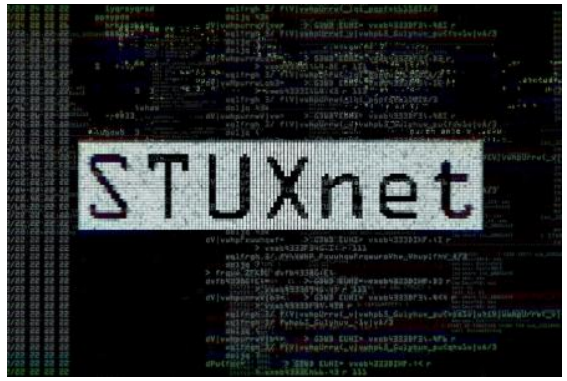
Understanding the Reproducibility of Crowd-reported Security Vulnerabilities

Dongliang Mu[12], Alejandro Cuevas[2], Limin Yang[3], Hang Hu[3],
Xinyu Xing[2], Bing Mao[1], Gang Wang[3]

1. Nanjing University
2. Pennsylvania State University
3. Virginia Tech

Real World Effects of Security Vulnerabilities



CVE-2010-2772
STUXnet

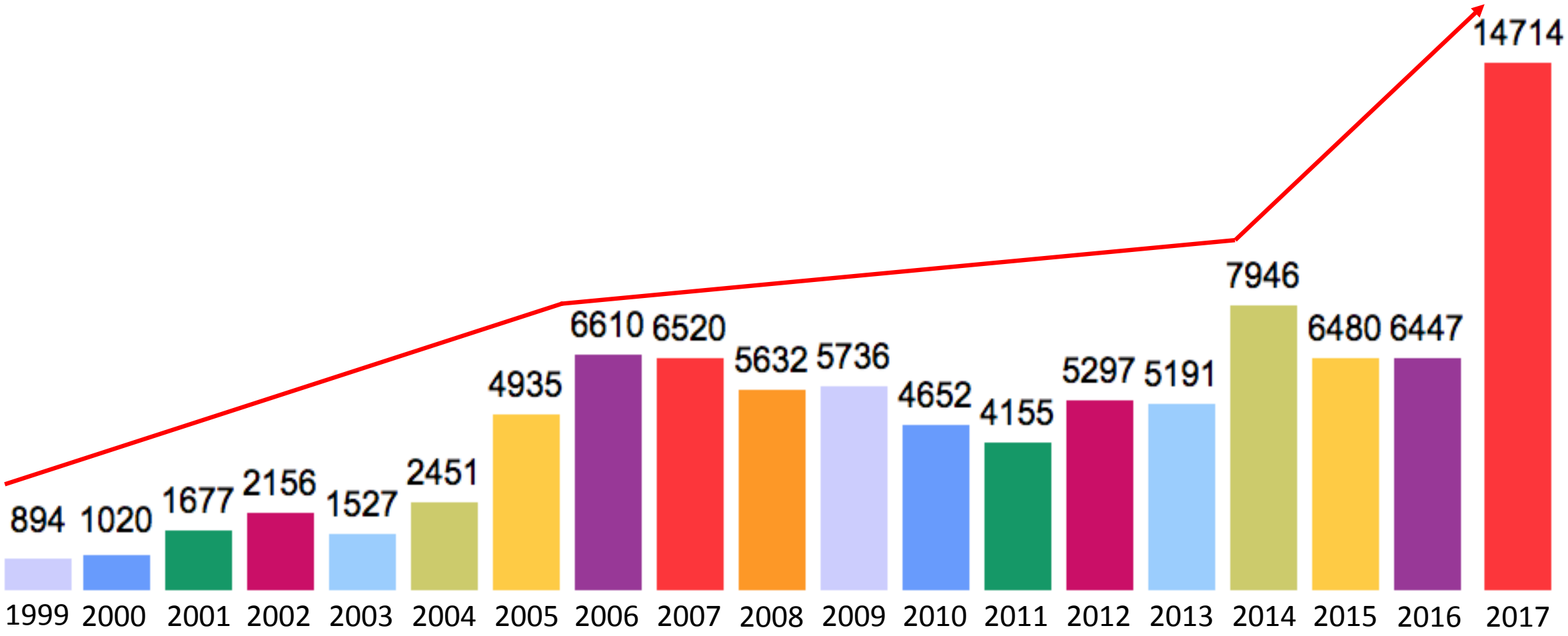
CVE-2014-0160
HeartBleed

CVE-2014-6271
ShellShock

CVE-2017-0144
WannaCry

It is infeasible for in-house teams to identify all possible vulnerabilities before a software release

