Challenges with Updating Software

EFCOG

Pasco, WA. 10/17/18

Gregory M. Pope CSQE

Lawrence Livermore National Laboratory

LLNL-PRES-756669

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Why Update Software?

- Fix known potential security vulnerabilities
- Add new functionality (Major upgrade)
- Add new data formats (Minor upgrade)
- Add new interfaces (Minor upgrade)
- Bug Fixes

Bugs Fixes

- Show Stoppers (major feature busted, no workaround)
- Major (feature does not work but there is a work around)
- Minor (seldom used feature does not work)
- Trivial (word misspelled, button alignment)

Security Vulnerabilities

Known and being exploited	Known and yet to be exploited
Unknown and being exploited	Unknown and yet to be exploited

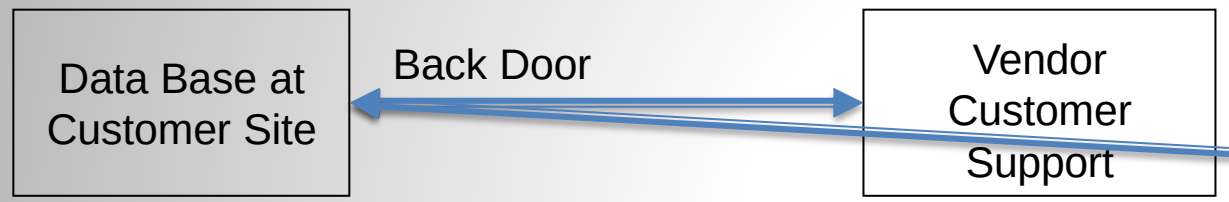
Is it a Bug or is it a Security Vulnerability?

- If the customer is hacked and reports it, it is a security vulnerability.
- If the vendor finds it first and fixes, it is a bug fix.[1]

1. John Allen LLNL OISSO

Is it a Security Vulnerability or a Feature?

Code can be deliberately added to support maintenance



Code can be deliberately added for testing purposes.

Real World Example, Left In Test Code

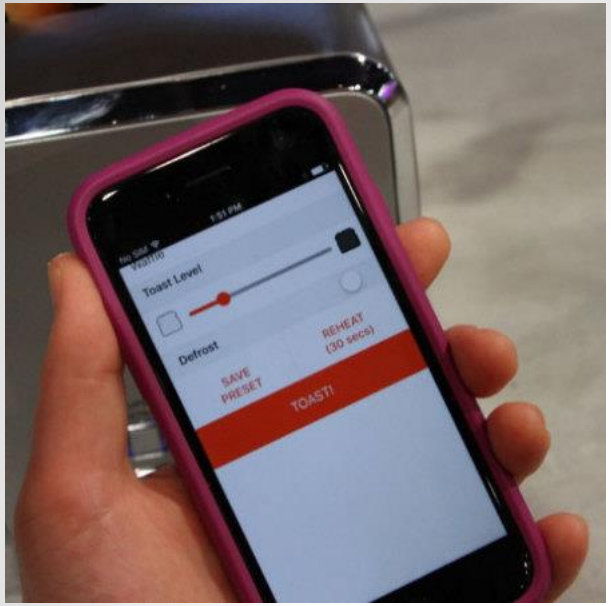
HP laptops found to have hidden keylogger

11 December 2017 Share

Hundreds of HP laptop models were affected

Hidden software that can record every letter typed on a computer keyboard has been discovered pre-installed on hundreds of HP laptop models.

