Exemplars, Laggards, and Hoarders

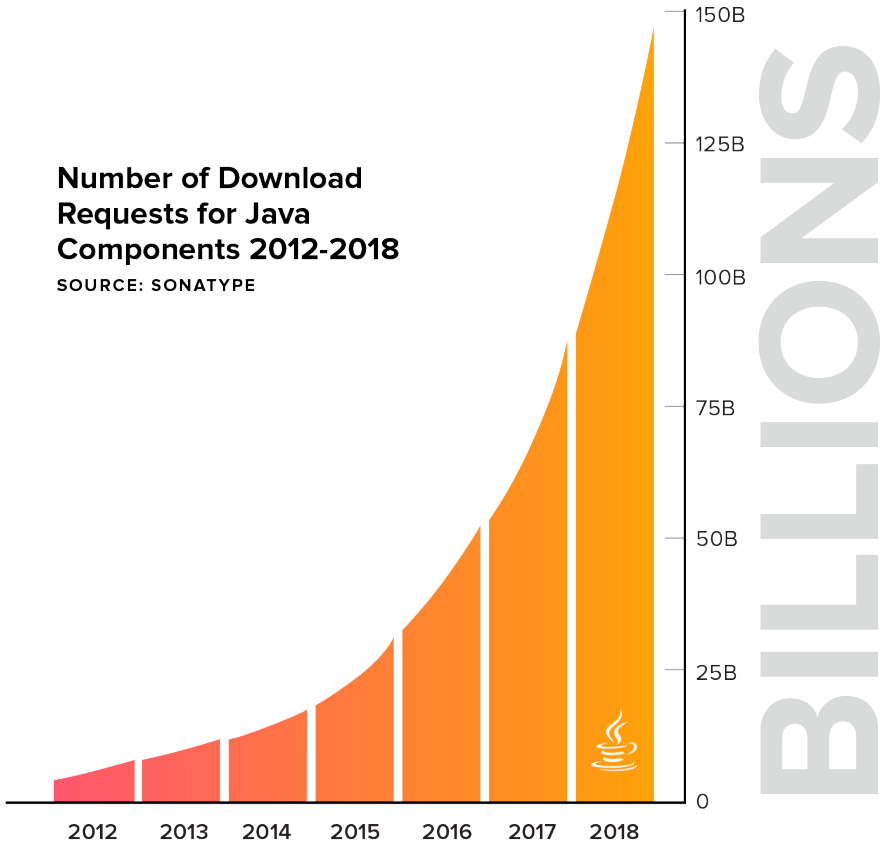
A Data-driven Look at Open Source Software Supply Chains

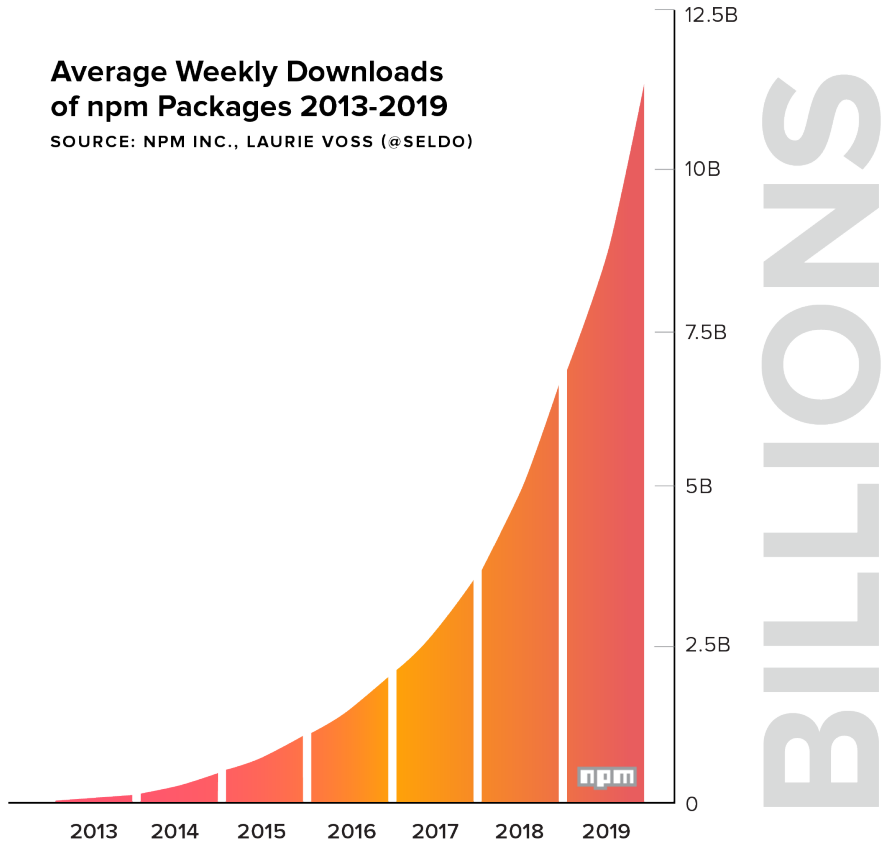
Dr. Stephen Magill
CEO, MuseDev
Principal Scientist, Galois, Inc.
@stephenmagill

Derek E. Weeks
VP and DevOps Advocate, Sonatype
Co-founder, All Day DevOps
@weekstweets

Development and release velocity have been shown to be highly predictive of project outcomes.

