RSA®Conference2019

San Francisco | March 4 – 8 | Moscone Center

BETTER.

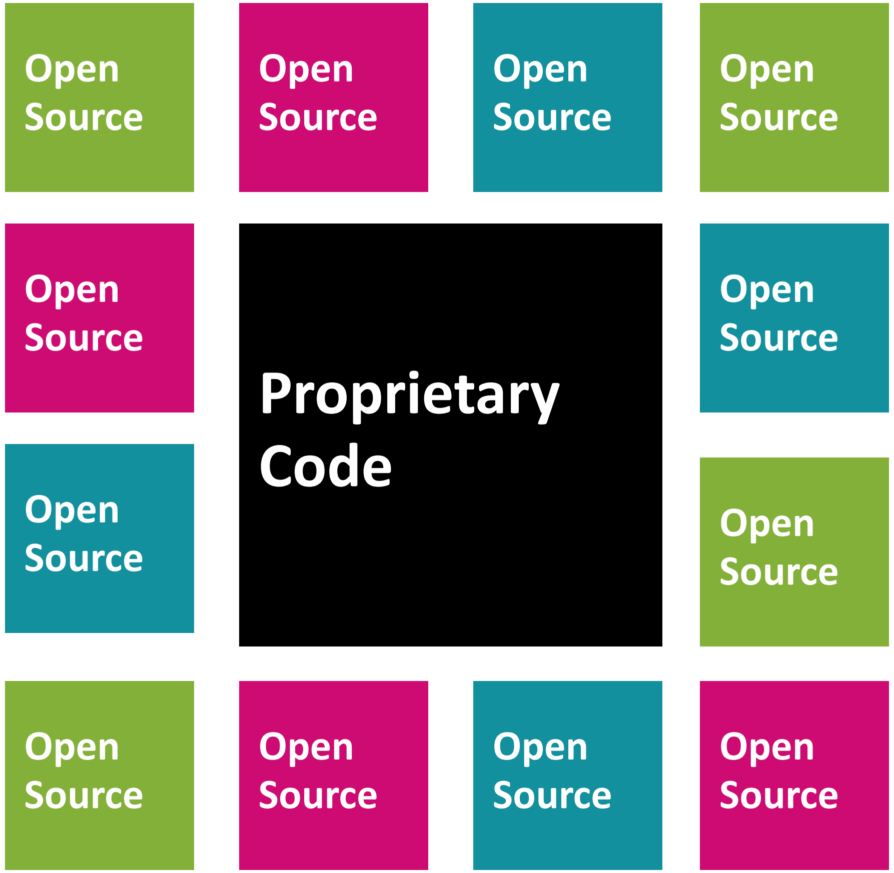
SESSION ID: PDAC-R11

How Understanding Risk Is Changing for Open Source Components

Chris Wysopal

CTO & Co-founder
Veracode
@weldpond

Built Mostly from Components



Open Source

Open Source

Open Source

Open Source

Open Source

Proprietary Code

Open Source

Open Source

Open Source

Open Source

Open Source

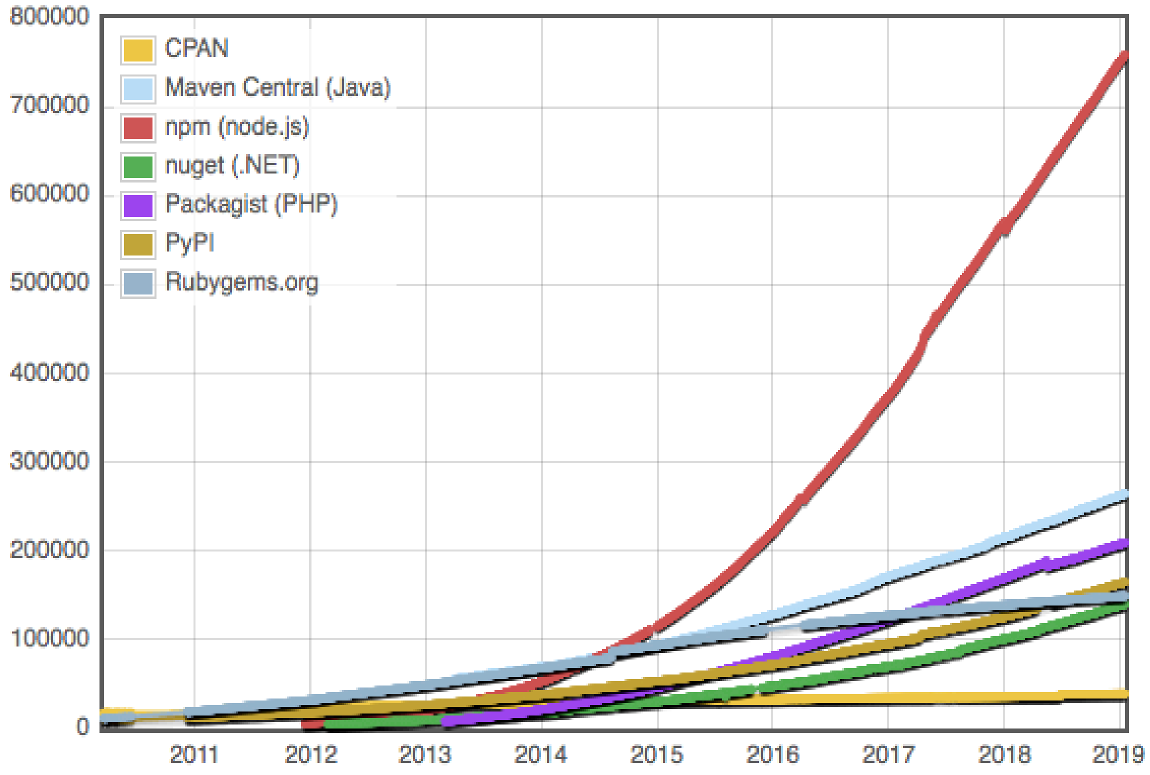
Open Source

Open Source

80% to 95% of **modern apps** consist of assembled components.

