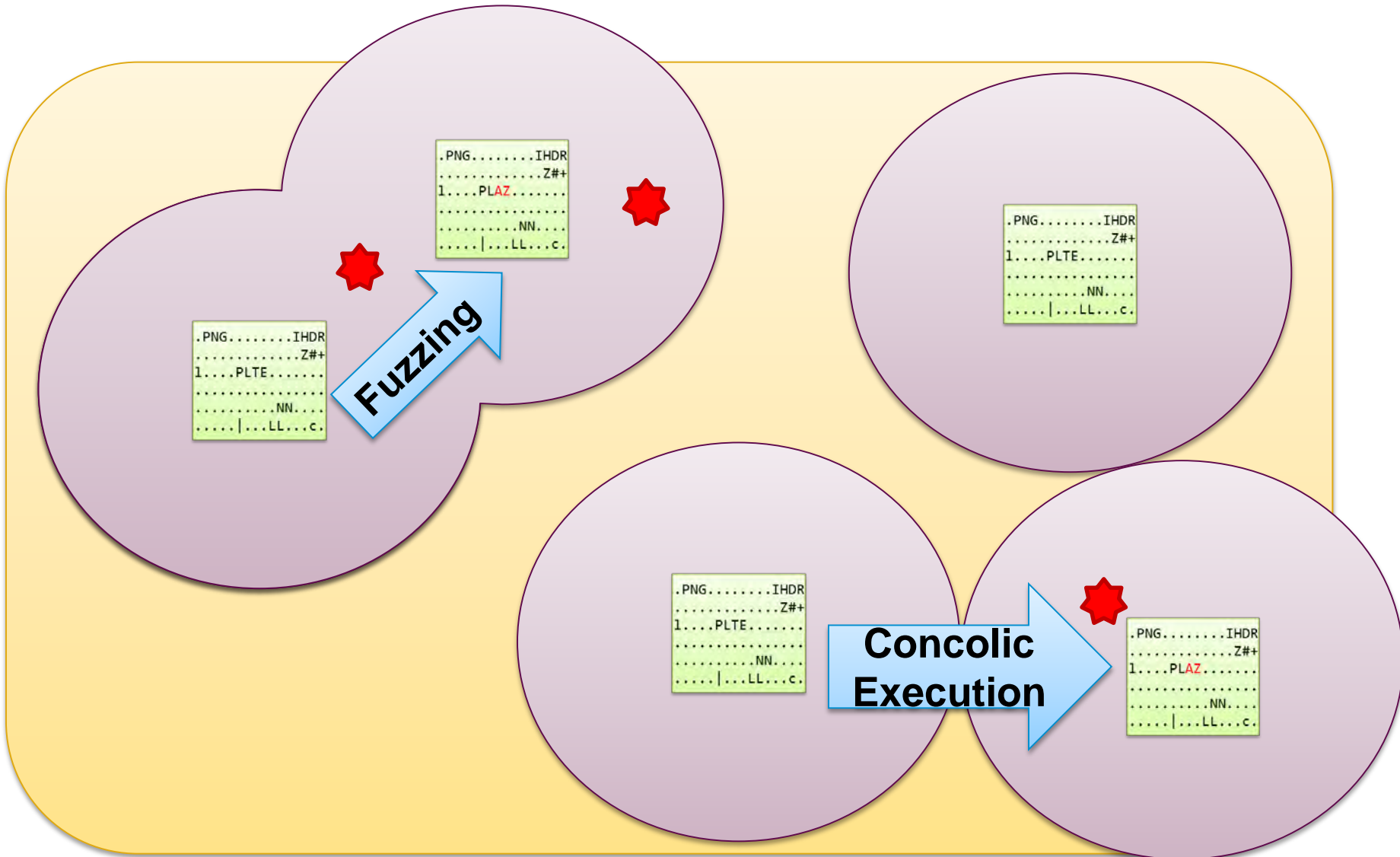
# The Patch Tree: Ability to Test Patches Independently



# Guided Fuzzing



# Fuzzing vs. Concolic Execution



# Combining Fuzzing and Concolic Execution