OSS download volumes are a proxy for build automation





@stephenmagill

@weekstweets

313,000
java component downloads annually

2,778
Component suppliers

8,200
Component release

2019 State of the Software Supply Chain

The 5th annual report on global open source software development

presented by

sonatype

in partnership with

galois | IT REVOLUTION

Everyone has a software supply chain.

(including open source projects)

36,000 components

3.7 million releases

12,000 organizations

6,200 developers

86,000 applications

Core DevOps Mantra

Faster is Better

Drives improved profitability, market share, quality.*

* Nicole Forsgren PhD, Jez Humble, Gene Kim. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press, March, 2018.

Faster
is Better

Does this hold for open source?

Two Different Worlds

Enterprise	Open Source
Multiple Deploys per Day	Versioned Releases
Consistent Development Team	Fluid Group of Developers
Well-resourced	Variable resource availability

With Similarities

Enterprise	Open Source
Deployment Frequency	Release Frequency
Mean Time to Restore	Time To Remediate Vulnerabilities
Organizational Performance	Popularity

With Similarities

Enterprise	Open Source
Deployment Frequency	**Release Frequency**
Mean Time to Restore	Time To Remediate Vulnerabilities
Organizational Performance	**Popularity**

Faster
is Better

Hypothesis 1:

Projects that release frequently
have better outcomes.

Faster
is Better

Hypothesis 1:

Projects that release frequently have better outcomes. (validated)

5x more popular

Attract 79% more developers

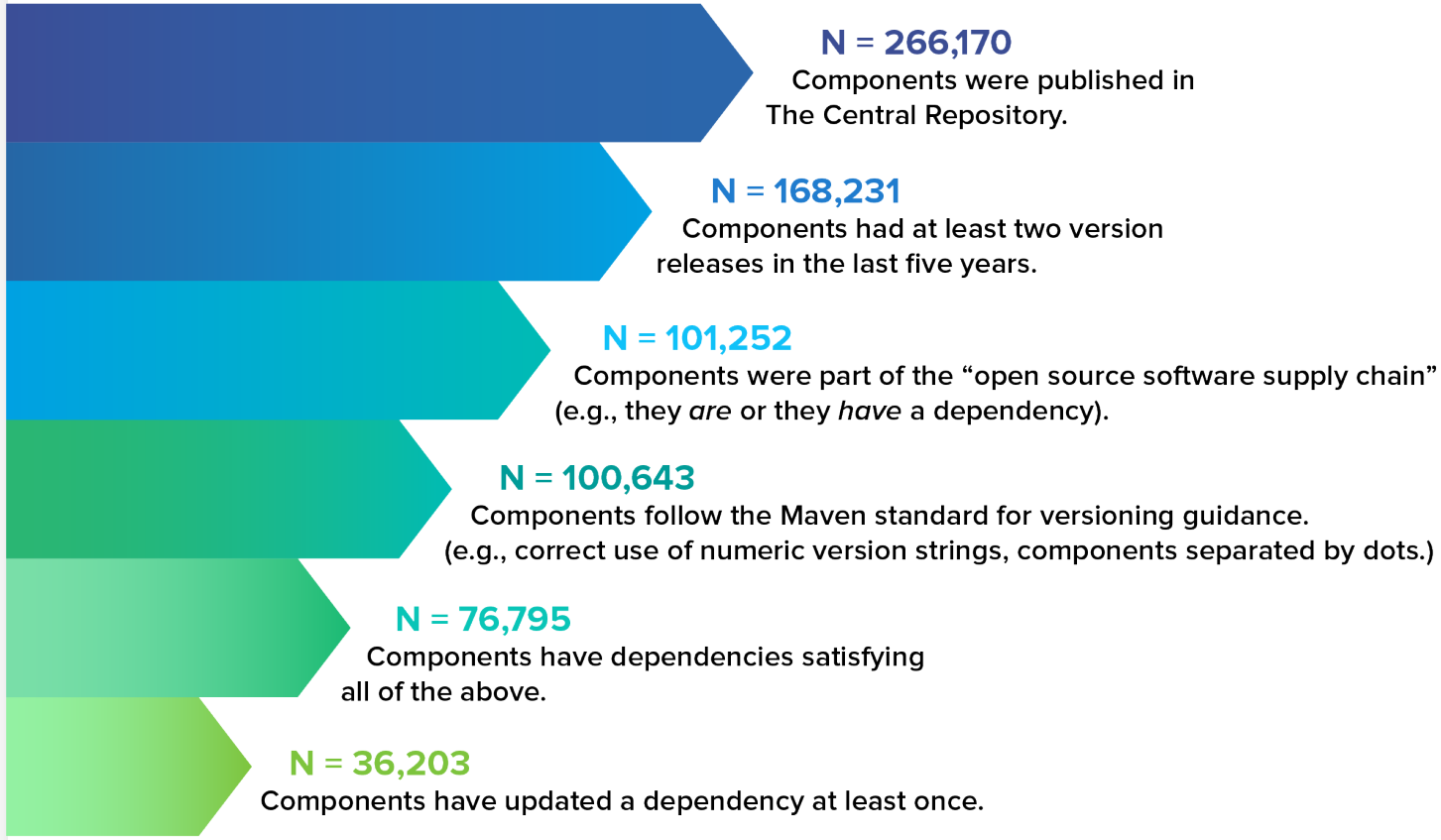
12% greater foundation support rates

With Similarities

Enterprise	Open Source
Deployment Frequency	Release Frequency
Mean Time to Restore	**Time To Remediate Vulnerabilities**
Organizational Performance	Download Counts

Are these effected by dev team size, number of dependencies, foundation support, etc.?