Open Source Repo Stats

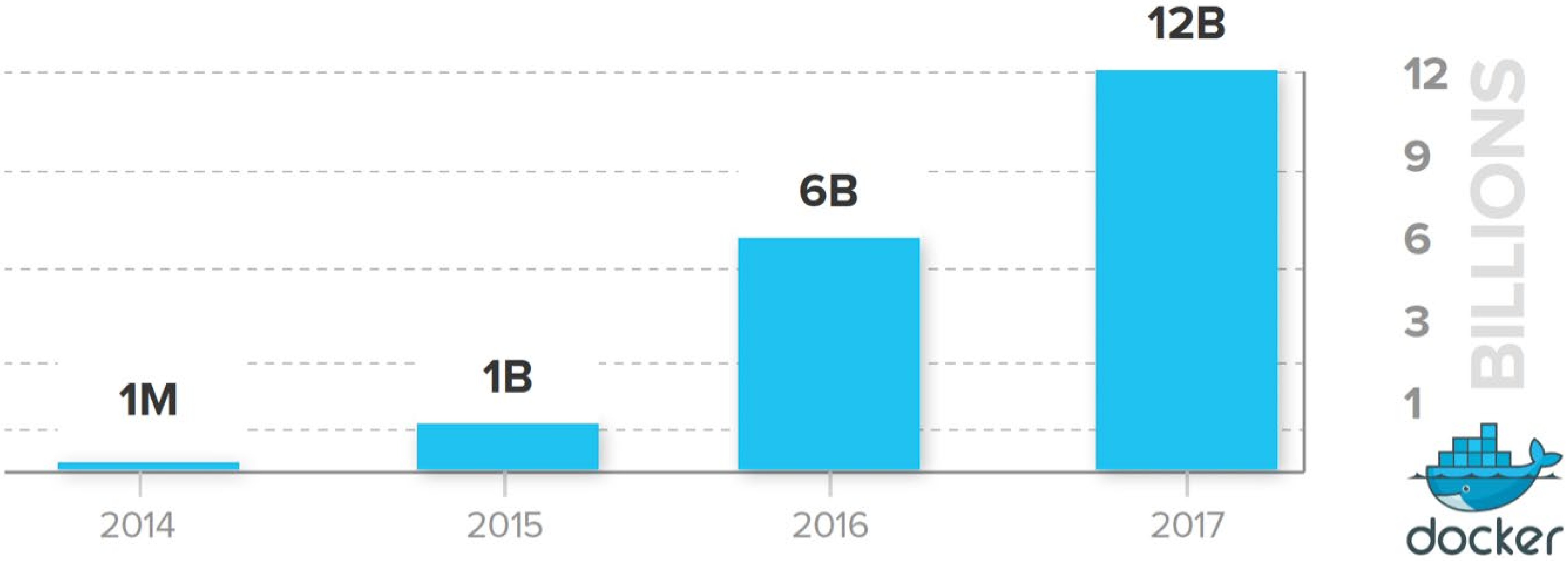
Module Counts



Language	Packages	Avg. Growth
Perl	39,416	2/day
Java	265,303	131/day
Node.js	762,073	507/day
.NET	140,541	67/day
PHP	210,210	134/day
Python	165,737	114/day
Ruby	149,579	27/day

http://www.modulecounts.com/ Data from January, 2019

Pulls from Docker Hub



DockerCon 2017

Open Source – More or Less Secure?

- Defect rate in open source is no better or worse than first party code
