Can Knowledge of Technical Debt Help Identify Software Vulnerabilities?

Robert L. Nord, Ipek Ozkaya, Edward J. Schwartz, Forrest Shull, Rick Kazman

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Software Engineering Institute | Carnegie Mellon University

DM-0003863

Security Issues

Crash - WebCore::TransparencyWin::initializeNewContext()

Project Member Reported by lafo...@chromium.org, Apr 24, 2009

```
This crash was detected in 2.0.176.0-qemu and was seen i
It is currently ranked #10 (based on the relative number
been 3 reports from 3 clients.
```

10977: Crash due to large negative number.

"*We could just fend off negative numbers near the crash site or we can dig deeper and find out how this -10000 is happening.*"

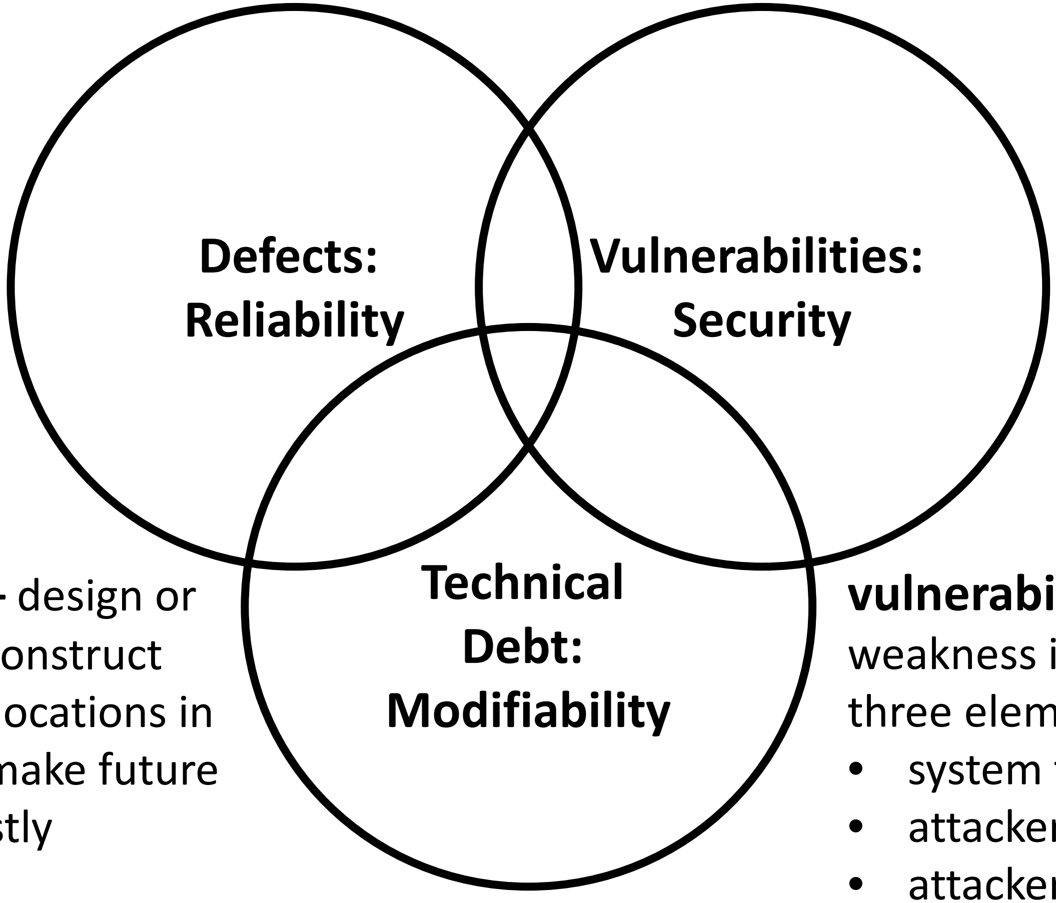
"Time permitting, I'm inclined to want to know the root cause. My sense is that if we patch it here, it will pop-up somewhere else later."