Constructing the Study Dataset (N = 36,203)



N = 266,170
Components were published in The Central Repository.

N = 168,231
Components had at least two version releases in the last five years.

N = 101,252
Components were part of the “open source software supply chain” (e.g., they *are* or they *have* a dependency).

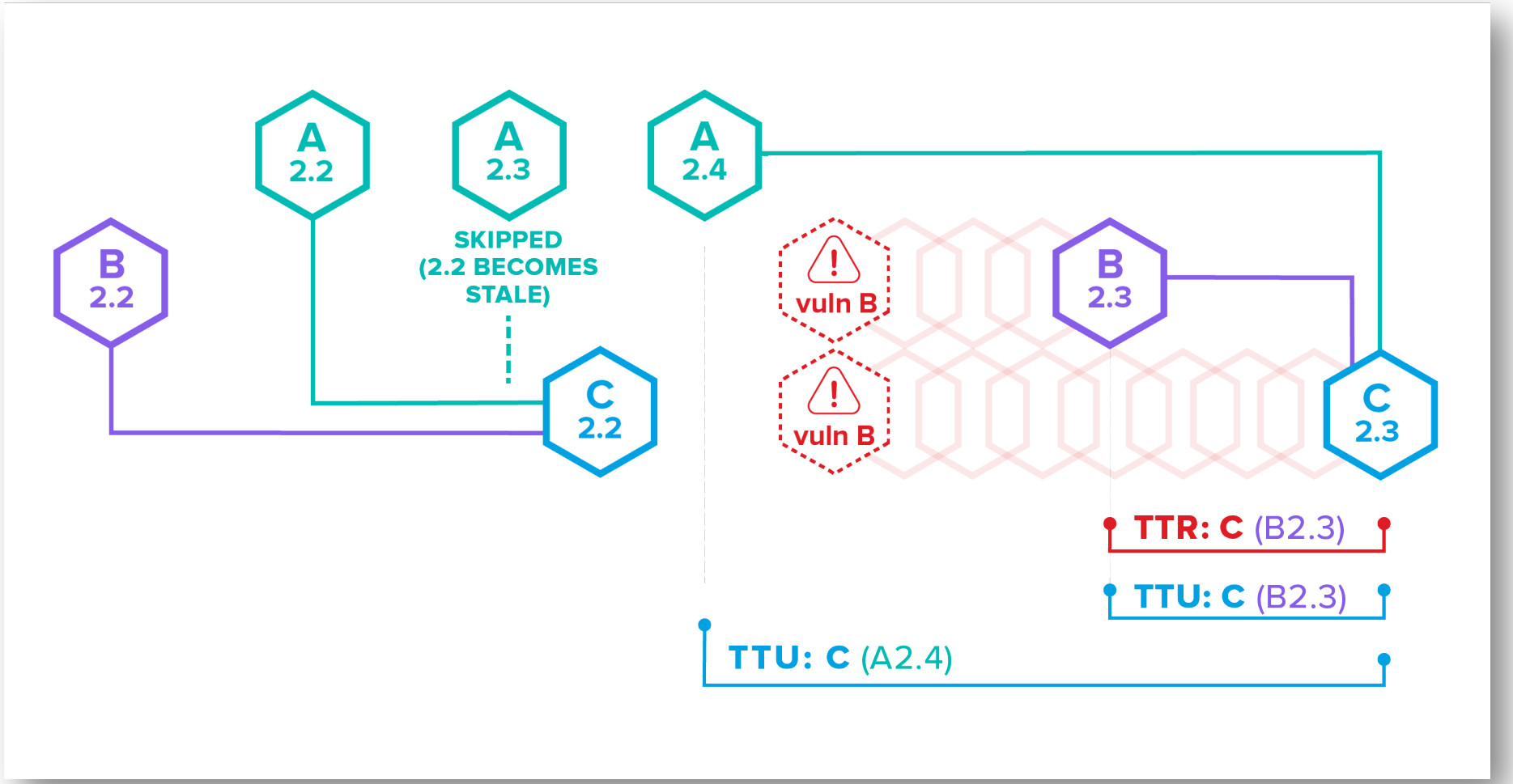
N = 100,643
Components follow the Maven standard for versioning guidance. (e.g., correct use of numeric version strings, components separated by dots.)

N = 76,795
Components have dependencies satisfying all of the above.