- The difference is that developers never revisit
- Integrated and abandoned
- It’s not a problem until a vulnerability is discovered

Sizing The Problem

96% of applications contain open source components

Source: Black Duck

46 Applications have an average of 46 components.

Source: Veracode

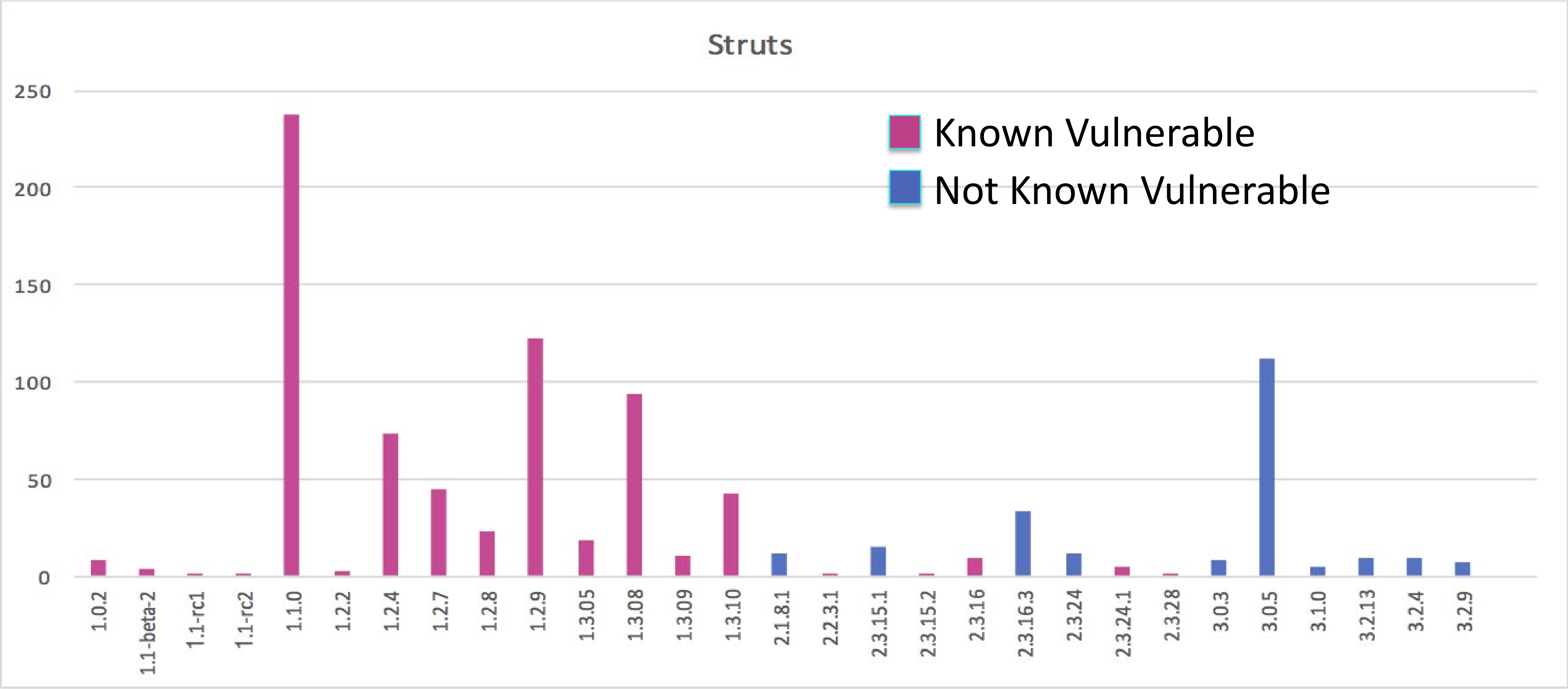
67% of applications had vulnerabilities in those components.

