OWASP

AppSec USA
San Jose, CA
8-12 October 2018


Human factors that influence secure software development

Dr. Anita D'Amico
Chris Horn

About the speakers

Dr. Anita D'Amico

CEO, Code Dx
Director, Secure Decisions

Chris Horn

Product management at Code Dx
Researcher, Secure Decisions

appliedvisions

↓

SECURE DECISIONS
A DIVISION OF APPLIED VISIONS, INC.

↓

CodeDx

Applied Visions, Inc.

- Software development since 1987
- Primarily develops business applications

dba, Secure Decisions

- Cyber R&D, focusing on application security
- Primarily serving DHS and DoD, some commercial companies

Code Dx, Inc.

- Spin-out to commercialize DHS-funded AppSec R&D

Outline of today's talk

PART I **Introduction**

- Problem
- Motivation for looking into human factors of software engineering and AppSec
- Our DARPA research project

PART II **Human factors that suggest where vulnerabilities are more likely**

- Developer behaviors & characteristics
- Environmental conditions
- Team characteristics

PART III **Wrap-up**

- Discussion about future ways to study human factors in software engineering

Introduction

Part I

A single software vulnerability can cascade into tens of thousands of cyber incidents

Heartbleed

- Two years to detect in OpenSSL
- Estimated that 500,000 web servers were affected [1]

Equifax breach

- Single Struts vulnerability CVE-2017-5638
- Exposed personal information of 147.9 million Americans [2]

EQUIFAX

Heartland Payment Systems

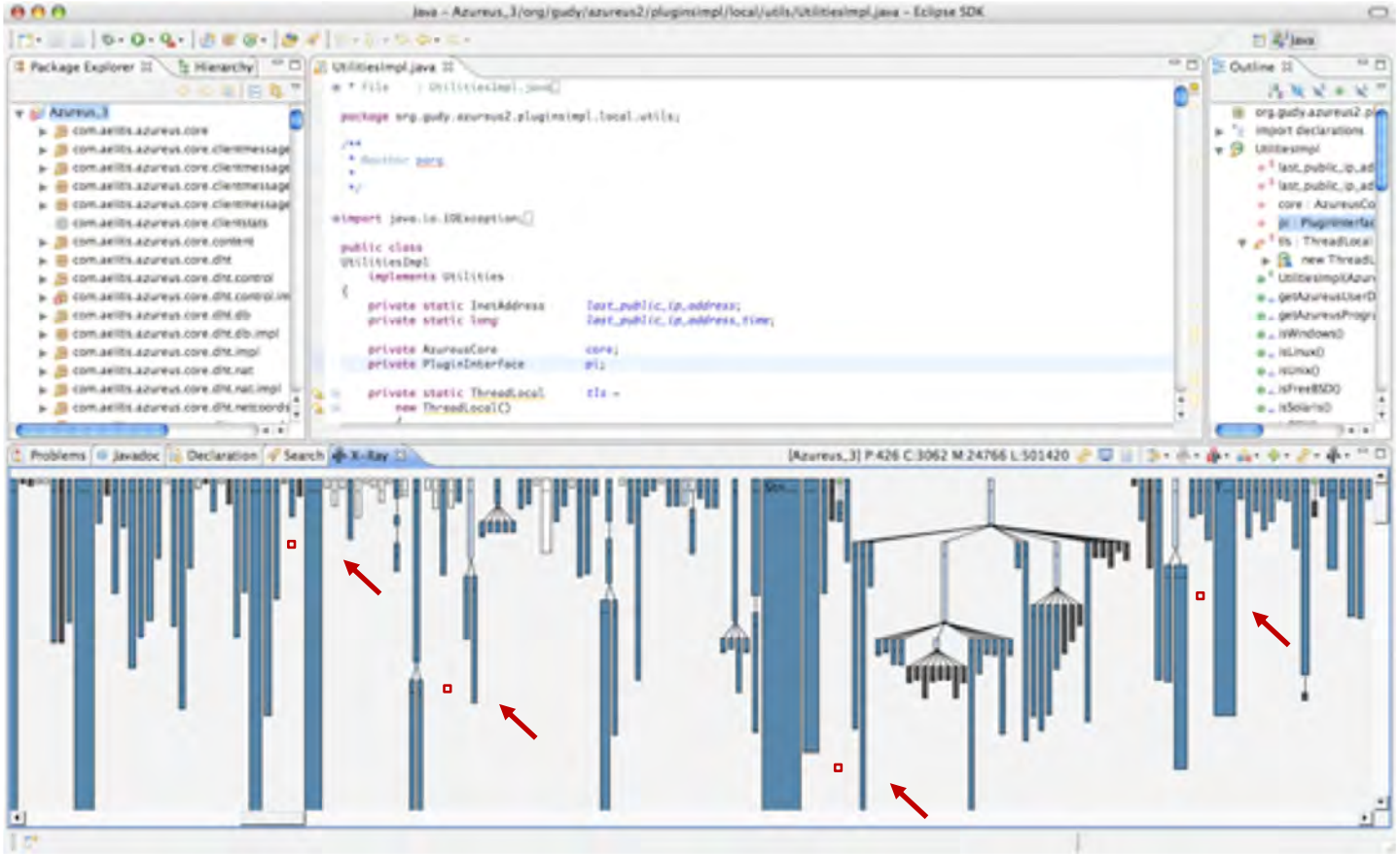
- SQL injection vulnerability known for years
- Exposed 134 million credit cards [3]