Internet of Things

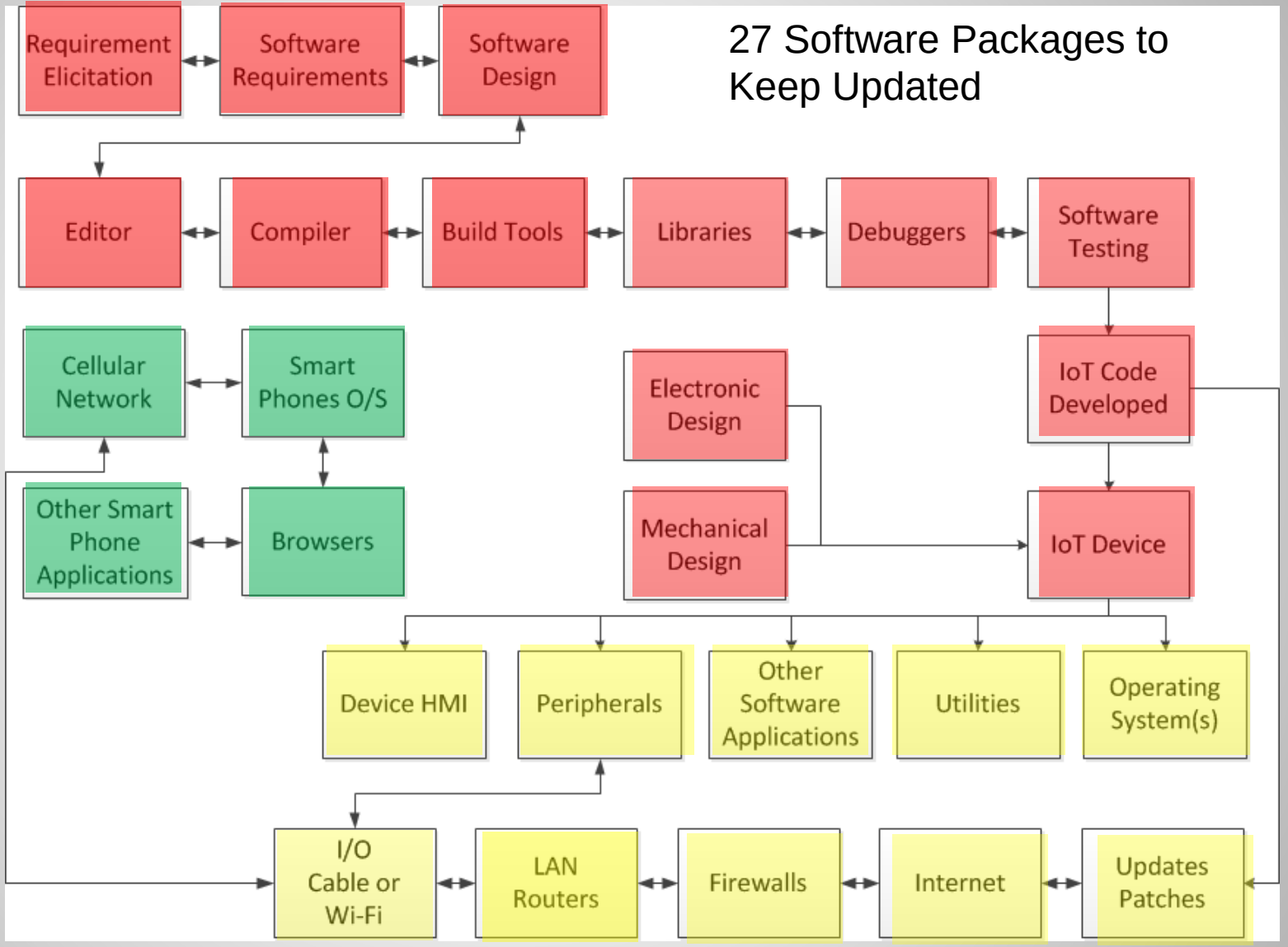


IoT Development Model

Development Subsystem

Mobile Phone Subsystem

External Communications Subsystem



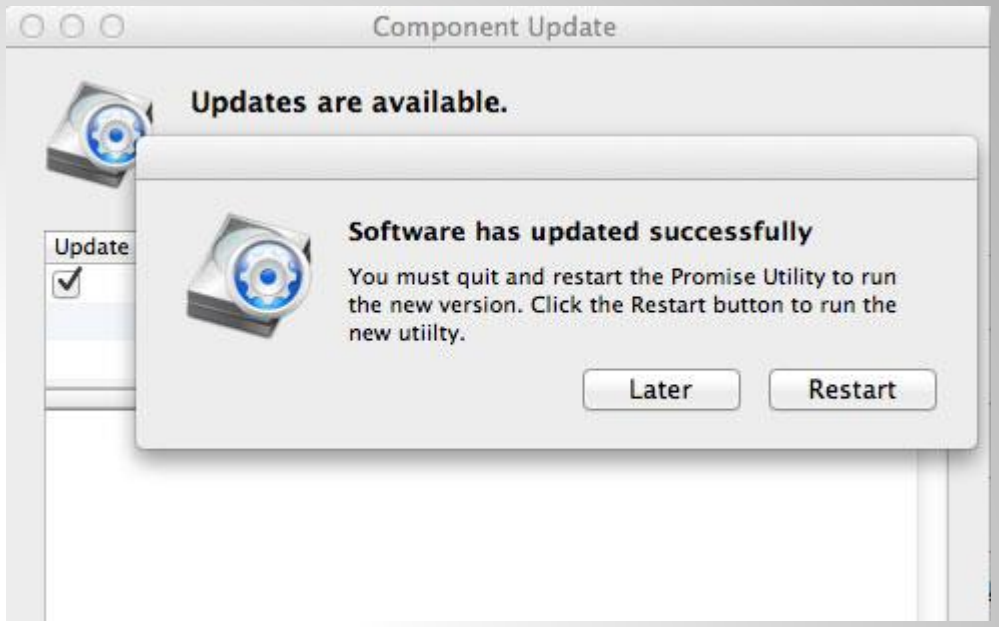
Best Update Security Strategy

- Stay on the most recent software update.
- Latest update probably has the fewest known security vulnerabilities.
- Latest update might also have new bugs, so …

Software Update Risk Mitigation

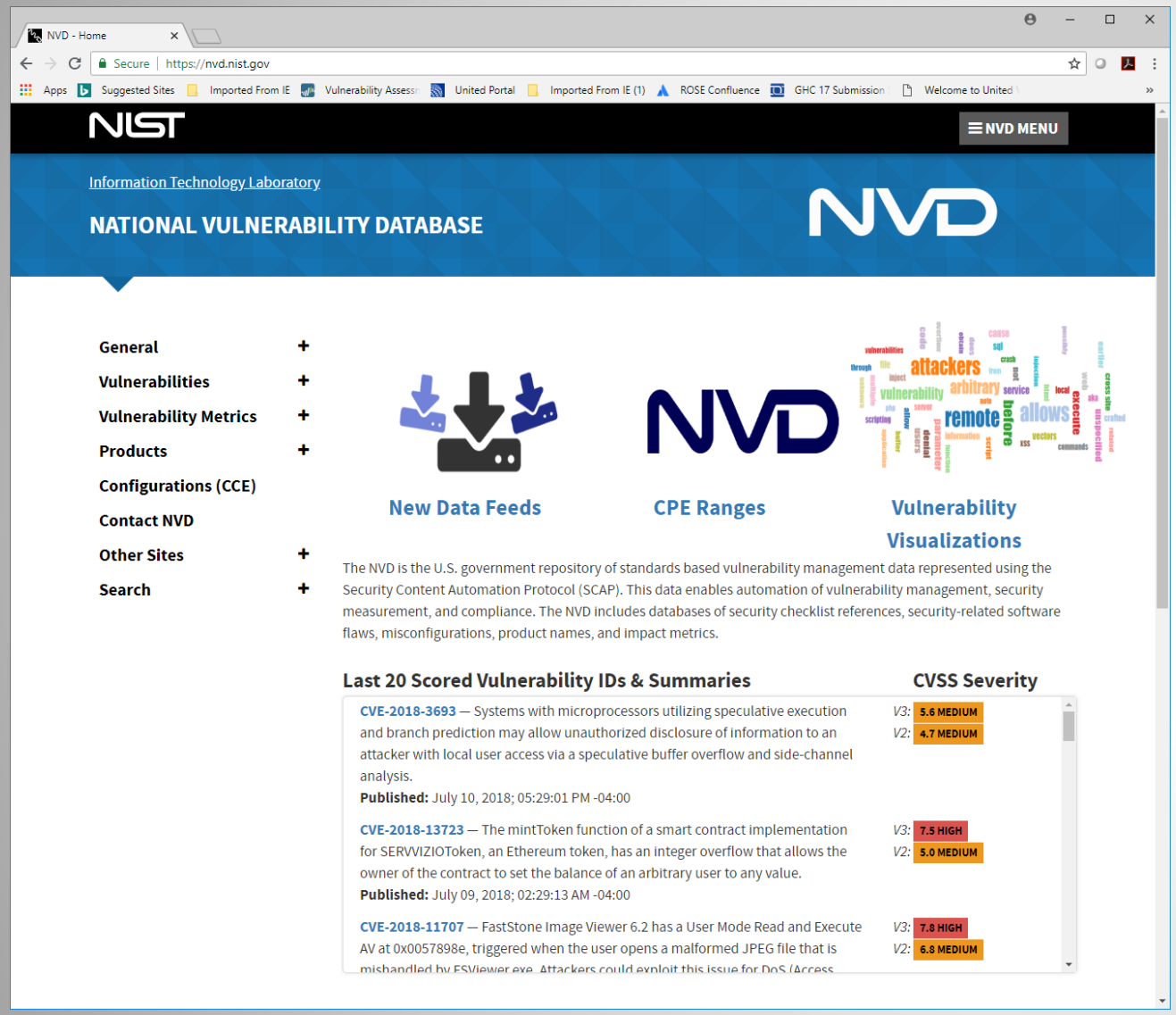
- Updates should require new qualification.
- Qualification should not be onerous
- Because if it is:
  1. may get skipped or
  2. may stay on the older version (with known security vulnerabilities).