Source: Black Duck

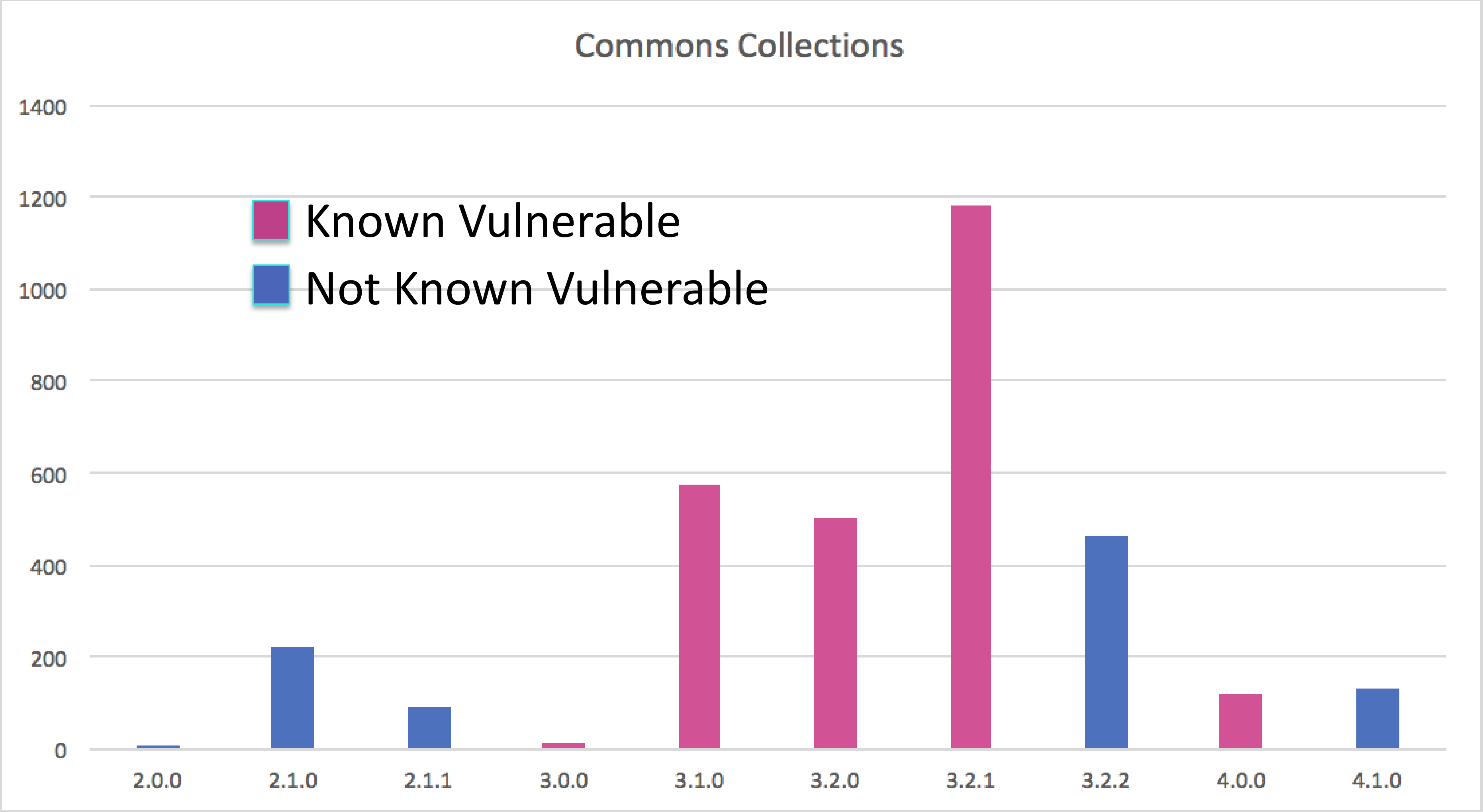
4 On average, vulnerabilities identified have been publicly known for 4 years