Heartland

And many more…

[1] Based on Synopsys estimate using 2014 Netcraft Web Server Survey data, http://heartbleed.com/
[2] B. Fung, "Equifax's massive 2017 data breach keeps getting worse," *Washington Post*. [Online].
[3] "Heartland Payment Systems," Wikipedia. 02-Aug-2018.

What if we could identify code more likely to be vulnerable based on the *human factors* in play when it was written?



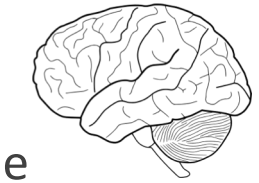
AppSec analysts & developers could better focus on suspect code

- Manual code review
- SAST finding investigation

Development managers could foster work environments more conducive to secure code development

Human factors are properties of people

Psychological



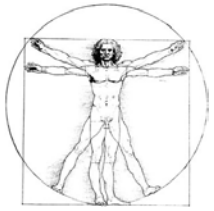
Individual

- Reaction time
- Attention
- Learning
- Short term & long term memory
- Decision making

Group

- Collaboration & conflict
- Communication
- Cultural norms & conventions
- And many more…

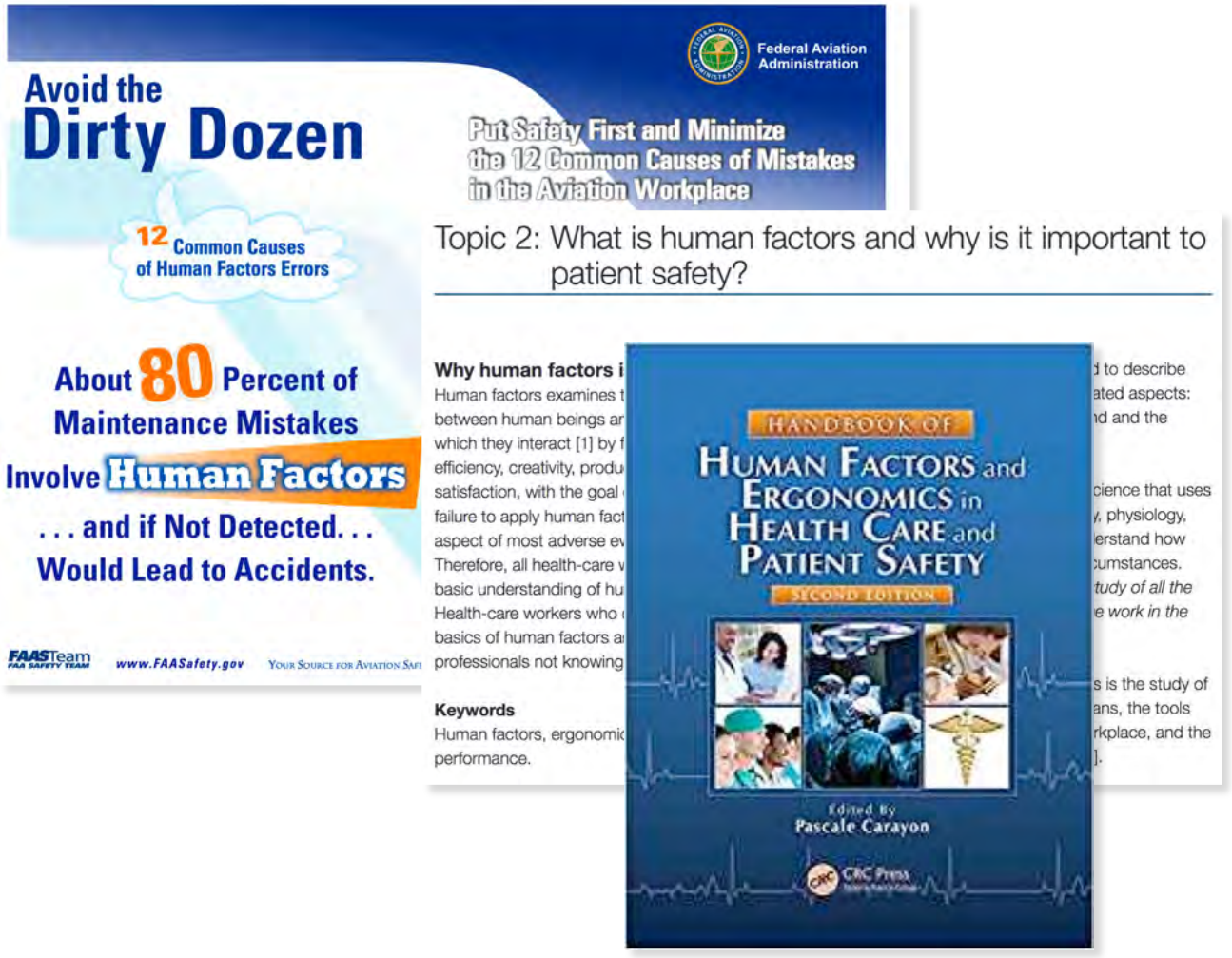
Physiological



- Visual acuity
- Hearing sensitivity
- Fatigue
- Circadian rhythm
- Endurance
- Strength
- Temperature tolerance
- Health
- And many more…

Human factors psychology and engineering account for these factors in the design and engineering of products, processes, and systems

Human factors are widely known to affect performance & safety



Transportation

- Federal Aviation Administration publishes “Dirty Dozen” list of 12 human factors that lead to accidents
- National Transportation Safety Board performs root cause analysis accident investigations

Medicine & healthcare

- World Health Organization, National Institutes of Health (NIH), and others publish educational materials

Occupational safety

We've been studying how human factors affect secure code development

We wanted more ways to prevent the introduction of vulnerabilities, and different ways to locate vulnerable code

- Are there specific characteristics of developers, teams or work environment that influence secure code development?
- Human factors are known to influence safety and security in other domains
- Human factors are common across people, & many are stable over time

We proposed a research project to DARPA

- Investigate human factors that affect the security of application source code
- Awarded Phase I SBIR late last year – 9 months – to see if methods and results were promising
- Recently selected to continue the research into Phase II – 2 years starting in 2019