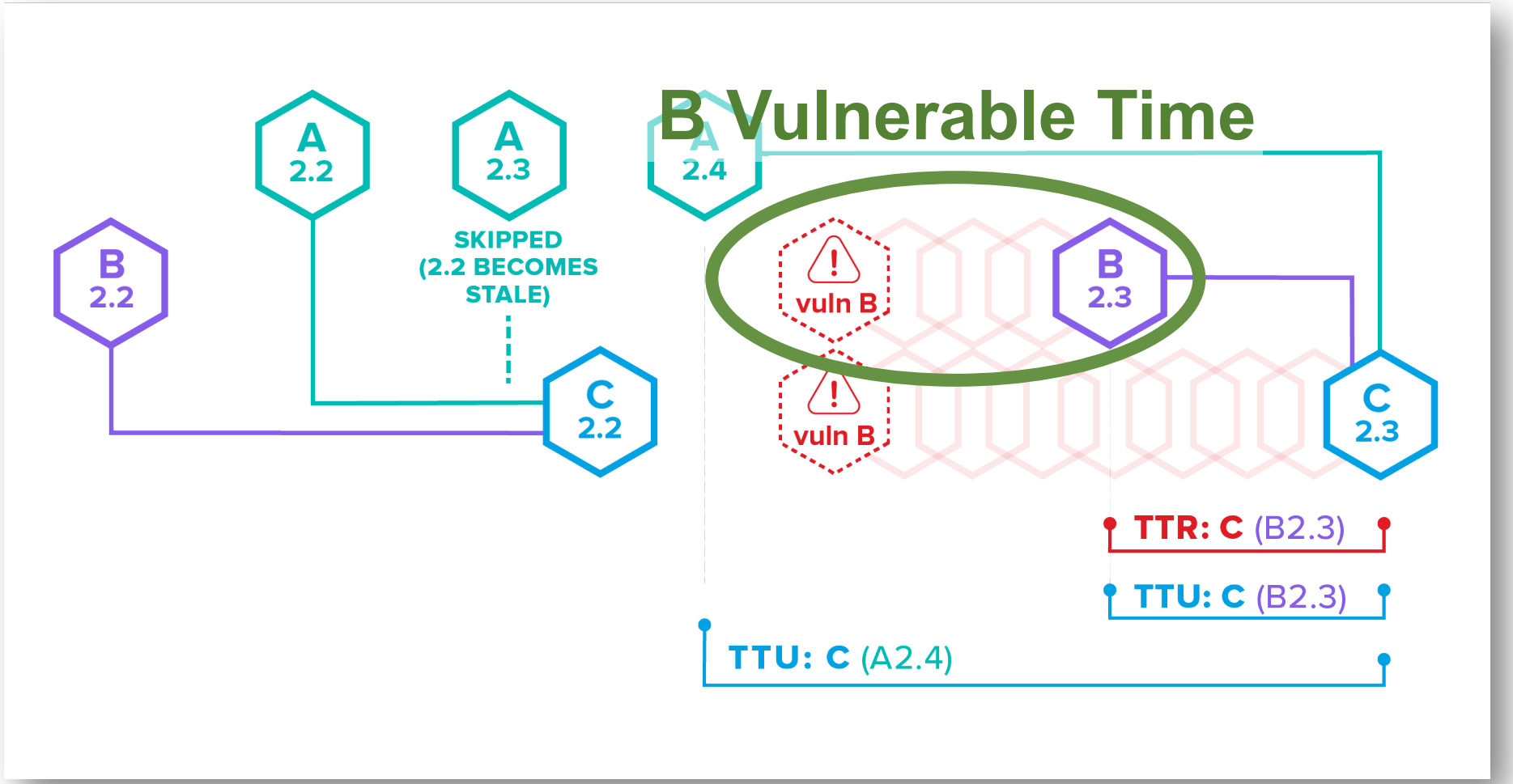
N = 36,203
Components have updated a dependency at least once.

Attributes	Measure
Popularity	Avg. Daily Maven Central Downloads
Size of Team	Avg. unique monthly contributors
Development Speed	Avg. commits per month
Release Speed	Avg. period between releases
Presence of CI	Presence of popular cloud CI systems
Foundation Support	Associated with an open source foundation
Security	More complicated
Update Speed	More complicated

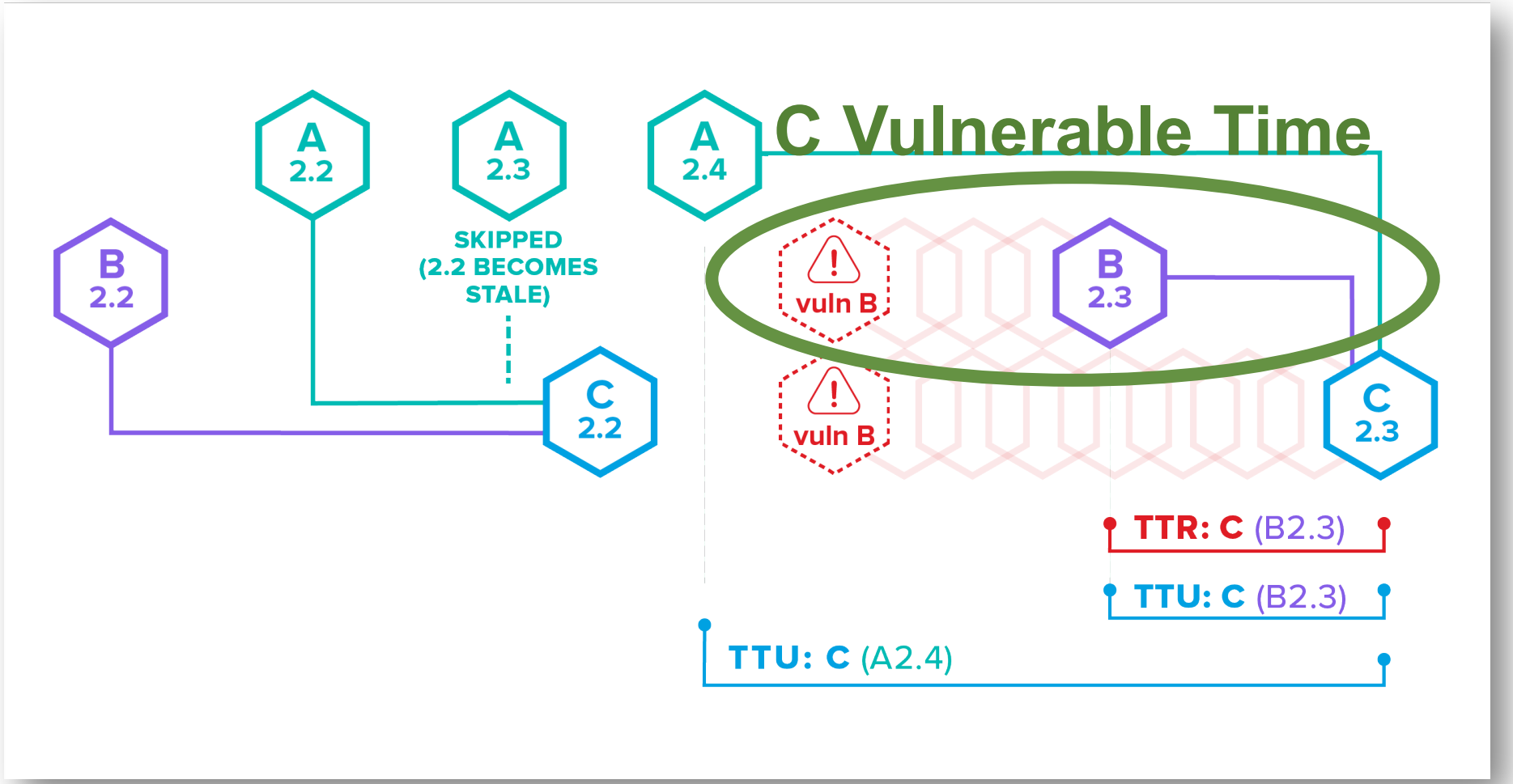
Security: Median Time To Remediate (MTTR)