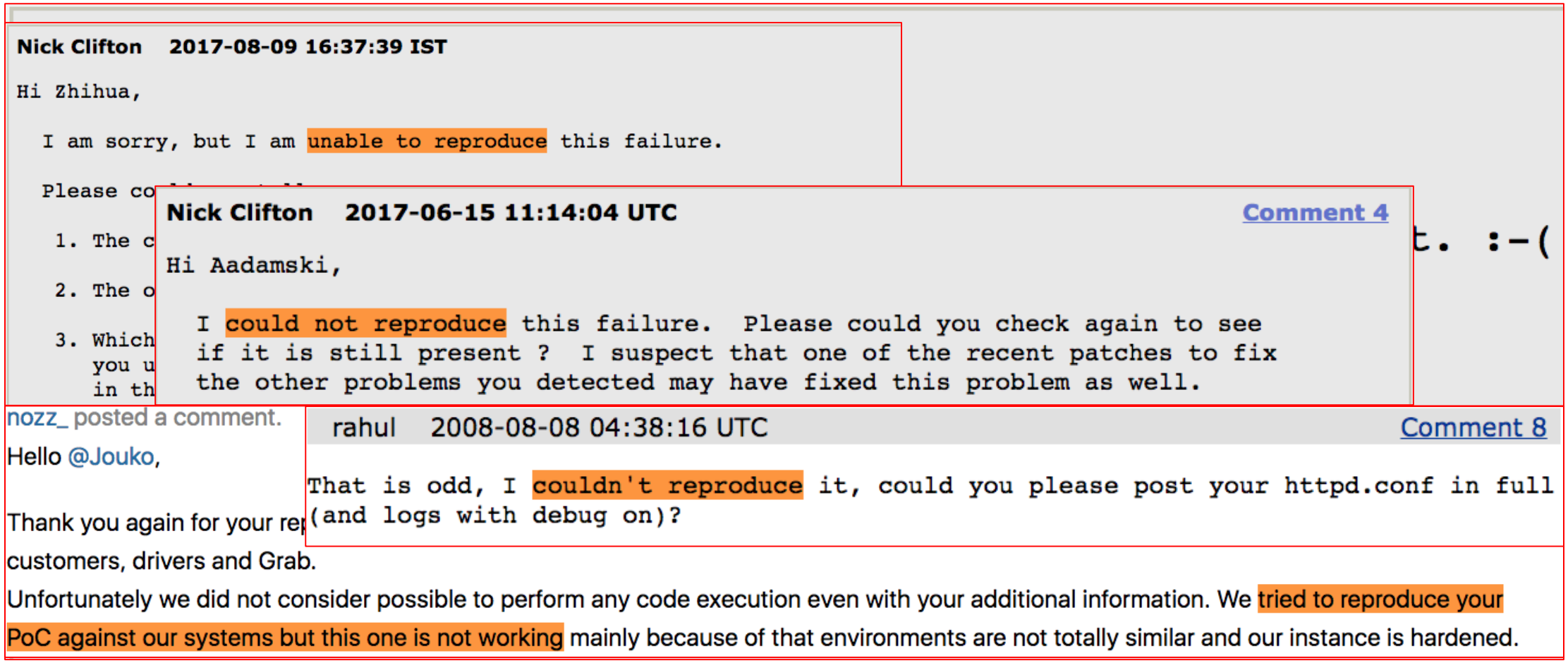
Massive Crowd-reported Vulnerabilities Over Time



Number of vulnerabilities reported to CVE[1] by year

[1] Common Vulnerabilities and Exposures

Vulnerability Reproduction Can Be Challenging



Nick Clifton 2017-08-09 16:37:39 IST

Hi Zhihua,

I am sorry, but I am unable to reproduce this failure.

Please co

1. The c
2. The o
3. Which
   you u
   in th

t. :-(

Nick Clifton 2017-06-15 11:14:04 UTC Comment 4

Hi Aadamski,

I could not reproduce this failure. Please could you check again to see if it is still present ? I suspect that one of the recent patches to fix the other problems you detected may have fixed this problem as well.

nozz_ posted a comment.

Hello @Jouko,

Thank you again for your re

customers, drivers and Grab.