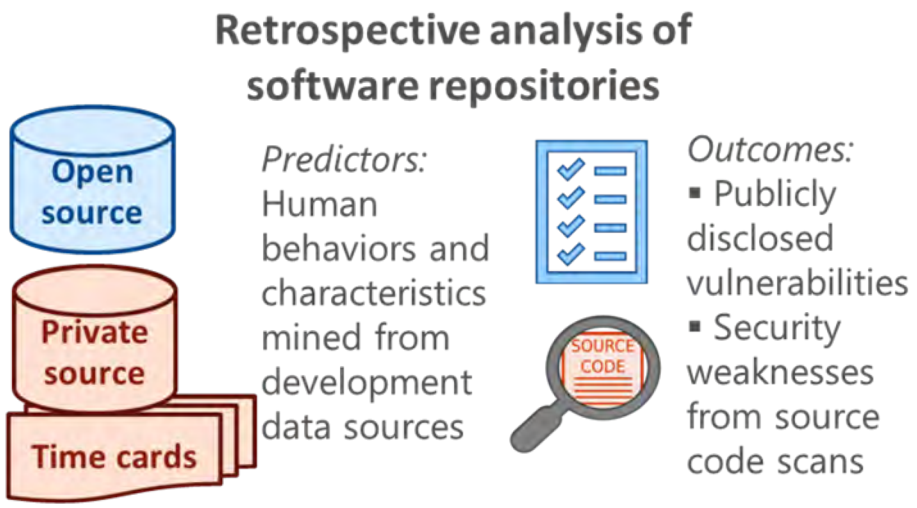
Overview of our DARPA-funded research

Identify human factors that correlate with the security & quality of source code

- Developer behaviors & characteristics
  - Experience / training
  - File editing behavior
  - Focused attention
  - Communication form and quality
  - Hours worked
- Team characteristics
  - Number of developers
  - Collaboration
  - Longevity as a team
  - Geographic proximity
- Environmental conditions
  - Programming language
  - Open vs. proprietary development
  - Ambient noise
  - Interruptions

We used two measures of security & quality:

1. Publicly disclosed vulnerabilities
2. Security and select quality weaknesses found by static application security tests (SAST)



We analyzed repositories and data from projects developed in both open source and proprietary SE environments

SE environment	Name of repository	Portion of repository studied	Primary language	Lines of code	% repo in primary lang.	# of commits
Open	Chromium	ui/base directory	C++	38,335	96.92%	3,880
Open	Apache HTTP	server directory	C	42,969	98.47%	4,701
Proprietary	ChatSecure Android	All	Java	37,213	99.63%	2,907
Proprietary	Project D	All	C#	289,993	98.87%	808 *

* Only a subset of the total 3,523 commits were analyzed because C# static code analyzers require the software to be built before testing, which would have required more time and resources than available in Phase I

Studied relationships over both multi-year and 6-month periods

Repository	Multi-year period dates	Range duration	6-month period dates
Chromium	1/2011 – 11/2017	6 years	7/2016 – 1/2017
Apache HTTP	6/1999 – 2/2018	19 years	7/2016 – 1/2017
ChatSecure Android	3/2010 – 1/2018	8 years	2/2015 – 8/2015
Project D	8/2014 – 8/2016	2 years	2/2015 – 8/2015