Source: Black Duck

Integrated and Abandoned Explicitly - Struts



Integrated and Abandoned Implicitly – Apache Commons Collections



Where do you find open source vulnerability data?

Not all public vulnerabilities are in the National Vulnerability Database (NVD)

- Public information about open source vulnerabilities is available directly from open source projects
- Security bulletins, release notes, commit comments, and source code comments contain vulnerability information
- This information is readily available to attackers and defenders
- Automated services can crawl this information daily. Security analysts can performs quality review, and compile augmented DB.

Percentage of vulnerabilities not in the NVD – 31%

Language	CVE	Reserved CVE	SVE	% SVE Low	% SVE High
JS	604	47	490	42.94%	44.79%
PHP	522	14	128	19.28%	19.69%
DOTNET	58	0	1	1.69%	1.69%
JAVA	749	60	335	29.28%	30.90%
RUBY	284	43	268	45.04%	48.55%
PYTHON	389	59	228	33.73%	36.95%
GO	90	5	218	69.65%	70.78%
CPP	193	8	12	5.63%	5.85%
OBJECTIVEC	631	14	9	1.38%	1.41%
CSHARP	33	3	0	0.00%	0.00%
	3553	253	1689	**30.74%**	**32.22%**

% SVE Low assumes reserved CVEs overlap with SVEs

% SVE High assumes reserved CVEs do not overlap with SVEs

Do I need to remediate every vulnerable component?

Component Vulnerability Exploitability

- A product is vulnerable when it contains a vulnerable component **and** the product uses the library in such a way that the vulnerable code can be exercised.
- Control flow analysis was used determine if vulnerable code is reachable from the product code.
- Analysis was not performed to determine if vulnerable code can be called directly by attacker or called when attacker has exploited another vulnerability.