Unfortunately we did not consider possible to perform any code execution even with your additional information. We tried to reproduce your PoC against our systems but this one is not working mainly because of that environments are not totally similar and our instance is hardened.

rahul 2008-08-08 04:38:16 UTC Comment 8

That is odd, I couldn't reproduce it, could you please post your httpd.conf in full (and logs with debug on)?

Consequences of Poor Reproducibility

Research Papers that use public vulnerabilities for evaluation	# of Vulnerability
SP'2018	9
Usenix'2017	8
Usenix'2015	6
NDSS'2015	7
Usenix'2015	8
NDSS'2011	14
SP'2008	5
Usenix'2005	4
Usenix'1998	8

Security Researchers

Poor reproducibility makes it hard to thoroughly evaluate security solutions

This Work

Q1: How reproducible are public security vulnerability reports?

Q2: What makes vulnerability reproduction difficult?

Q3: How to improve the efficiency of vulnerability reproduction?

We answer three questions by
manually reproducing vulnerabilities

Roadmap

- Methodology
- Findings
- ~~Survey~~
- Suggestions
- Conclusion

We surveyed 48 external security professionals from both academia and industry to examine people's perceptions towards the vulnerability reports and their usability

Vulnerability Report Dataset

- We randomly selected a large collection of reported vulnerabilities
  - We focused on Memory Error Vulnerabilities due to their high severity (Average CVSS Score 7.6 > Overall Average CVSS Score 6.2) and significant real-world impact
  - We focused on Open Source Linux Software due to debugging and diagnosing capabilities
- We collected two datasets including,
  - A primary dataset of 291 vulnerabilities with CVE IDs
  - A complementary dataset for 77 vulnerabilities without CVE ID

CVSS Score	Rating
0.1 - 3.9	Low
4.0 - 6.9	Medium
7.0 - 8.9	High
9.0 - 10.0	Critical

Vulnerability Report Dataset (cont.)

We collect vulnera ed in the CVE website.

Top 5 source websites in our dataset

EXPLOIT DATABASE

SecurityFocus.com

Red Hat Bugzilla

Security tracker SM

Openwall

CVE-2008-5 vulnerability reports

The Analyst Team

- We formed a team of 5 security analysts to carry out our experiments

Security Analysts

In-depth knowledge of memory error vulnerabilities

First-hand experience analyzing vulnerabilities, writing exploits, and developing patches

Rich Catch-The-Flag experience, and have discovered and reported over 20 new vulnerabilities to CVE website

Reproduction Workflow

Security Analysts

- Vulnerable Version
- Operating System
- Software Installation
- Software Configuration
- Proof-of-Concept File
- Trigger Method
- Vulnerability Verification

Default Setting for missing information

Reproduction Workflow (cont.)

- Set up the operating system for vulnerable software analysis

Information	Default Setting
Operating System	A Linux system that was released in (or slightly before) the year when the vulnerability was reported

- Vulnerable Version
- ✓ Operating System
- Software Installation
- Software Configuration
- Proof-of-Concept File
- Trigger Method
- Vulnerability Verification

Reproduction Workflow (cont.)

- Compile vulnerable software with the compilation options
- Install vulnerable software with the configuration options

Building System	Default Setting
automake	make; make install
autoconf & automake	./configure; make; make install
cmake	mkdir build; cd build; cmake ../; make; make install

- ✓ Vulnerable Version
- Operating System
- ✓ Software Installation
- ✓ Software Configuration
- Proof-of-Concept File
- Trigger Method
- Vulnerability Verification

Reproduction Workflow (cont.)

- Trigger the vulnerability by using the Proof-of-Concept File

Type of PoC	Default Setting
Shell commands	Run the commands with the default shell
Script program (e.g., python)	Run the script with the appropriate interpreter
C/C++ code	Compile code with default options and run it
A long string	Directly input the string to the vulnerable program
A malformed file (e.g., jpeg)	Input the file to the vulnerable program

- Vulnerable Version
- Operating System
- Software Installation
- Software Configuration
- ✓ Proof-of-Concept File
- ✓ Trigger Method
- Vulnerability Verification