Human factors

Part II

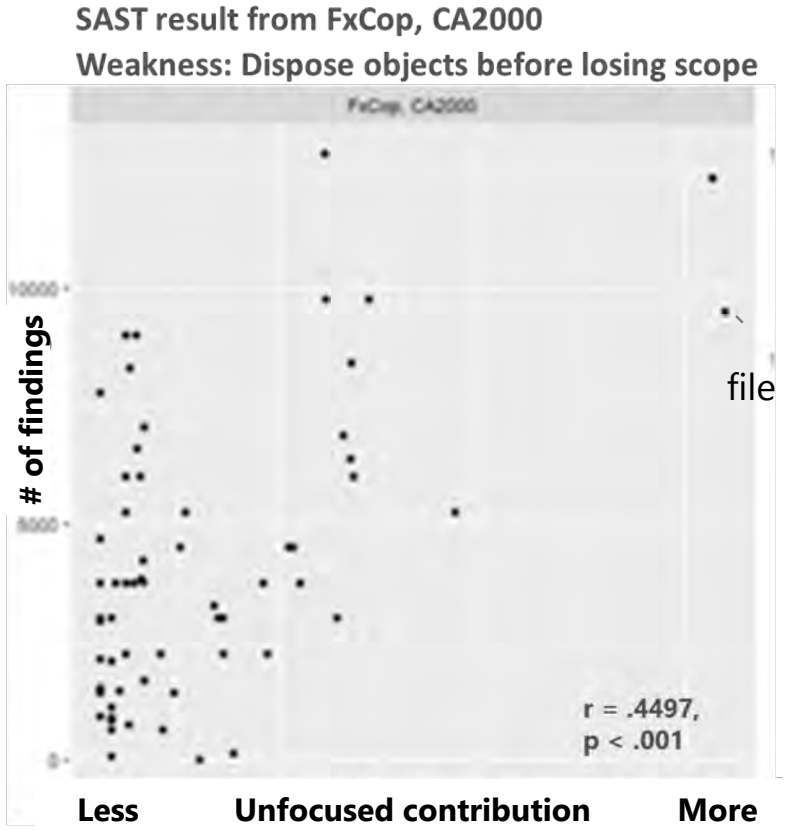
Diffusion of developer attention across files is associated with insecure code

Unfocused contribution is an indicator of how much attention developers focus on specific files being developed

- A file has high unfocused contribution when:
  - Developers of a file are also busy modifying *other* files, or
  - When the number of unique contributors to a file increases
- Measured using a PageRank centrality score

More unfocused contribution → more insecure code

- In multi-year analyses of all four repositories, more diffuse attention correlated with:
  - Greater likelihood of publicly disclosed vulnerabilities
  - More SAST findings, by type
  - More SAST findings, across all types
- Results less clear for individual SAST types in 6-month period of repository activity



Multi-tasking and excessive work have small, significant effects on source code security

Context switching was measured using number of distinct billing codes charged on time card

Excessive work was measured by number of work hours recorded by each developer

More SAST findings found in source code that was committed after periods when developers:

- Distributed their time across more unique billing codes, i.e. more context switching or multi-tasking
- Worked more hours

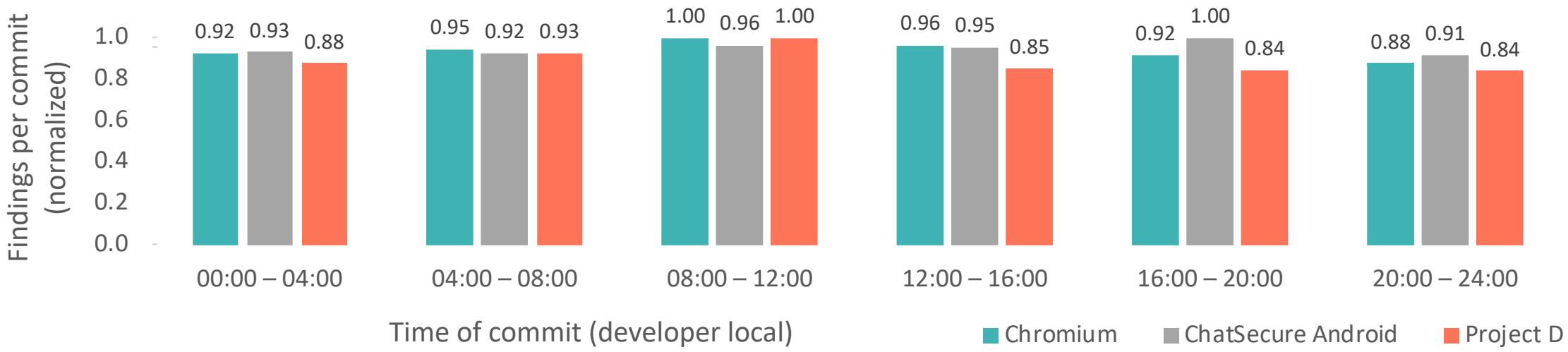
	Spearman rank correlation between predictor and number of new & unresolved SAST findings, multi-year period for AVI proprietary project	
Predictor	Number of charge codes in 7 days prior to commit	Number of charge codes since last commit
Unique billing codes	rho = 0.17 p-value < 0.01	rho = 0.12 p-value < 0.01
Number of hours billed	rho = 0.19 p-value < 0.01	rho = 0.11 p-value < 0.01