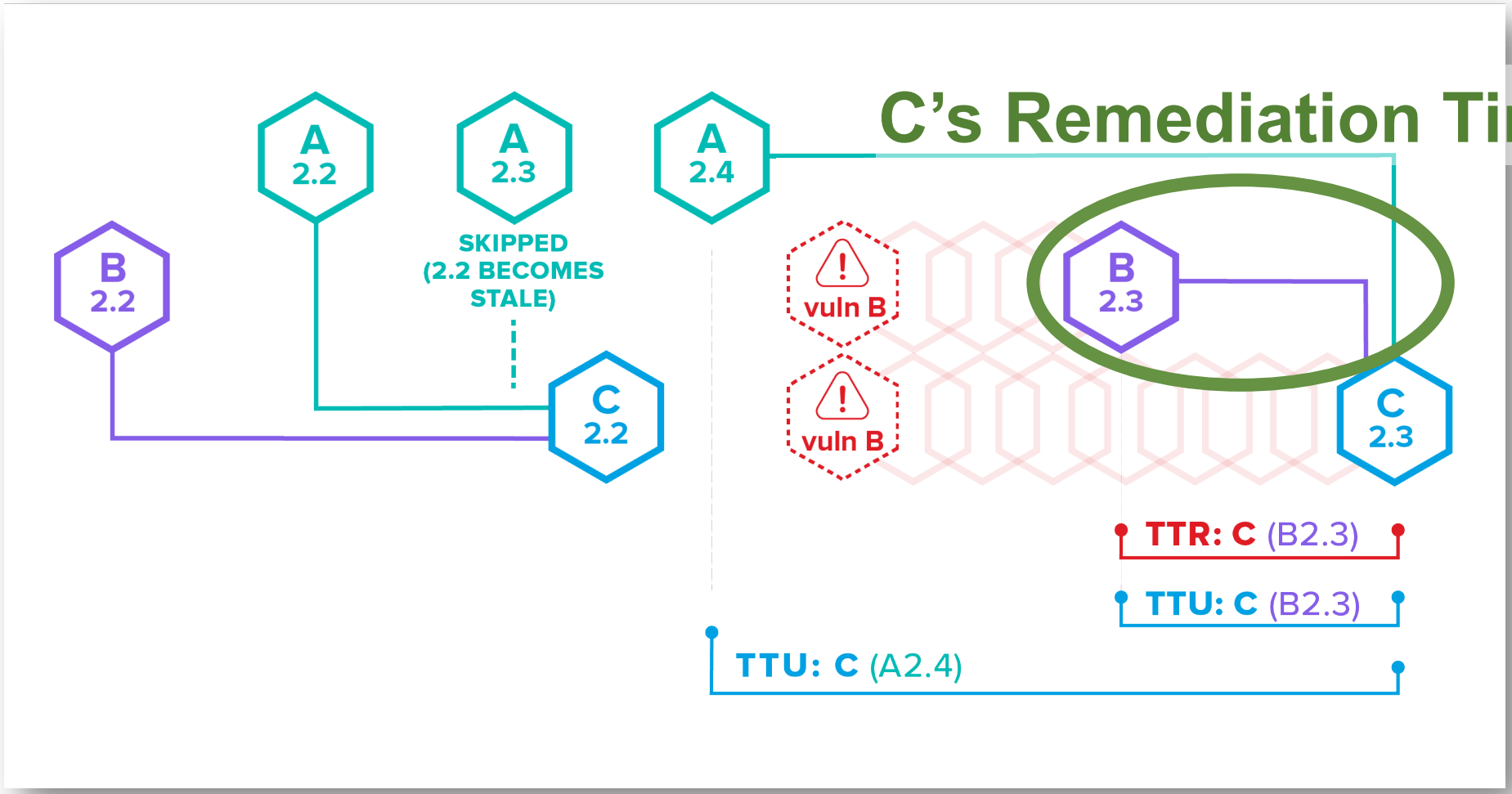
Security: Median Time To Remediate (MTTR)