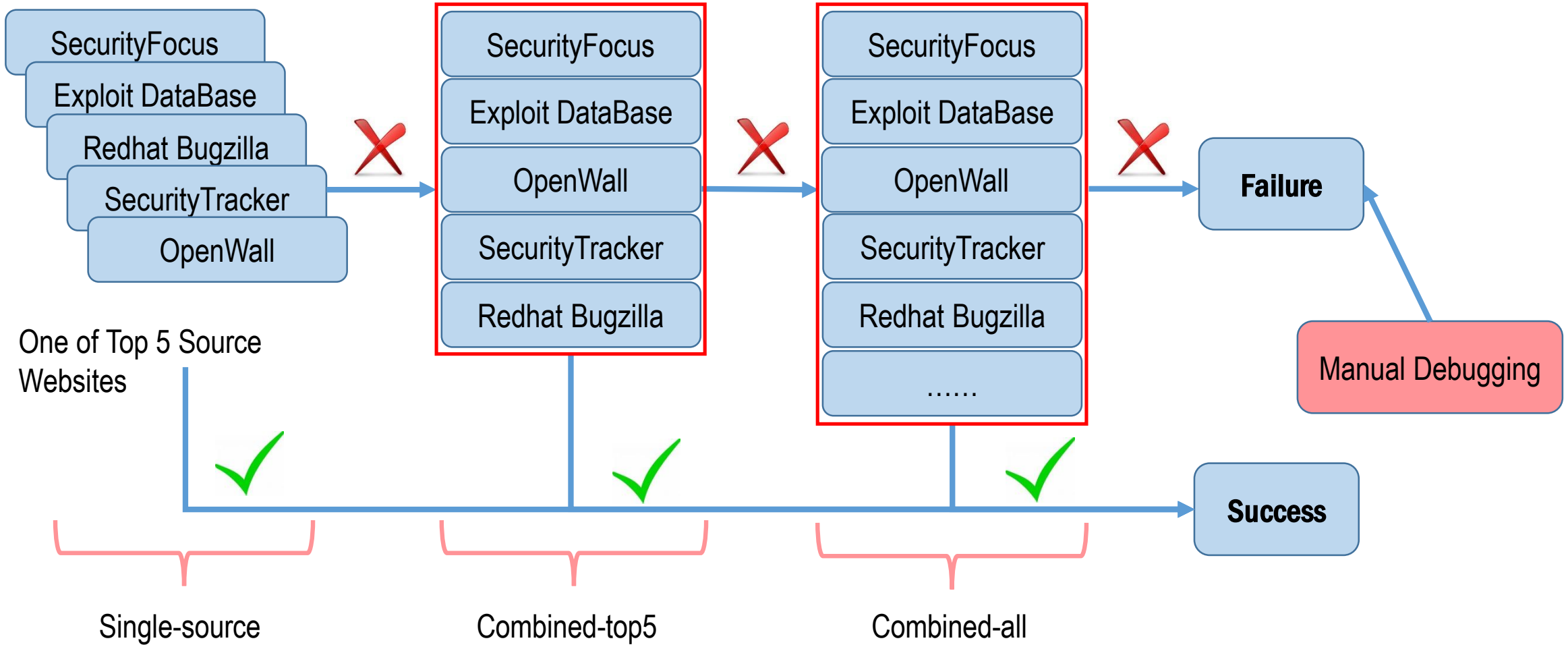
Reproduction Workflow (cont.)

- Verify the vulnerability with expected program behavior

Information	Default Setting
Vulnerability Verification	Unexpected program termination (or program “crash”)

- Vulnerable Version
- Operating System
- Software Installation
- Software Configuration
- Proof-of-Concept File
- Trigger Method
- Vulnerability Verification

Reproduction Experiment: Controlled Information Source



Roadmap

- Methodology
- Findings
- Suggestions
- Conclusion

Finding 1: Vulnerability Is Difficult to Reproduce

Information Source	CVE Reproduction (N=291)		
	# of	# of Success	Success Rate
SecurityFocus	256	32	12.6%
Redhat Bugzilla	195	19	9.7%
ExploitDB	156	46	29.5%
OpenWall	153	67	43.8%
SecurityTracker	89	4	4.5%
Combined-top5	287	126	43.9%
Combined-all	291	182	62.5%