Time of day when code is committed is not always associated with its security

Time of day of developer activity measured using six 4-hour periods:

- Time of commit, developer's local time zone

No correlation was found between commit time period and the number of weaknesses found by SAST tools

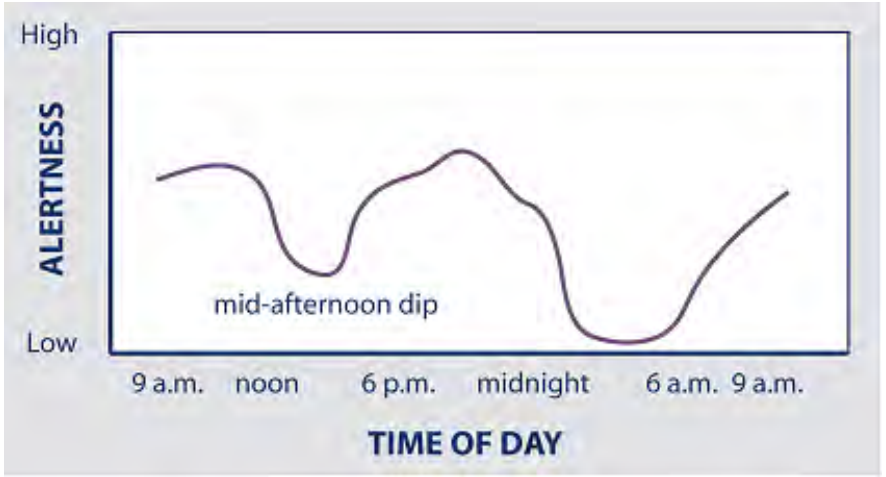


Time of day when code is committed correlates with some security outcome measures

In Chromium, files with a vulnerability in their history had significantly more of their lifetime churn committed during typically low-alertness times of day

	Significant Mann-Whitney-Wilcoxon test result, multi-year period for Chromium
Commit time, developer's local time of day	Significantly more median churn in files with a vulnerability than in neutral files *
0—4	✓
4—8	
8—12	
12—16	✓
16—20	
20—24	✓

* Neutral files are those with no publicly disclosed vulnerability



Notional chart of typical circadian rhythm

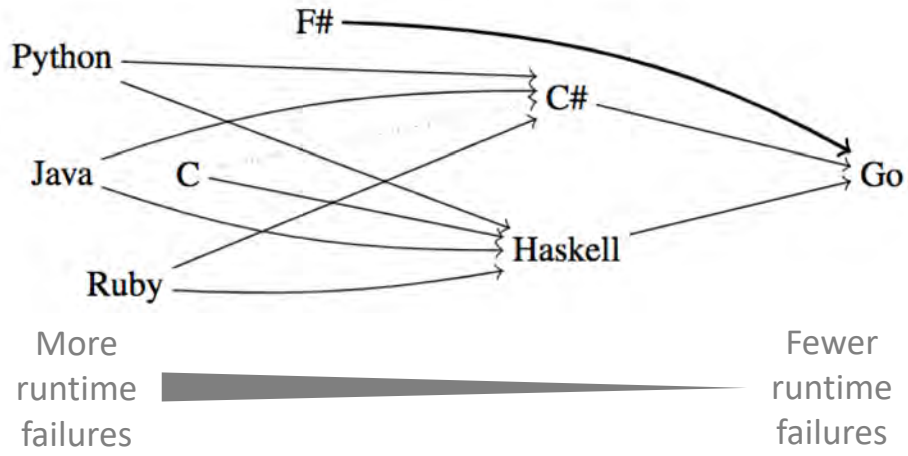
Environmental conditions

Programming language might affect software error rate

Mixed evidence that language affects quality

- "C++ code is less complex, less prone to errors and requires less effort to maintain [than C code]" [1]
- "Compiled strongly-typed languages are significantly less prone to runtime failures than interpreted or weakly-typed languages" [2]
- Other researchers have *not* found correlations [2]