“There have been 28 reports from 7 clients… 18 reports from 6 clients”

“hmm ... reopening. the test case crashes a debug build, but not the production build. I have confirmed that the original source code does crash the production build, so there must be multiple things going on here.”

What are we measuring?

defect – error in coding or logic that causes a program to malfunction or to produce incorrect/ unexpected results



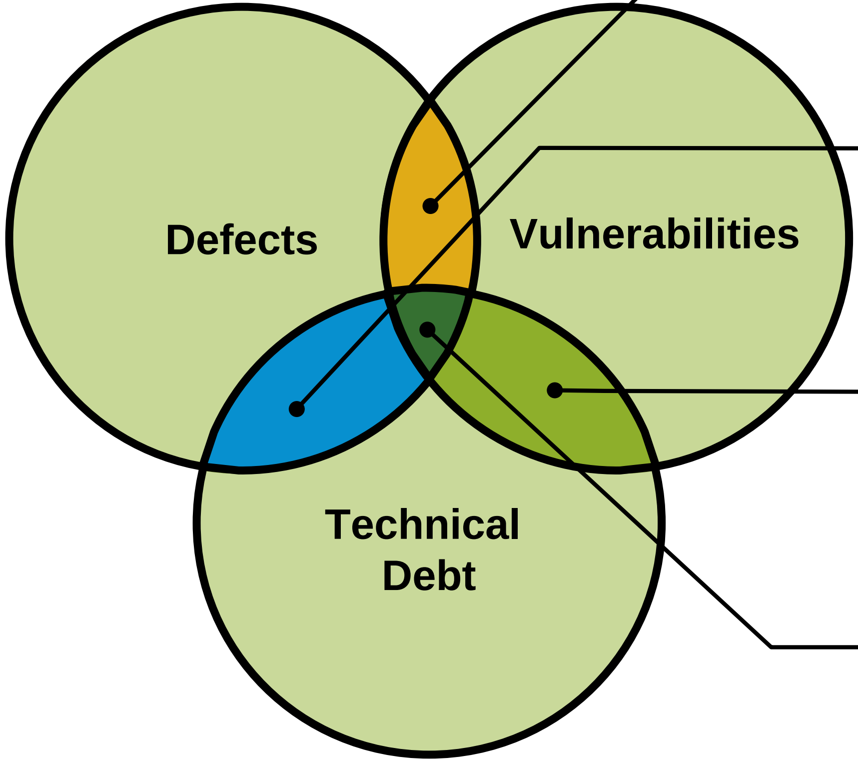
technical debt – design or implementation construct traced to several locations in the system, that make future changes more costly

vulnerability – system weakness in the intersection of three elements:

- system flaw,
- attacker access to the flaw,
- attacker capability to exploit the flaw

Compare with IEEE Std 1044-2009:
IEEE Standard Categorization for Software Anomalies.

Technical debt



Not all defects are vulnerabilities, but defect proneness does imply increased vulnerability risks

Defects are not technical debt, but technical debt as it lingers in the system increases defect proneness.

Similarly, technical debt increases vulnerability risks.

Some issues just overlap, making it hard to tease apart!

Research question

Are software components with accrued technical debt more likely to be vulnerability-prone?

Data Set

Chromium version:	17.0.963.46
Released:	February 8, 2012
Files:	18,730; 11k files with bugs 289 files with vulnerabilities
Issue range:	Feb 1, 2010 – Feb 8, 2012
Issues:	#bug: 14k; #security: 79

Chromium project

- Began in 2008
- Complex web-based application that operates on sensitive information and allows untrusted input from both web clients and servers.

Approach

Identify software vulnerabilities
- security label
- identify indicator from issue: CWE
- trace to file

Identify technical debt
- apply classification rules to issues
- extract design problem and rework from issues
- trace to file
- indicator from file: bugs and churn
- indicator from file: design flaws

Model relationships
- design concepts
- technical debt indicators

Visually denote indicators (CWE, design flaws, bugs and churn) in some way

Test for correlations between technical debt prone files and files with known vulnerabilities.