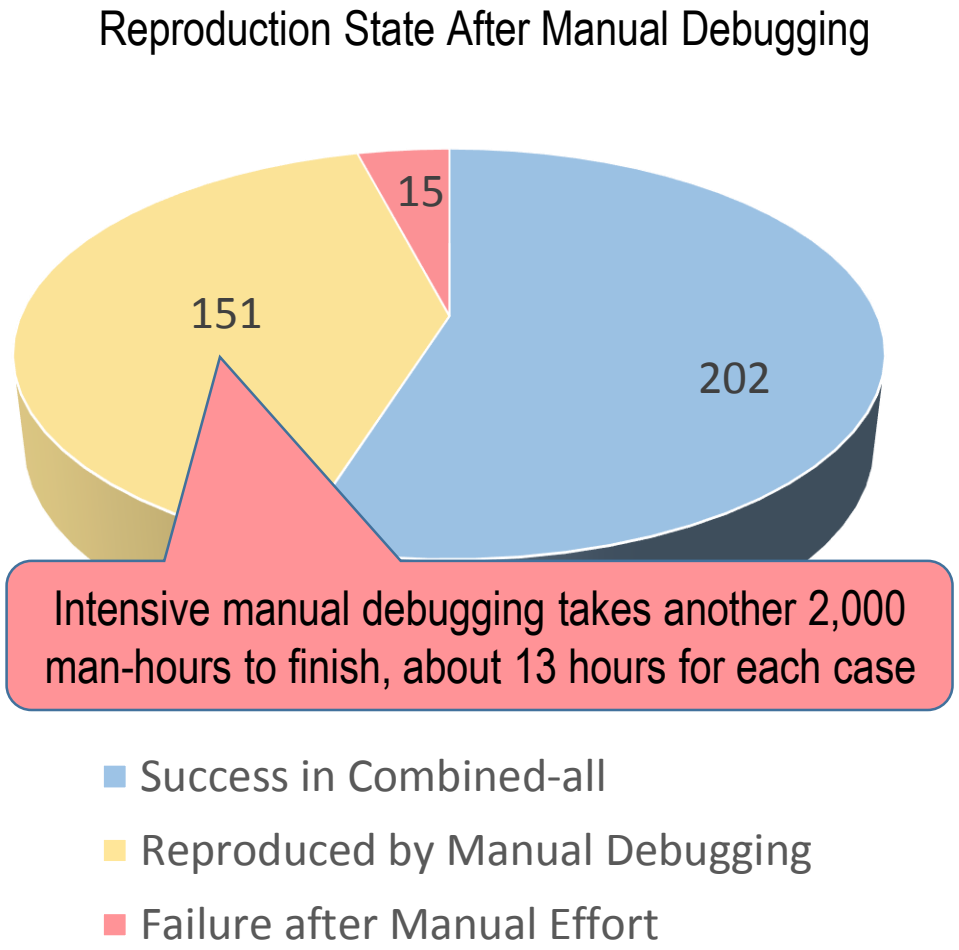
Information Source	Non-CVE Reproduction (N=77)		
Combined-all	77	20 (25.6%)	25.6%

The single-source returns a low success rate

"Combined-top5" has clearly improved the success rate

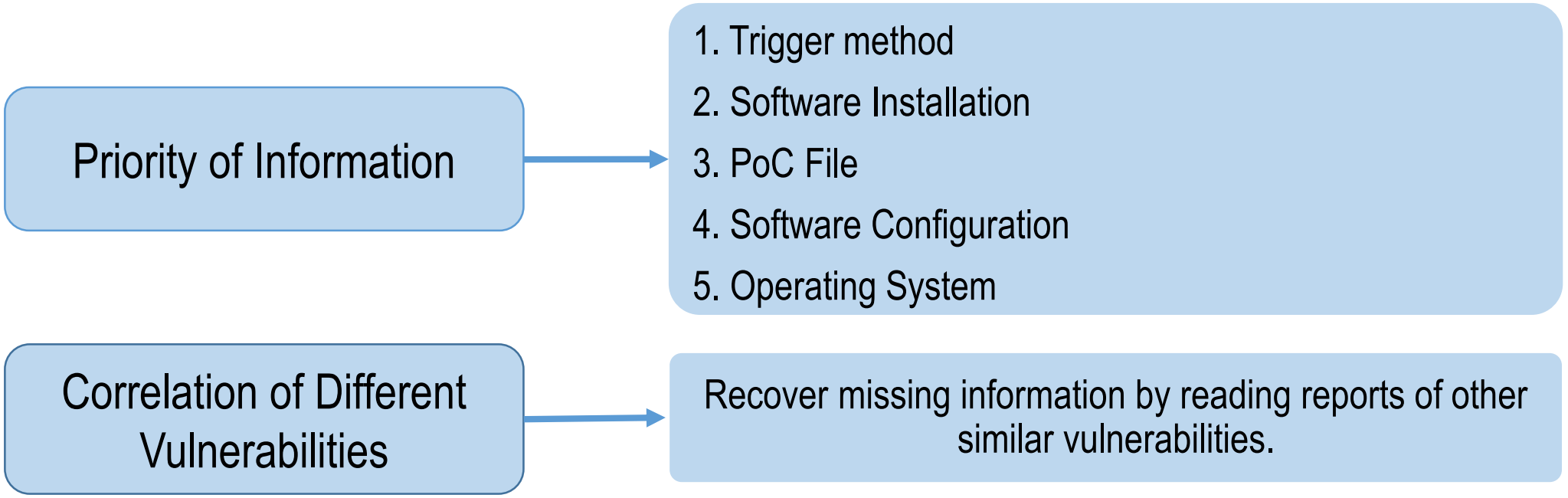
The success rate is improved to 62.5% by "Combined-all"