Example Update Assurance Process



1. Assure update media or download from a valid source
2. Assure update files scanned for known vulnerabilities
3. Assure current version is backed up or a test system is used to assure restore
4. Assure successful update message appears, screen shot for record
5. Assure updated software passes in house test cases that represent features used
6. Assure test report, screen shot, appropriate records updated.

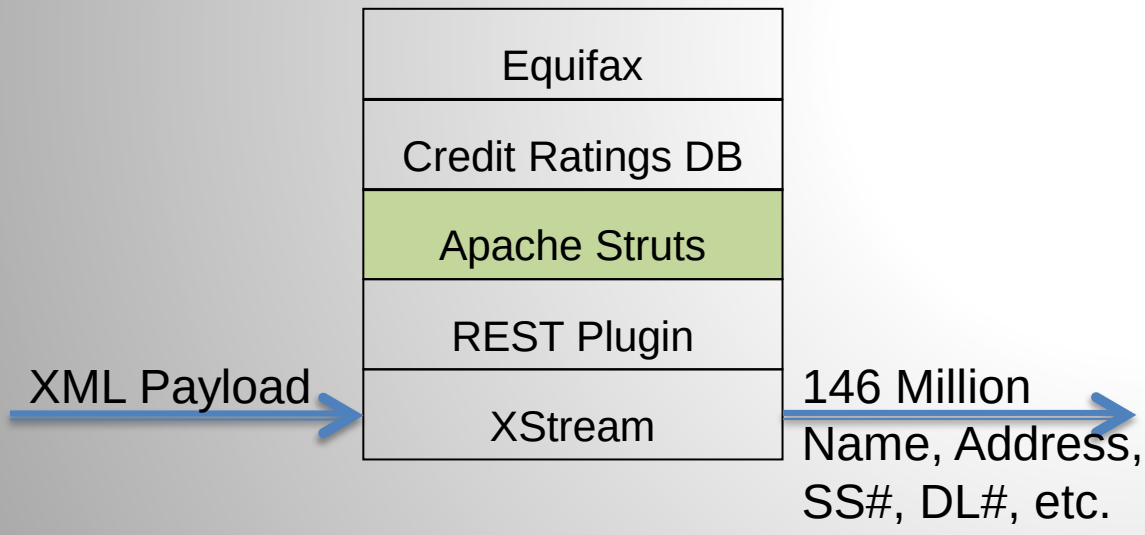
NIST - National Vulnerability Data Base



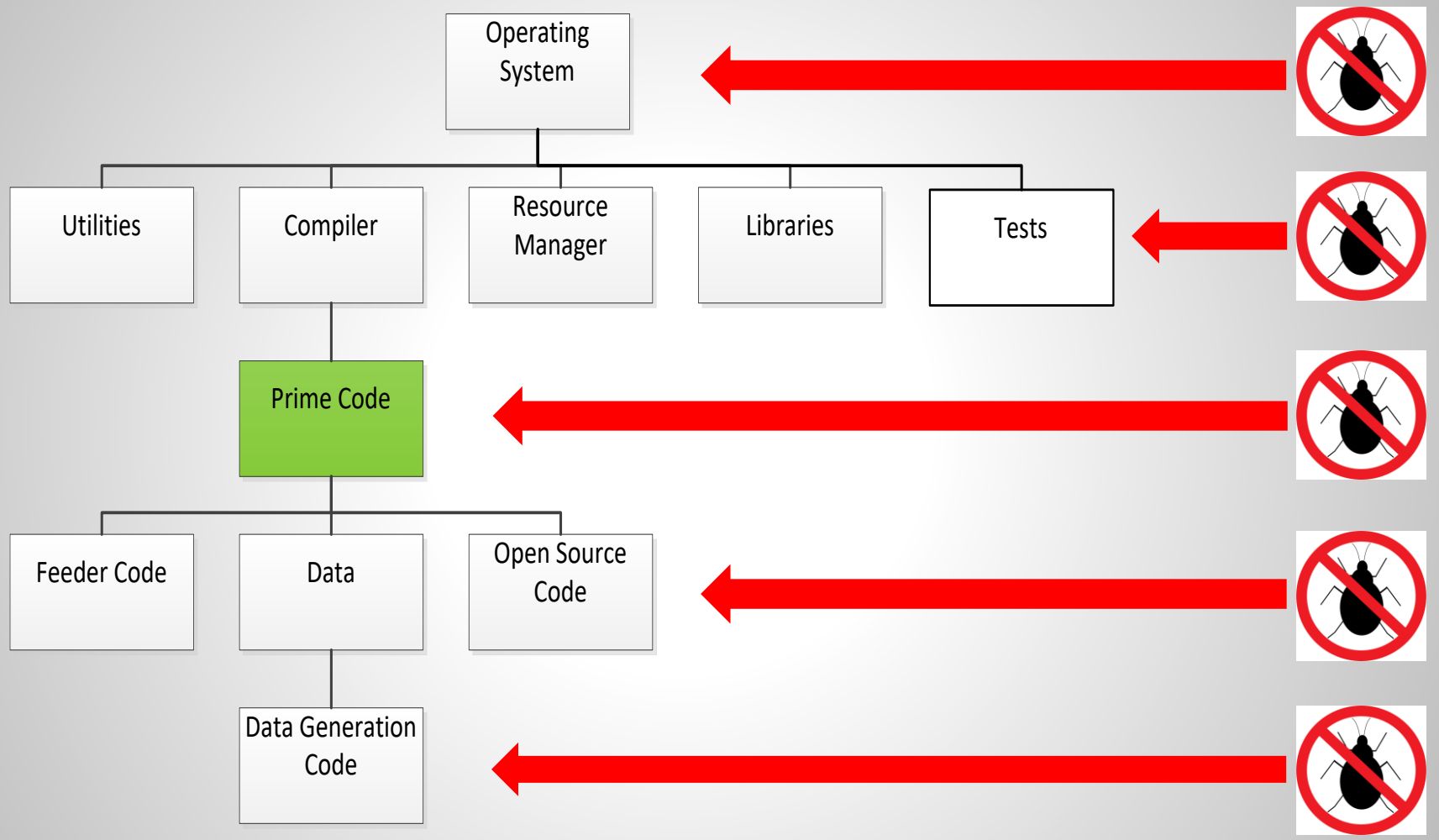
https://nvd.nist.gov/

NIST NVD Challenge – Update Fast

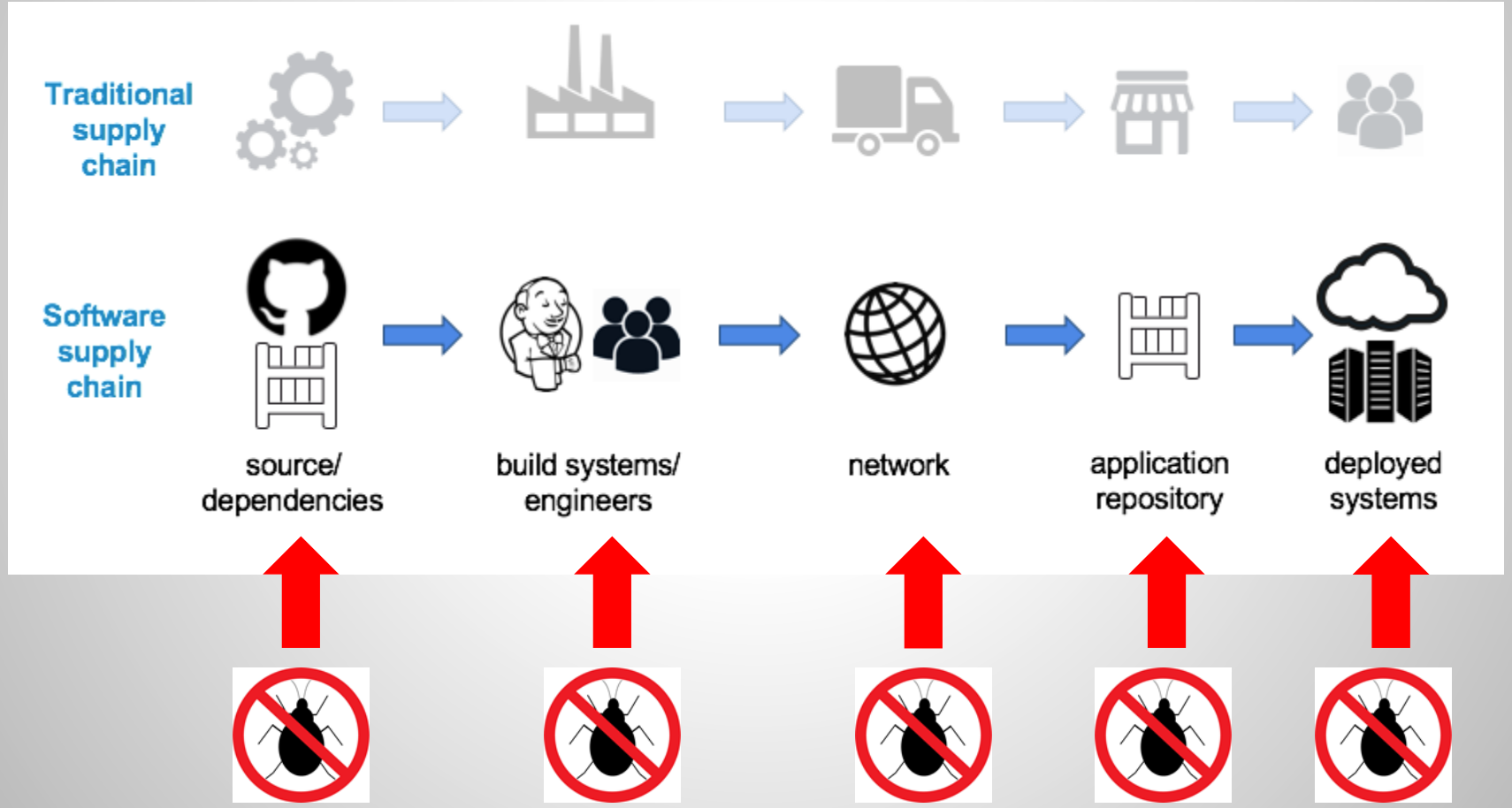
- Is the security vulnerability in any of the software I am using?
- Do you use Apache Struts?
- It may be in your “stack”.