For Java, Ruby and Python, less than 5% of products that contain a library with a vulnerability are vulnerable

	repos analyzed	% component vulnerabilities that make the products vulnerable
Ruby	624	**3.79%**
Java	5897	**4.34%**
Python	624	**0.93%**

JavaScript study found 26.7% made products vulnerable

Towards Smoother Library Migrations: A Look at Vulnerable Dependency Migrations at Function Level for npm JavaScript Packages

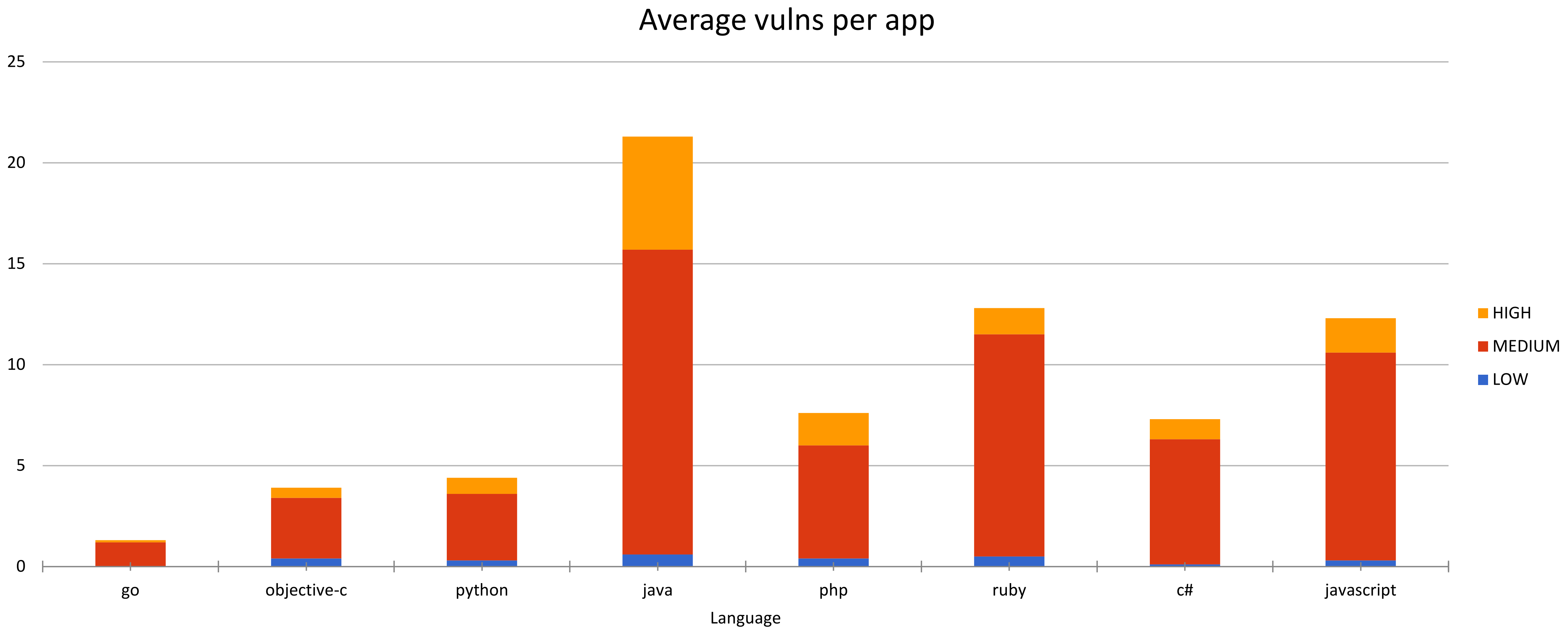
http://se-naist.jp/pman3/pman3.cgi?DOWNLOAD=652

What does a good process look like?