Issue
Name
Status
Priority
Label: Security, Impact, Severity
Type: Bug, Bug-Security
CVE

Commit History
Issue
Code
Version history

Code File
Name
LOC
Age

Technical debt

In software-intensive systems, technical debt is a software design issue that:

- Exists in an **executable system artifact**, such as code, build scripts, data model, automated test suites;
- Is traced to **several locations** in the system, implying issues are not isolated but propagate throughout the system artifacts.
- Has a **quantifiable** effect on system attributes of interest to developers (e.g., increasing defects, negative change in maintainability and code quality indicators).

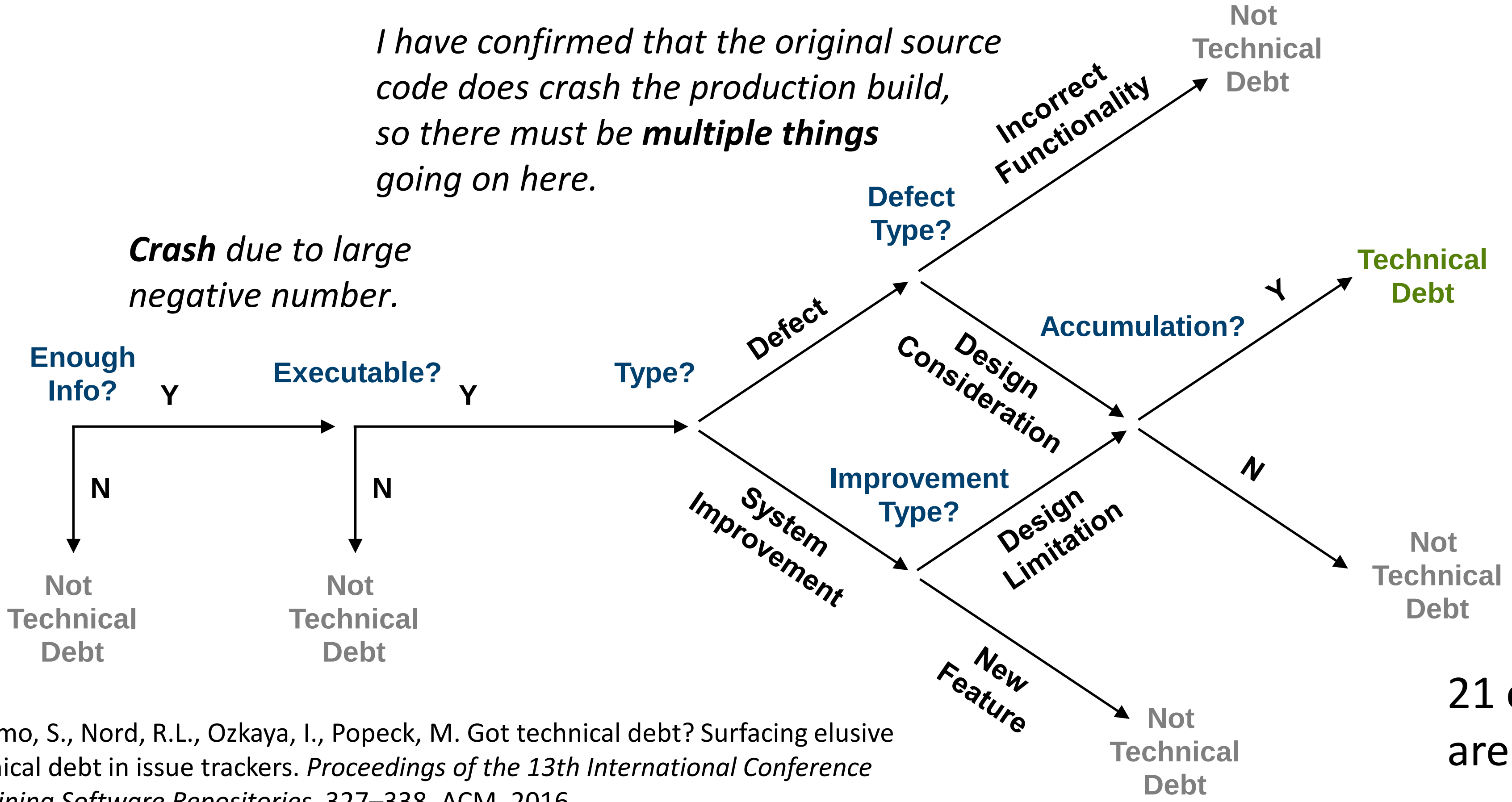
Indicator: Technical Debt Tag

*We could just fend off negative numbers near the crash site or we can **dig deeper***

*Time permitting, I'm inclined to want to know the **root cause**.*

*I have confirmed that the original source code does crash the production build, so there must be **multiple things** going on here.*

***Crash** due to large negative number.*



*There have been **28 reports** from 7 clients… **18 reports** from 6 clients*

*My sense is that if we patch it here, it will **pop-up somewhere else later**.*

*hmm ... **reopening**. the test case crashes a debug build, but not the production build.*

21 of 79 issues labeled security are classified as technical debt.

Bellomo, S., Nord, R.L., Ozkaya, I., Popeck, M. Got technical debt? Surfacing elusive technical debt in issue trackers. *Proceedings of the 13th International Conference on Mining Software Repositories*, 327–338. ACM, 2016.

Finding: Developers Use Technical Debt Concepts

Developers addressing security issues are using technical debt related concepts (italicized):