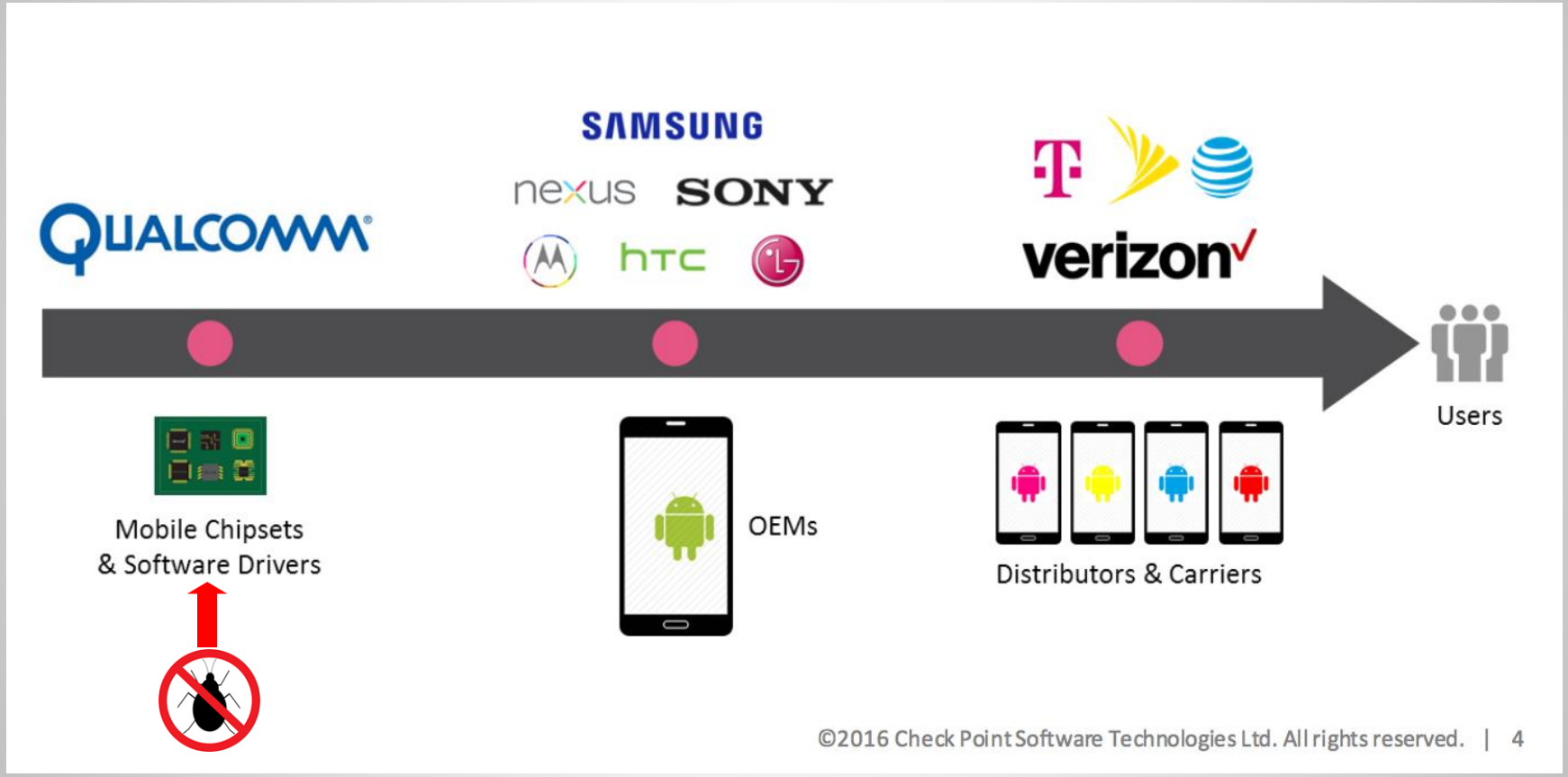
Systems View of Software



Software Supply Chain Vulnerability



Supply Chain Example – QuadRooter
900 Million Android Phones Impacted

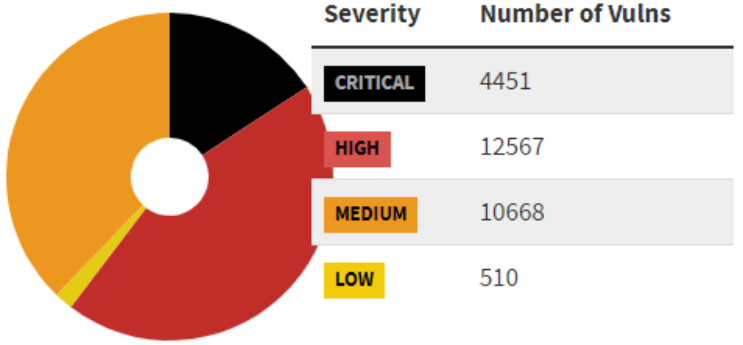


Todays NVD List (7/13/2018), Over 110,000 Served

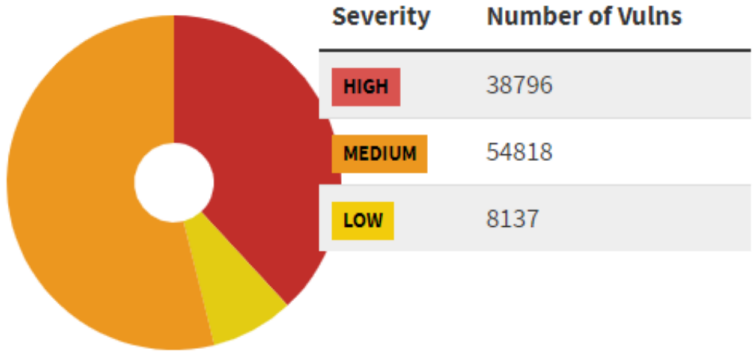
CVEs Received and Processed

Time Period	New CVEs Received by NVD	New CVEs Analyzed by NVD	Modified CVEs Received by NVD	Modified CVEs Re-analyzed by NVD
Today	0	58	0	0
This Week	698	214	236	2
This Month	1153	410	412	2
Last Month	1902	1287	1307	11
This Year	9697	7607	8269	97