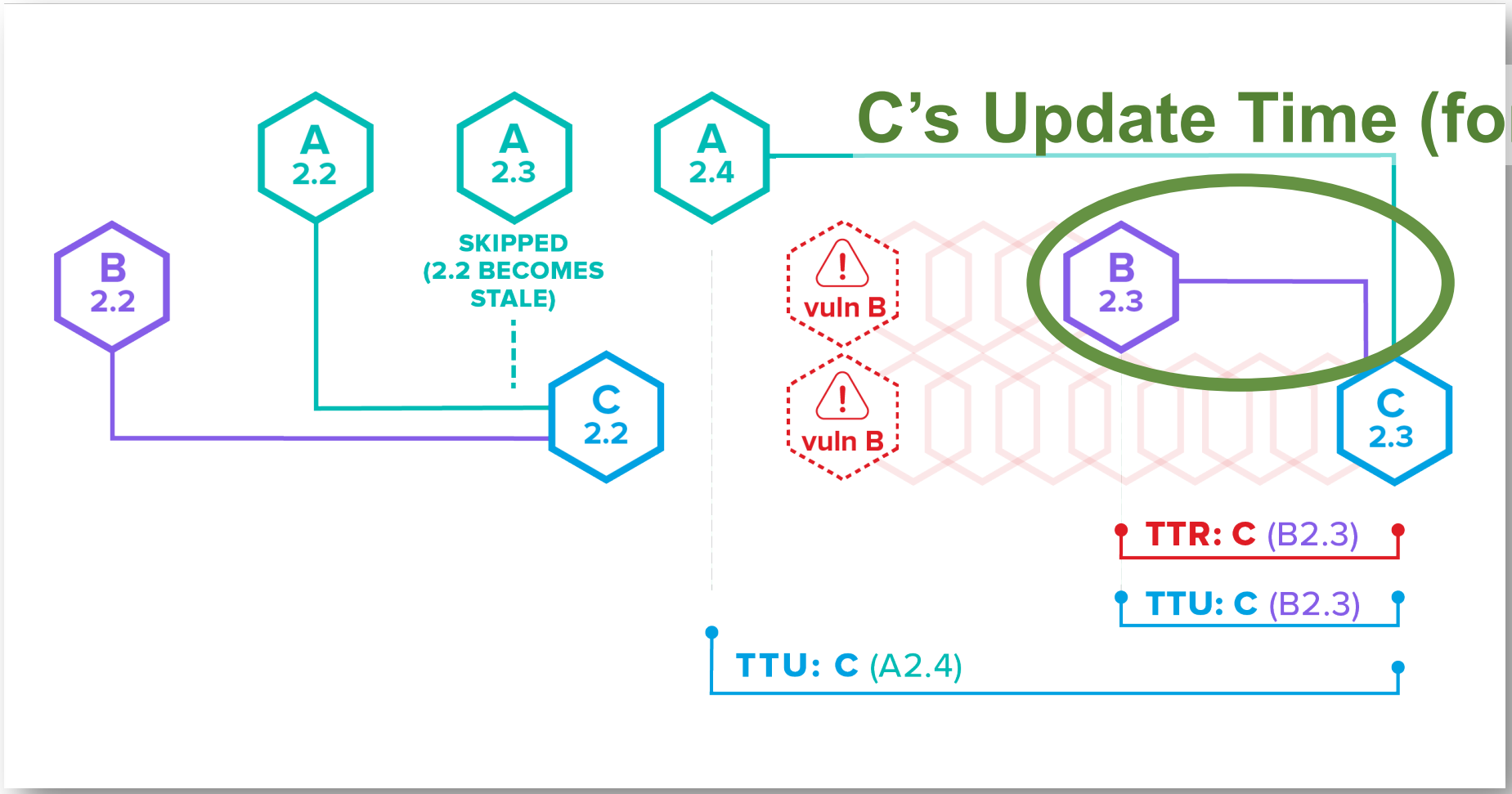
Security: Median Time To Remediate (MTTR)