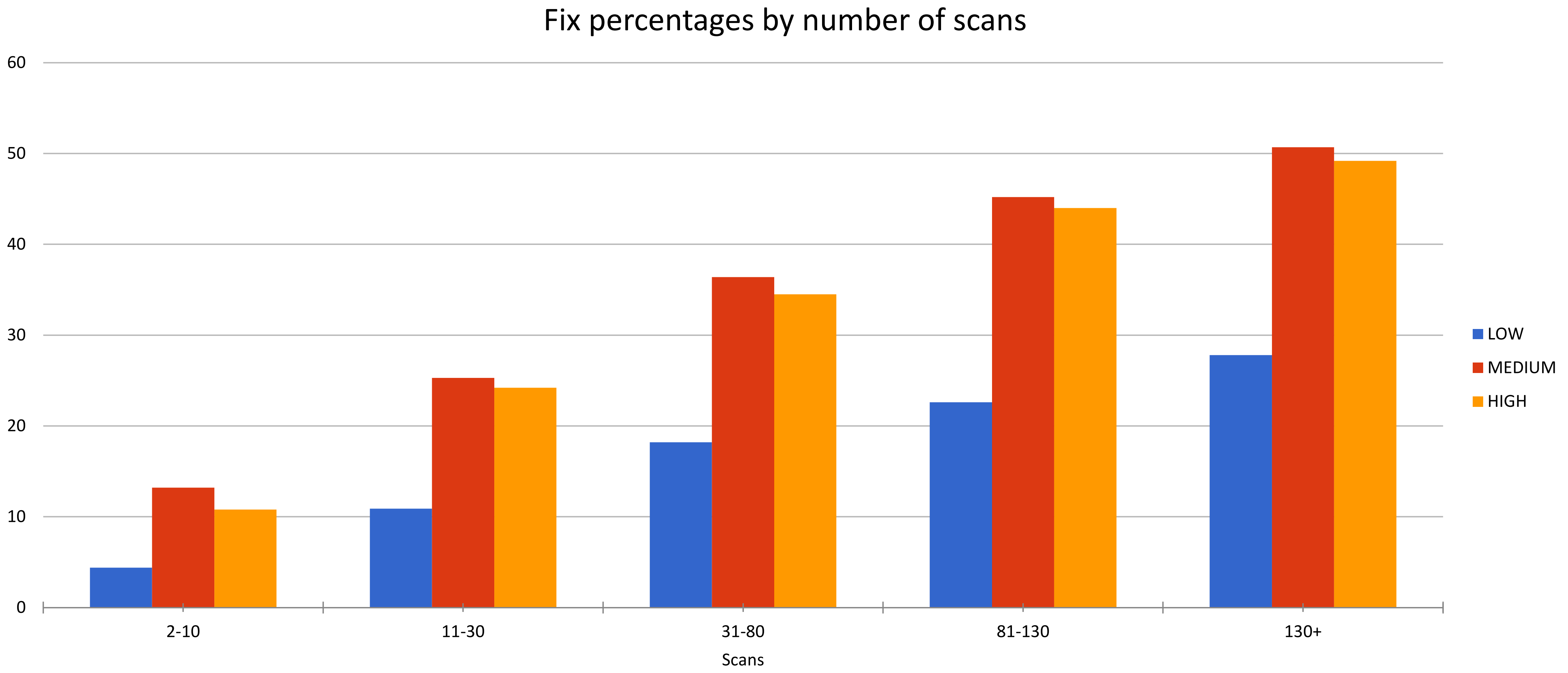
Building a solution into your development lifecycle.

- Integrate into your CI/CD pipeline so you have a record of what goes into production.
- Scanning your repos is OK but pipeline could apt-get update to a vulnerable version.
- Open tickets in your ticketing system for each component that should be updated.
- Create a policy of grace period by severity and vulnerability.

How much risk are organizations finding from open source components



How much risk are organizations remediating?



What if we know the component with a vulnerability is making my app vulnerable right now?