- getting to the *root cause*
- understanding the *underlying design* issues
- recording symptoms where changes are taking *longer than usual* or problems are *reoccurring*
- predicting consequences in the *longer term*
- building evidence for a more *substantial fix*

Indicator: Design flaws

Unstable Interface

Modularity Violation

Improper Inheritance

Cycle

Xiao, L., Cai, Y., Kazman, R. Design rule spaces: A new form of architecture insight. *Proceedings of the 36rd International Conference on Software Engineering*, 967–977. ACM, 2014.

Unstable Interface

		1	2	3	4	5	6	7	8	9	10	11
1	ui.gfx.size.cc	(1)	Use,3	,2	**,3**	,3	,1		,1	,2		
2	ui.gfx.size.h	Call,3	(2)	,5	**,4**	,2		,1	,2	,1	,1	
3	ui.gfx.point.h	,2	,5	(3)	**,5**	,3		,1	,1	,2	,1	,1
4	ui.gfx.rect.h	**Call,3**	**Call,4**	**Call,5**	**(4)**	Call,6	,2	,2	,2	,5	,2	,2
5	ui.gfx.rect.cc	Call,3	Call,2	Call,3	**Call,6**	(5)	,1	,1	,1	,3	,1	,2
6	webkit.plugins.ppapi.ppapi_plugin_instance.cc	Call,1	Call,	Call,	**Call,2**	Call,1	(6)	,1	,5	,2	,2	,2
7	content.renderer.paint_aggregator.cc		Call,1	Call,1	**Call,2**	Call,1	,1	(7)	,2	,2	,2	,1
8	content.renderer.render_widget.cc	Call,1	Call,2	Call,1	**Call,2**	Call,1	Call,5	Call,2	(8)	,3	,1	,1
9	ui.gfx.rect_unittest.cc	,2	Call,1	,2	**Call,5**	Call,3	,2	,2	,3	(9)	,2	,2
10	webkit.plugins.webview_plugin.cc		,1	,1	**Call,2**	,1	,2	,2	,1	,2	(10)	,1
11	ui.gfx.blit.cc		Call,	Call,1	**Call,2**	Call,2	,2	,1	,1	,2	,1	(11)

Modularity Violation

Shared secret between files

Should be extracted as design rules

		1	2
1	ContextConfig.java	(1)	**,31**
2	TldConfig.java	**,31**	(2)

Analysis: Design Flaws - 1

Increased rates of design flaws are strongly correlated with increased rates of security bugs.