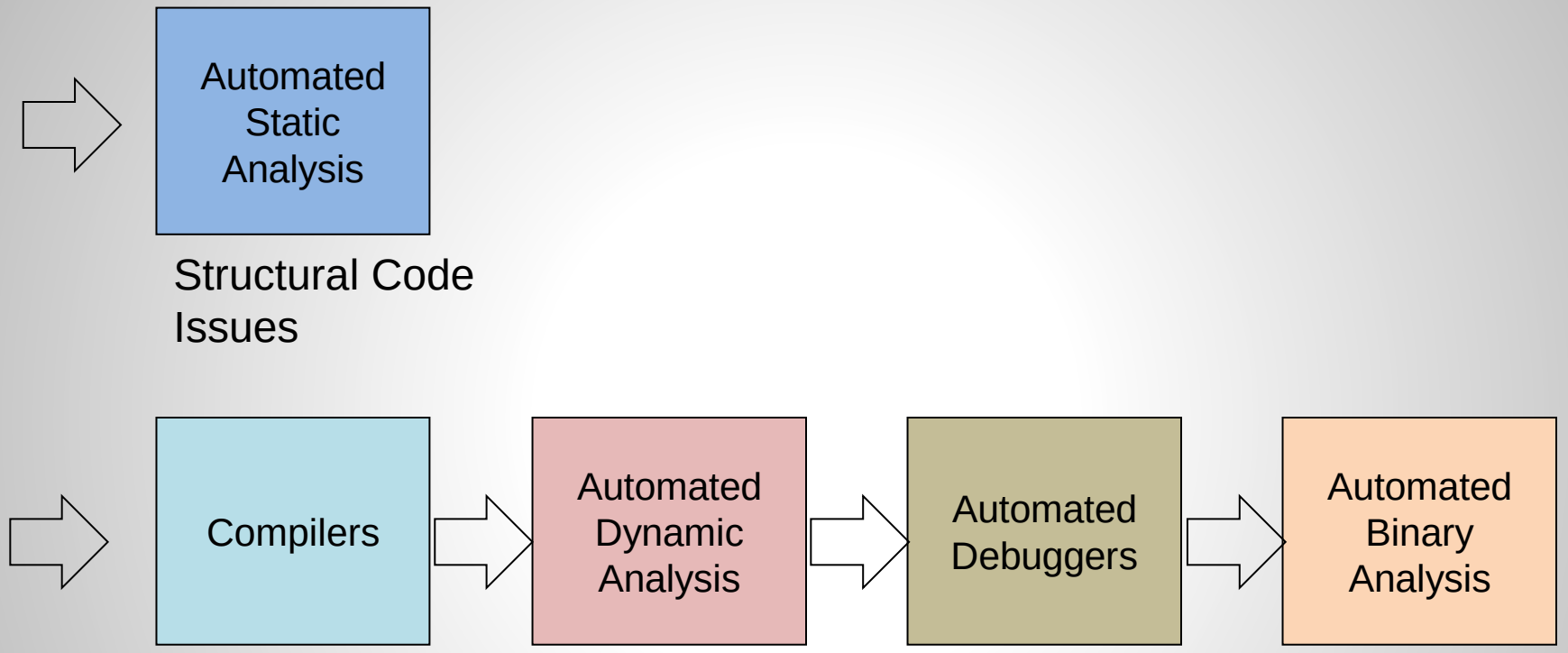
CVSS V3 Score Distribution



CVSS V2 Score Distribution



Tools Can Find Potential Vulnerabilities in Code



COTS Static Analysis and Cyber Exposure Tools

CODESONAR

COVERITY BY SYNOPSYS

PARASOFT

MICRO FOCUS Fortify

RogueWave SOFTWARE | Klocwork

tenable

ROSE

Static Analysis By language
https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis#C,_C++

40 More
http://www.softwaretestinghelp.com/tools/top-40-static-code-analysis-tools/

These Tools Analyze Software Code Structure (Not Requirements)

- Static Analyzers
- Dynamic Analyzers
- Binary Analyzers
- Debuggers
- Compilers
- Cyber Exposure Tools

These Tools Could Find 50% of Faults Pre-Test

Requirements	8.1%
Features and Functionality	16.2%
Structural Bugs	25.2%
Data	22.4%
Implementation and Coding	9.9%
Integration	9.0%
System Software Architecture	1.7%
Test Definition and Execution	2.8%
Other	4.7%

25.2%

15%

7%

4.5%

51.7%

Sample size 6,877,000 statements (comments included)
Total defects 16,209, Bugs per 1000 statements 2.3
Software Testing Techniques, Boris Beizer, Second Edition, Van Nostrand Reinhold, page 57, Table 2.1
Using Automated Static Analyzers to Debug Your Code, Pope, Ferrari, Oliver Better Software Magazine July/August 2008, page 36

There is an Almost Infinite Set of Software Possible Faults

- Since we can not easily identify all faults:
- Identify the most common C++, C faults*:
  - Research LLNL scientific C,C++ codes
  - Research Industry Codes C, C++ codes
- Identify faults that can be security vulnerabilities

* According to a 2016 survey by IEEE Spectrum, C and C++ took the top two spots for being the most popular and used programming languages in embedded systems.

The Pareto Principle Shows Up In Fault Detection In Software (IEEE)

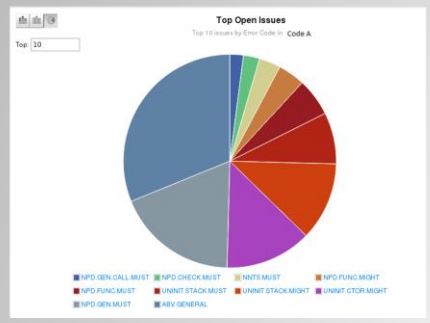
TABLE 13
Pareto Effect in ASA Faults

	% all faults	% critical faults
Top 1 fault: Possible use of NULL Pointer	45.92	60.86
Top 5 faults: Top 1 fault + Possible Access Out-of-Bounds (Custodial) pointer not freed or returned Memory leak Variable not initialized before using	77.26	85.11
Top 10 types: Top 5 faults + Inappropriate deallocation Suspicious use of ; Data overrun Type mismatch with switch expression Control flows into case/default	89.87	90.42