It's difficult to control for developer experience



Compiled, strongly-typed languages are less prone to runtime failures [2]

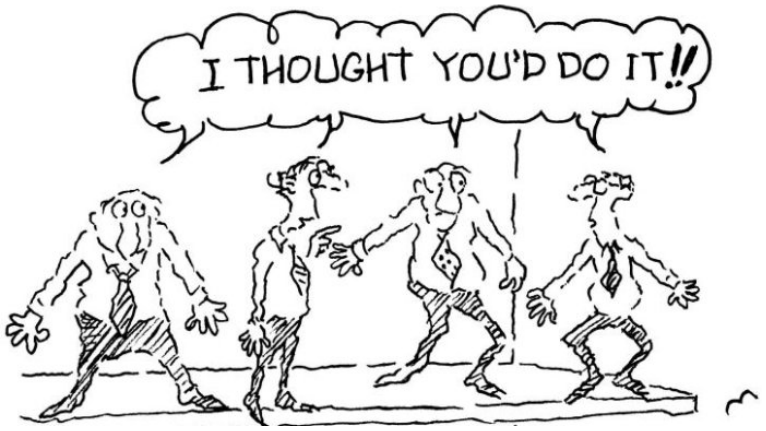
[1] P. Bhattacharya and I. Neamtiu, "Assessing Programming Language Impact on Development and Maintenance: A Study on C and C++," p. 10, May 2011.
[2] S. Nanz and C. A. Furia, "A Comparative Study of Programming Languages in Rosetta Code," 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, pp. 778–788, May 2015.

Bigger teams are correlated with less secure code

Team size is simply the number of developers

The more developers that work on a file, the more likely it is to have security weaknesses and vulnerabilities

- In two open source repositories that we analyzed (see table)
- In two proprietary repositories that we analyzed
  - Files with more developers have more SAST findings, in both the multi-year and 6-month sample windows
- In the Linux kernel, source code "files with changes from nine or more developers were 16 times more likely to have a vulnerability" [1, p. 453]
- At Microsoft, Windows Vista study



Perhaps a *bystander effect*?

	Median # of developers, multi-year open source repo	
	Files with at least one vulnerability	Files with no vulnerabilities
Chromium	7 developers	1 developer
Apache HTTP	15 developers	1 developer

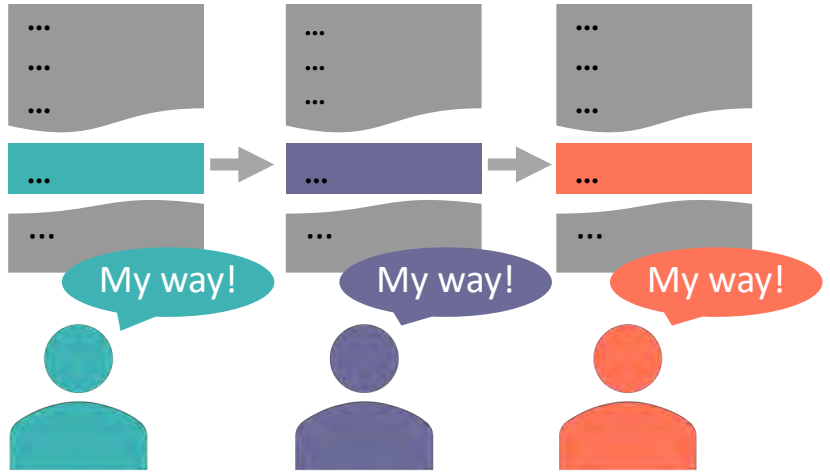
[1] A. Meneely and L. Williams, "Secure Open Source Collaboration: An Empirical Study of Linus' Law," in Proceedings of the 16th ACM Conference on Computer and Communications Security, New York, NY, USA, 2009, pp. 453–462 [Online]. Available: http://doi.acm.org/10.1145/1653662.1653717