Project	Bug/Design Flaw Correlation	Change/Design Flaw Correlation	Sec Bug/Design Flaw Correlation
Chrome	0.987	0.988	0.979

Design flaws extracted using dependency analysis at the class level within files: unstable interface, modularity violation, improper inheritance, cycles.

Feng, Q., Kazman, R., Cai, Y., Mo, R., Xiao, L. Towards an architecture-centric approach to security analysis. *Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture*. IEEE, 2016.

Analysis: Design Flaws - 2

Moreover, being involved in more types of design flaws correlates with the presence of vulnerabilities.

# Types of Design Flaws	Non-vuln files	Vuln files	% have vulns.
0	8544	47	0.5%
1	7357	141	*2%*
2	2345	91	*4%*
3	194	10	*5%*
4	1	0	*0%*

Finding: Evidence of Correlation

Finding: We see evidence of correlations between vulnerabilities and technical debt indicators such as design flaws and code churn: the more types of design flaws a file is involved in, the higher the likelihood of it also having vulnerabilities; files with vulnerabilities tend to have more code churn.

Qualitative and Quantitative Analysis

Classifying TD from Issues labeled Security

Detecting Design Flaws in Code

	Not TD	TD
Design Flaws	50	15
No Design Flaws	8	6

79 issues are labeled security

- 21 are classified as technical debt
- 65 trace to files containing design flaws

Design Flaws and Future Consequences

Classifying TD from Issues labeled Security

Detecting Design Flaws in Code	Not TD	TD
Design Flaws	50	**15**
No Design Flaws	8	6

66931: *"Is it a workaround … the root bug ...long term fix“*
Flaw: modularity violation

68766: *I've got my bandaid fix all reviewed and ready to check in once the tree reopens. But this problem sounds nasty enough that we definitely need a real fix.”*
Flaws: modularity violation, cycle, improper inheritance

Partial Evidence

Classifying TD from Issues labeled Security

Detecting Design Flaws in Code

	Not TD	TD
Design Flaws	**50** Defect: 26 Feature: 1 Design Problem: 23	15
No Design Flaws	8	6

67577: *"This is a 2-liner. I'll take it, if only to get our rampant security bug list down by one."*
Flaw: modularity violation

64108: *"feature was never fully implemented, we may not have put in proper checks to prevent this."*
Flaws: modularity violation, cycle

Supplement Static Analysis with Developer Knowledge

Classifying TD from Issues labeled Security

Detecting Design Flaws in Code	Not TD	TD
Design Flaws	50	15
No Design Flaws	8	6

10977: “we could just fend off … or we can dig deeper” “if we patch it here, it will pop-up somewhere else later”

70589: “My plan is to back out the brokenness, and fix it properly later”

Multiple Sources of Information

Our preliminary findings demonstrate that analysis necessitates using code, issue trackers, and commit history in concert.

Threats to Validity

Data quality and size

Manual inspection

Identification of technical debt and vulnerabilities

Summary / Future Work

Technical debt matters.

- **Finding 1:** When they address security issues, software developers use technical debt concepts to discuss design limitations and their consequences on future work.
- **Finding 2:** Correlations between vulnerabilities and technical debt indicators warrant further research.

Future work

- Experiment with modifiability and security taxonomies
- Apply to additional data sets
- Investigate causal links
- Codify as design flaws that tools can analyze

Contact Information

Presenter

Robert L. Nord

Senior Member of Technical Staff

Telephone: +1 412.268.5800

Email: info@sei.cmu.edu

http://www.sei.cmu.edu/architecture/research/arch_tech_debt/