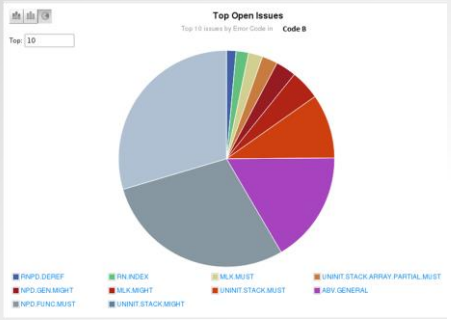
On the Value of Static Analysis for Fault Detection in Software Jiang Zheng, Student Member, IEEE, Laurie Williams, Member, IEEE, Nachiappan Nagappan, Member, IEEE, Will Snipes, Member, IEEE, John P. Hudepohl, and Mladen A. Vouk, Fellow, IEEE https://collaboration.csc.ncsu.edu/laurie/Papers/TSE-0197-0705-2.pdf

LLNL Code Research (Klocwork)

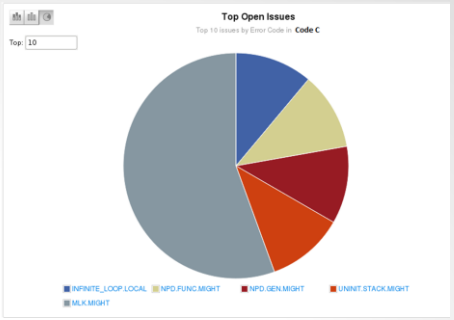
Code A - 937 KSLOC



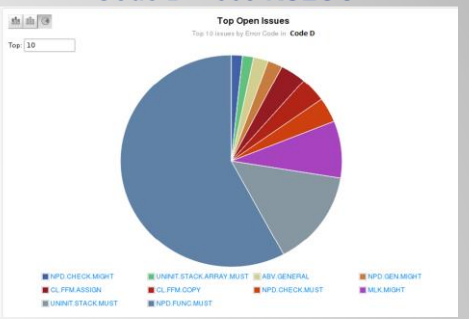
Code B - 696 KSLOC



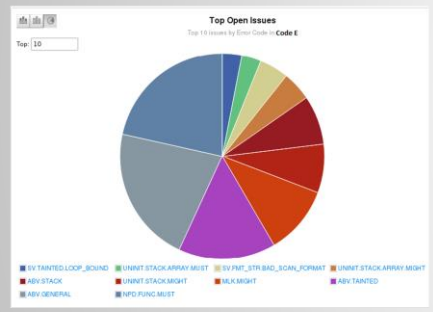
Code C – 35 KSLOC



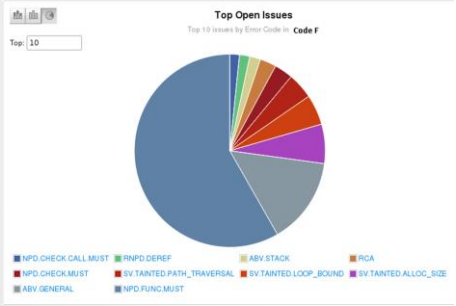
Code D – 509 KSLOC



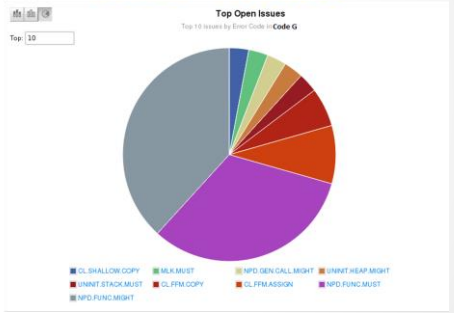
Code E – 25 KSLOC



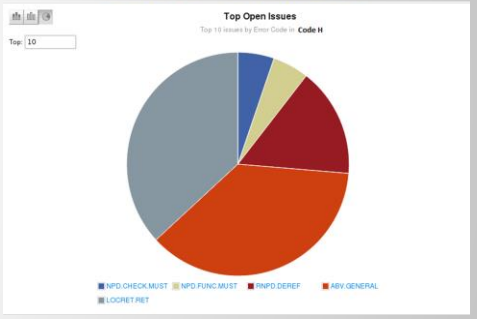
Code F – 1,277 KSLOC



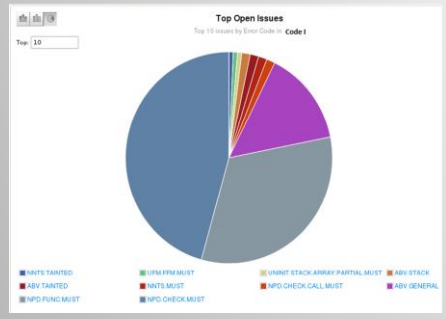
Code G – 143 KSLOC



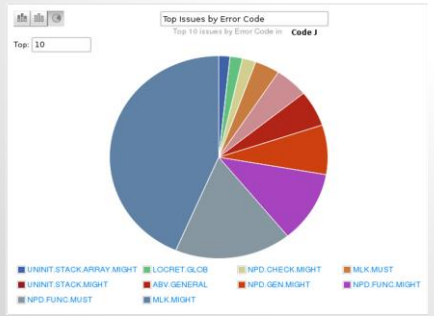
Code H – 191 KSLOC



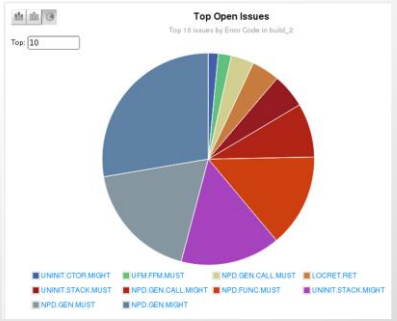
Code I – 124 KSLOC



Code J – 104 KSLOC



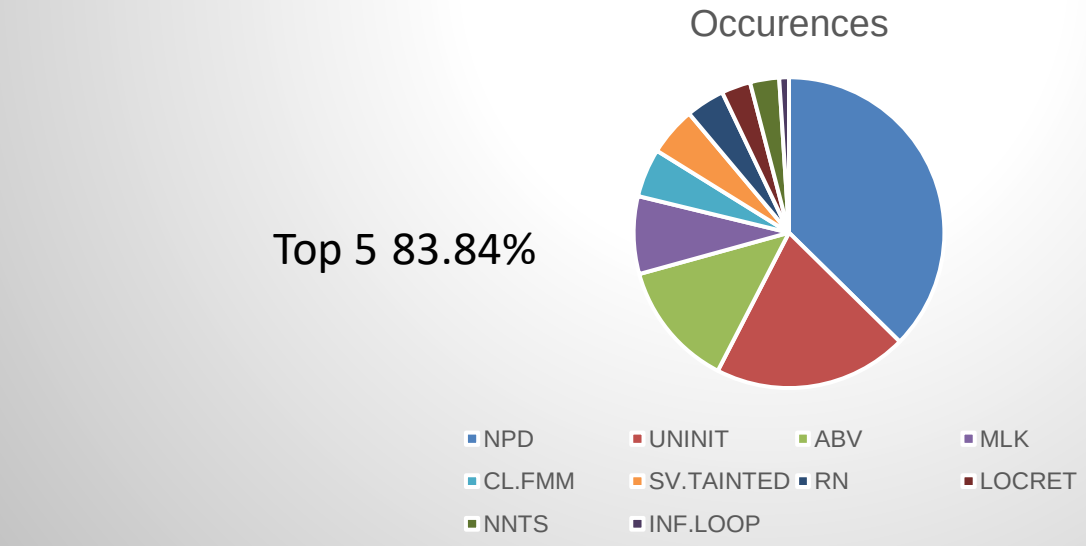
Code K – 1790 KSLOC



Research Results

- IEEE (NCSU) Study ~ 33 million LOC C, C++, since 2001 NORTEL (Network Services Code)
- LLNL Study ~ 6 million LOC C,C++, Scientific Codes since 2006

Top 5 83.84%



Code	KSLOC
A	973
B	696
C	35
D	509
E	25
F	1277
G	143
H	191
I	124
J	104
K	1780
Total	5857

Research Comparisons
Top 5 Agree

IEEE Research Faults
Top 1 fault:
Possible use of NULL Pointer
Top 5 faults:
Top 1 fault +
Possible Access Out-of-Bounds
(Custodial) pointer not freed or returned
Memory leak
Variable not initialized before using
Top 10 types:
Top 5 faults +
Inappropriate deallocation
Suspicious use of ;
Data overrun
Type mismatch with switch expression
Control flows into case/default

LLNL Research Faults

Occurences	Issue
37	Null Pointer Deref
20	Uninitialized Variable