Finding 2: Key Factors Make Reproduction Difficult



Report Information	# of vulnerabilities addressed by Manual Debugging
Trigger Method	74
Software Installation	43
PoC File	38
Software Configuration	6
OS information	4
Software version	1
Vulnerability Verification	0

Finding 3: Useful Tips for Information Recovery



For 74 cases that failed on trigger method, we recovered 68 cases by reading other similar vulnerability reports

Roadmap

- Methodology
- Findings
- Suggestions
- Conclusion

Our Ideas of Making Vulnerability Reproduction Easier

Vulnerability Reporters

Reporting Websites

Reproducers

CVE-2007-1001 misses Trigger Method
CVE-2013-7226 misses Installation Options
CVE-2007-1465 misses Proof-of-Concept
.....

Manually generating standardized reports is really time-consuming

With standardized reports, it's a waste of resource if we still reproduce vulnerability entirely by manual efforts

1 Standardize Vulnerability Reports

2 Develop Useful Automated Tools to Collection Information

3 Automate the Vulnerability Reproduction

Conclusion

Vulnerability reproduction is difficult and requires extensive manual efforts

A crowdsourcing approach could increase the reproducibility

Apart from manual debugging based on experience, Internet-scale crowdsourcing and some heuristics could help recover missing information

There is an urgent need to automate vulnerability reproduction and overhaul current vulnerability reporting systems

Data Sharing

- DataSet : https://vulnreproduction.github.io/ (12 Virtual Machine Images)
- Github Repo : https://github.com/VulnReproduction/LinuxFlaw
- We provide 300+ Reproducible Vu
- For each vulnerability, we have :
  - Fully-tested Proof-of-Concept
  - Pre-configured virtual machine or Docker Image
  - Detailed instructions on how to reproduce the vulnerability
  - Structured information fields (in HTML and JSON)

Name: Dongliang Mu
Homepage: http://mudongliang.me/about/
Email: dzm77@ist.psu.edu

References

	Research Papers that use public vulnerabilities for evaluation	# of Vulnerability
Usenix'2005	Non-control-data attacks are realistic threats	4
SP'2008	Preventing memory error exploits with wit	5
Usenix'2015	Control-flow bending: on the effectiveness of control-flow integrity	6
NDSS'2015	Preventing Use-after-free with Dangling Pointers Nullification	7
Usenix'1998	StackGuard : automatic adaptive detection and prevention of buffer-overflow attacks	8
Usenix'2017	Towards efficient heap overflow discovery	8
Usenix'2015	Automatic Generation of Data-Oriented Exploits	8
SP'2018	Data-oriented programming : On the Expressiveness of Non-Control Data Attacks	9
NDSS'2011	AEG: Automatic exploit generation	14

Finding : Vulnerability Reports Are Often Incomplete

Experiment Strategy		Vulnerability Reports for CVE with Missing Information						
	Covered CVE IDs	Software Version	Software Installation	Software Configuration	OS Information	PoC File	Trigger Method	Vulnerability Verification
SecurityFocus	256	9	255	233	116	131	210	227
Redhat Bugzilla	195	48	195	179	0	154	168	147
ExploitDB	156	5	155	137	132	20	100	111
OpenWall	153	28	152	140	153	72	72	71
SecurityTracker	89	3	87	71	73	69	62	61
Combined-top5	287	3	284	259	55	70	125	138
Combined-all	291	1	280	256	52	17	82	106

Case Study

GAS CVE-2005-4807 misses ***Trigger Method***

Solution : C Code should be compiled to malicious assemble file by "gcc -S", and then feed it to GAS.

libgd CVE-2013-7226 misses ***Installation Information***

Solution: **"--with-gd" is a necessary compilation option to build PHP**

coreutils CVE-2013-0221 misses ***Environment Information***

Solution: patch vulnerable "coreutils-i18n.patch" applied in OpenSUSE by default