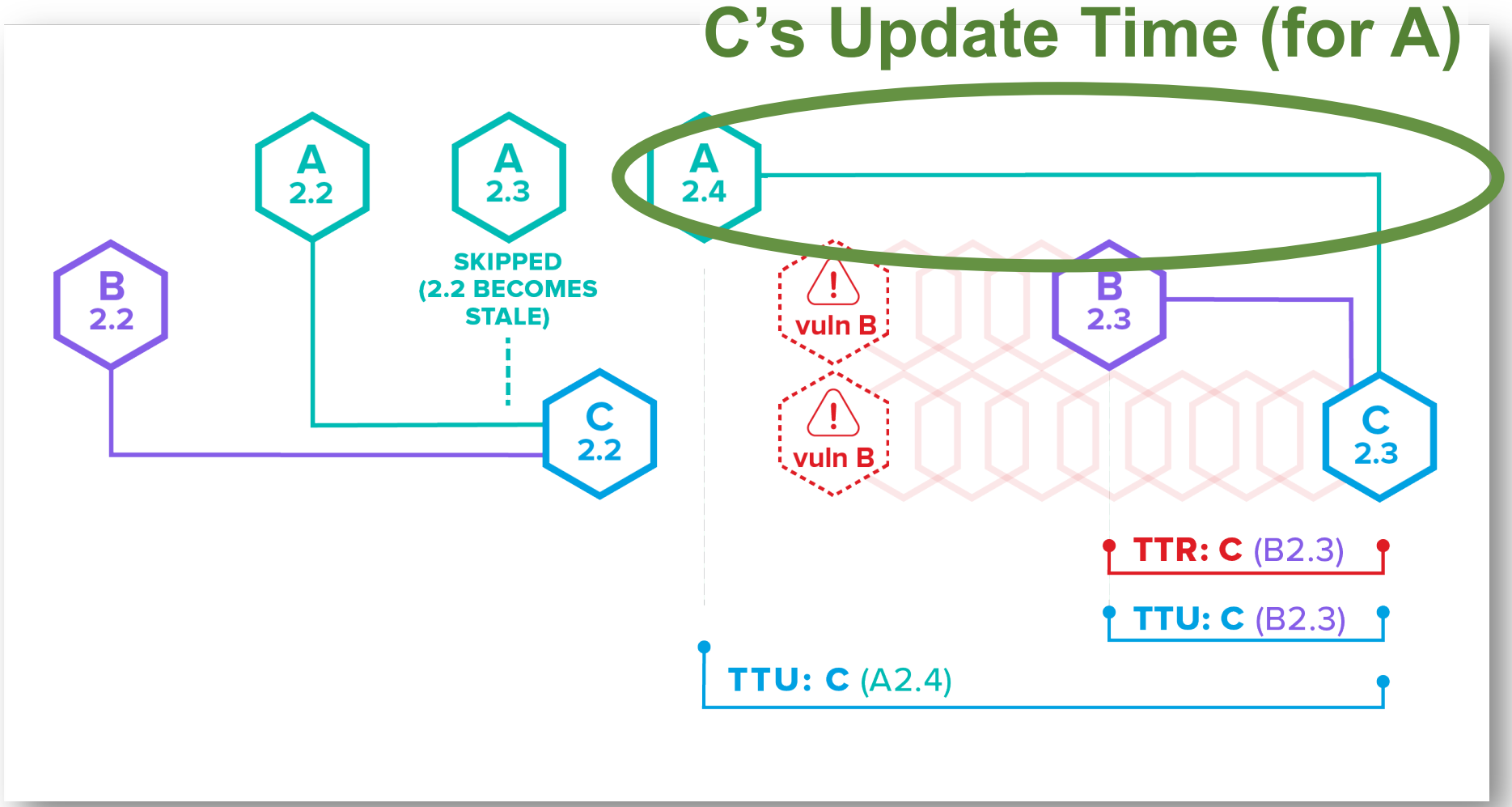
Security: Median Time To Remediate (MTTR)



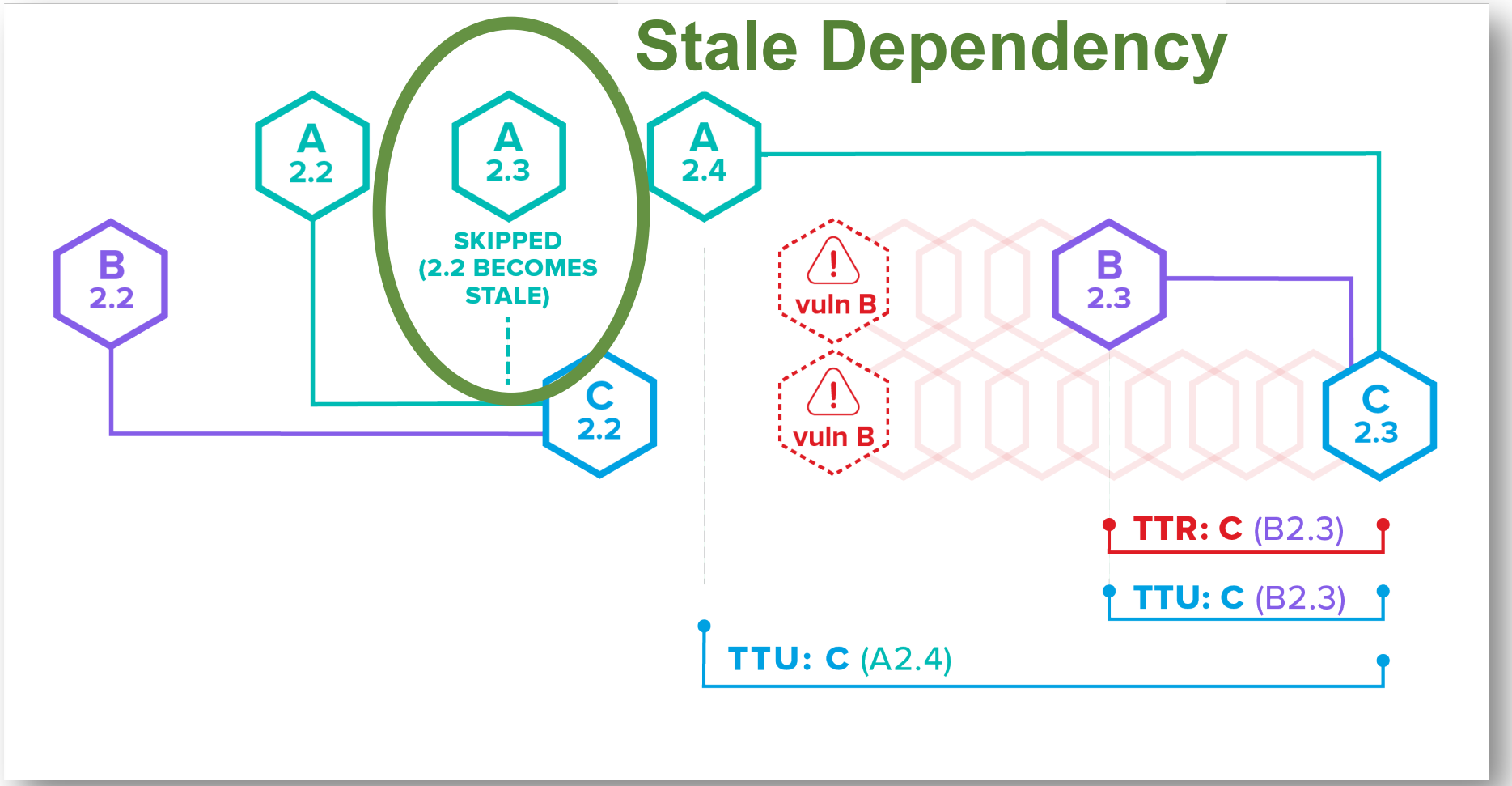
Security: Median Time To Update (MTTU)