13	Buffer Overflow
8	Memory Leak
5	Freeing Freed Memory
5	Unvalidated Loop Iterator
4	Suspicouis Use before null check
3	Return Local Var
3	Non Null Terminated
1	Infinite Loop

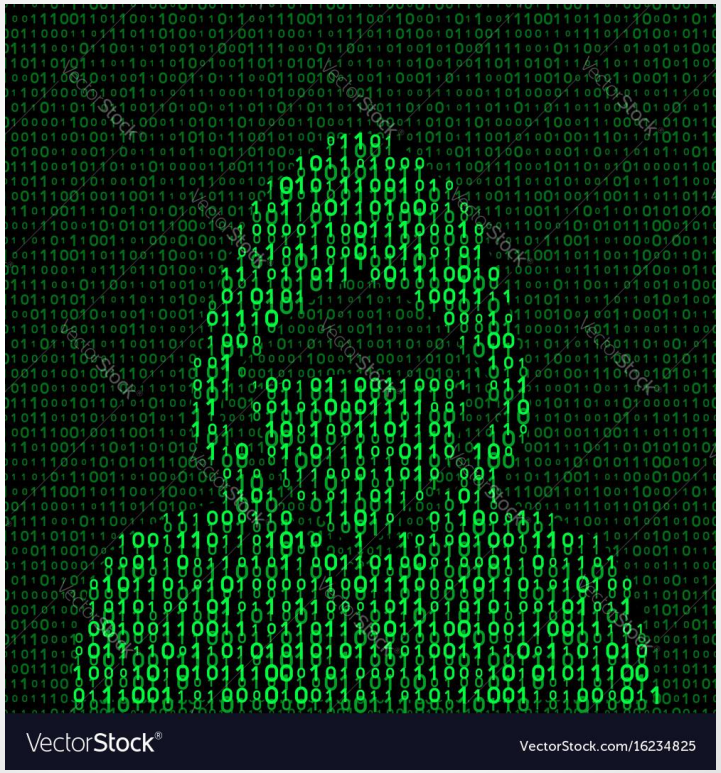
Top 5 Faults in C ++ and C Codes Could Cause Computer Systems To Reboot

1. Null or Stale Pointer Use
2. Memory Out of Bounds
3. Memory Leak
4. Variable Not Initialized Before Use
5. Inappropriate Deallocation

FAA orders Boeing 787 safety fix: Reboot power once in a while – Seattle Times 12/1/2016

May Not Have the Source Code e.g. Binary Analysis Examples

- A priory virus and worm signature detection
- Suspicious file tampering
- Similarity / Differences
- Obstification techniques detection
- Dead code detection



- Back door detection
- Alteration detection
- Feasibility path analysis
- Third party code detection
- Big Five fault detection

The ROSE Compiler Framework (Open Source)

Botox

- Allows tools to be written to do the following:
  - Detect Source and Binary code potential vulnerabilities
  - Rewrite code and eliminate detected vulnerabilities
  - Use Source and Binary files to determine risk levels
  - Translate legacy codes (Ada, Jovial) into contemporary codes (C++)
  - Port and Optimize code for new HPC architectures
  - Developed over 25 years at LLNL http://rosecompiler.org/

Eliminates Wrinkles

Eliminates Migraines

Summary

- Certain software bugs can be potential security vulnerabilities
- There are ways to detect and eliminate these software bugs
- Other types of vulnerabilities may be features that are misused
- The NIST NVD alerts us to known vulnerabilities
- Being on the latest software update is the safest security policy
- Software updates should pass appropriate* assurance process

* Goldilocks Rule: Not too hard and not too easy