Lots of editing of others' code is associated with vulnerabilities in open source software

Interactive churn is the number of source code lines that a developer modifies that were last modified by another developer

Files with a known vulnerability in their history have more interactive churn than files without a known vulnerability

- Chromium: almost 5 times more interactive churn
- Apache Web Server: >11 times more interactive churn



In ChatSecure Android interactive churn found more findings for 16% of SAST weaknesses types

	Interactive churn median SLOC, multi-year period	
	Files with one or more vulnerabilities	Neutral files *
Chromium	1,201 SLOC	250 SLOC
Apache HTTP	5,265 SLOC	447 SLOC

* Neutral files are those with no publicly disclosed vulnerability

Lack of collaboration increases likelihood of vulnerabilities

Files worked by multiple, separate clusters of developers are more likely to be vulnerable [2, p. 783], [3, p. 460]

Developers working too much on only their own code increases the chance of vulnerabilities [1, p. 71]

[1] A. Meneely, H. Srinivasan, A. Musa, A. R. Tejeda, M. Mokary, and B. Spates, “When a Patch Goes Bad: Exploring the Properties of Vulnerability-Contributing Commits,” in 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, 2013, pp. 65–74.
[2] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, “Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities,” IEEE Trans. Softw. Eng., vol. 37, no. 6, pp. 772–787, 2011.
[3] A. Meneely and L. Williams, “Secure Open Source Collaboration: An Empirical Study of Linus’ Law,” in Proceedings of the 16th ACM Conference on Computer and Communications Security, New York, NY, USA, 2009, pp. 453–462 [Online]. Available: http://doi.acm.org/10.1145/1653662.1653717

Co-location of team members did not materially affect code quality at Microsoft

Microsoft studied the post-release failure rate for Windows Vista binaries authored by co-located and distributed teams

- Determined team co-location based on whether 75% of developers committing to the binary's source code shared the same building, cafeteria, campus, locality, or continent

Virtually same number of failures between co-located and distributed teams

- Binaries authored by distributed teams had ≤6% higher rate of failure

Found that only team size could explain differences between binary failure rates

[1] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy, "Does distributed development affect software quality? An empirical case study of Windows Vista," 2009, pp. 518–528.

Wrap-up

Part III

When we can use human factors to identify code more likely to be vulnerable, then …

Analysts triaging SAST findings can focus on code written under work conditions or other human factors associated more insecure code

Manual code reviews can focus on code written by developers or dev teams whose characteristics (e.g. team size) or environment (e.g. time of day) suggest vulnerabilities are more likely

Development managers can change the work environment to be more conducive to secure code development

Discussion time! How can we further assess human factors that influence code security and quality?

Some ideas:

- Concurrent analysis of developers as they code
  - Capture data about developers that's otherwise lost: e.g. noise, fatigue, team communications, new training
  - Could be done in proprietary development environments
  - Could it be done in open source environments?
- Vulnerability history project (Rochester Institute of Technology)
  - Database of historical context & root cause analysis of open source vulnerabilities
- Hackathons
  - Could experimentally control factors such as goals, team size & diversity, level of collaboration, type and degree of distraction
- Embed security champions into open source development teams, then measure effects of influencers

Concurrent analysis of software being developed

More Predictors and Outcomes: Collected from instrumented development environment

Contact information

Dr. Anita D'Amico

CEO, Code Dx
Director, Secure Decisions

@AnitaDamico
Anita.DAmico@CodeDx.com

Chris Horn

Product management at Code Dx
Researcher, Secure Decisions

@chornsec
chorn@codedx.com

LET US KNOW

If you would like to participate in this type of research.
If you have a software repository you would like us to study.