Avg # of scans to fix a vulnerable component: **22**

Avg # of scans to fix a vulnerable component with a vulnerable method: **17**

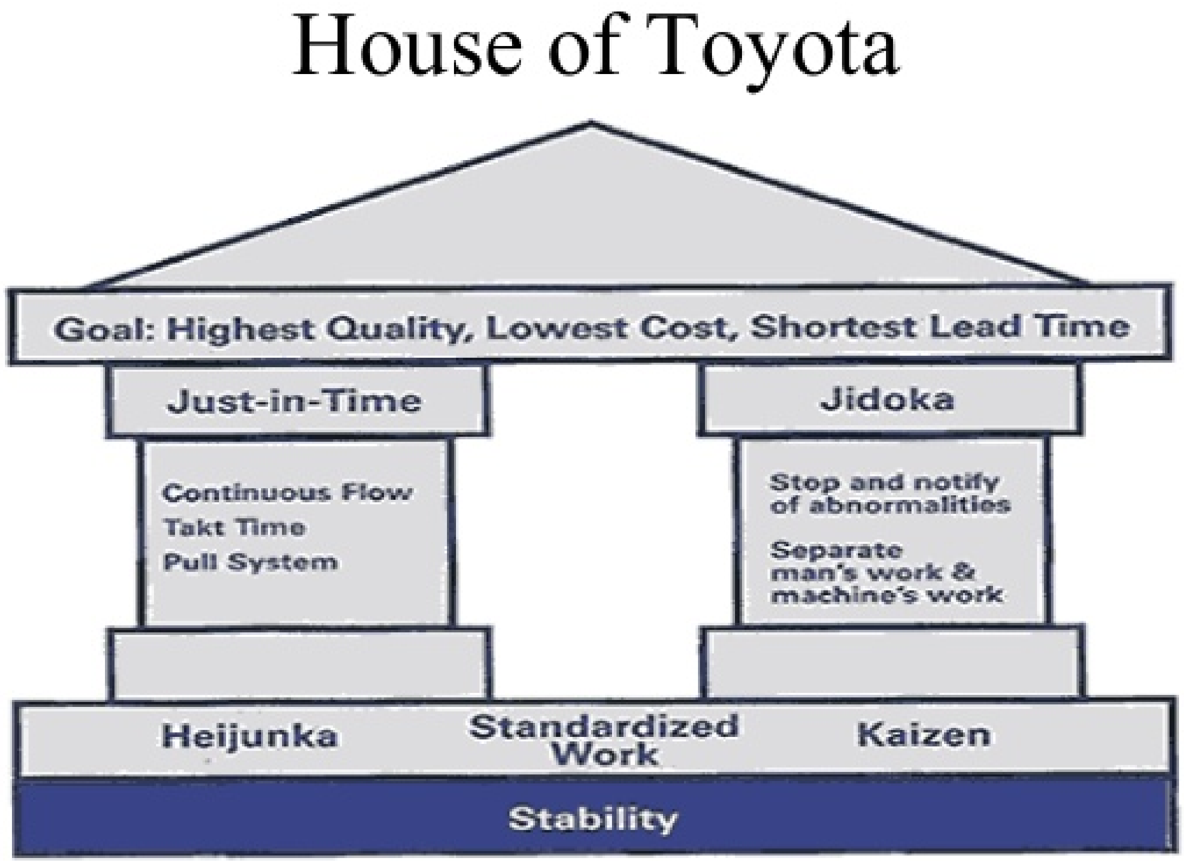
Proportion of vulnerable components fixed: **32%**

Proportion of vulnerable components with vulnerable methods fixed: **41%**

Developers fix more and fix faster when they have vulnerable method information available.

Use fewer, better suppliers

- Toyota's lean manufacturing model uses fewer, better suppliers
- Use containers, libraries and frameworks that are proven to work and vetted by your security team and in your repository
- Keep track of what you have!



House of Toyota
Goal: Highest Quality, Lowest Cost, Shortest Lead Time
Just-in-Time
Continuous Flow
Takt Time
Pull System
Jidoka
Stop and notify of abnormalities
Separate man's work & machine's work
Heijunka
Standardized Work
Kaizen
Stability

Toyota Production System "House"

Apply What You Have Learned Today

- Next week you should:
  - Understand the process for managing open source code within your development organization
- In the first three months following this presentation you should:
  - Create an inventory of open source code
  - Remediate where outdated and vulnerable open source is used in critical applications
- Within six months you should:
  - Integrate a process into your development lifecycle to monitor what open source is going into production

RSA Conference2019

Questions?

Thank You!

Chris Wysopal

@weldpond