Security: Median Time To Update (MTTU)



Security: Stale Dependencies



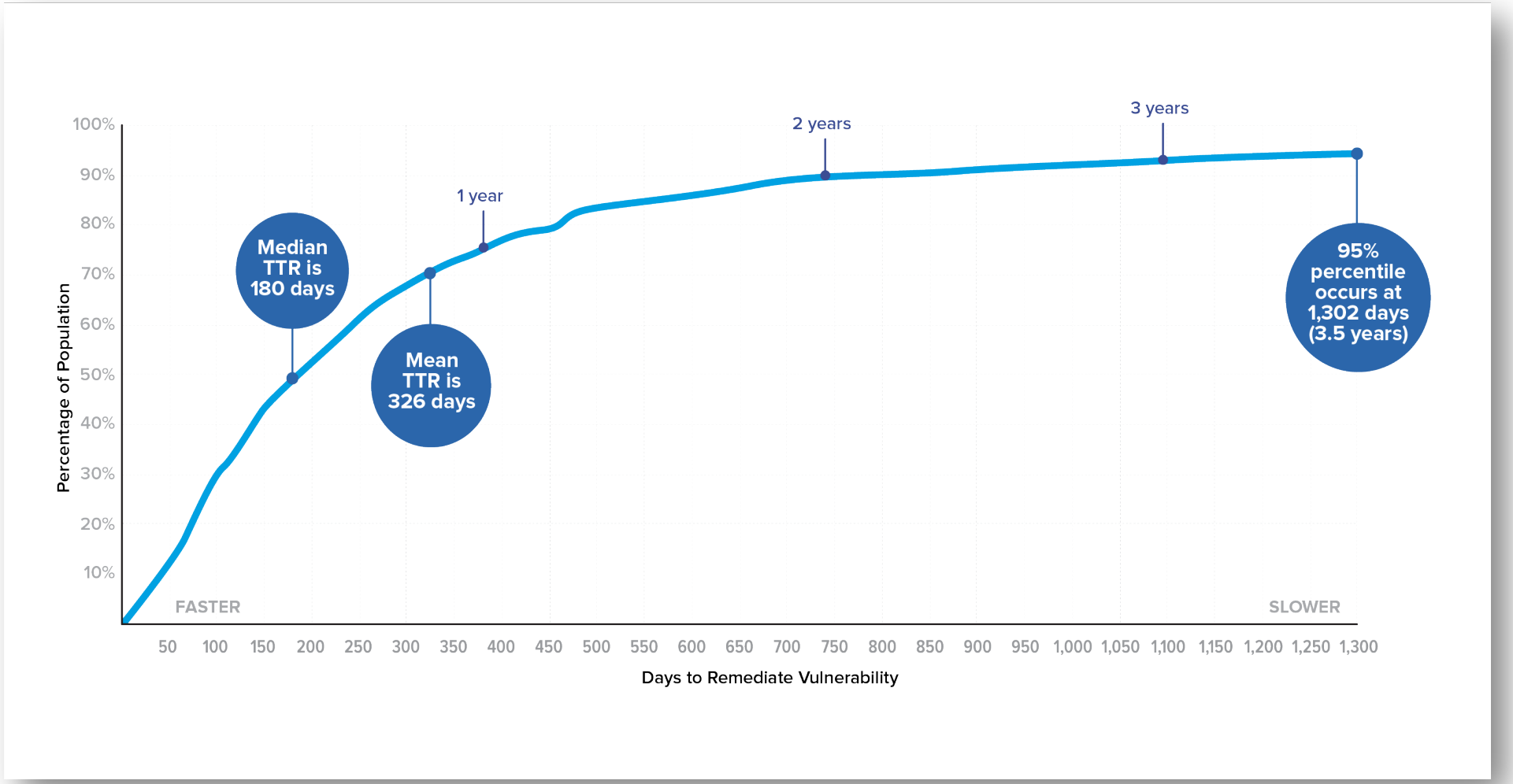
Three Key Metrics

Time To Remediate

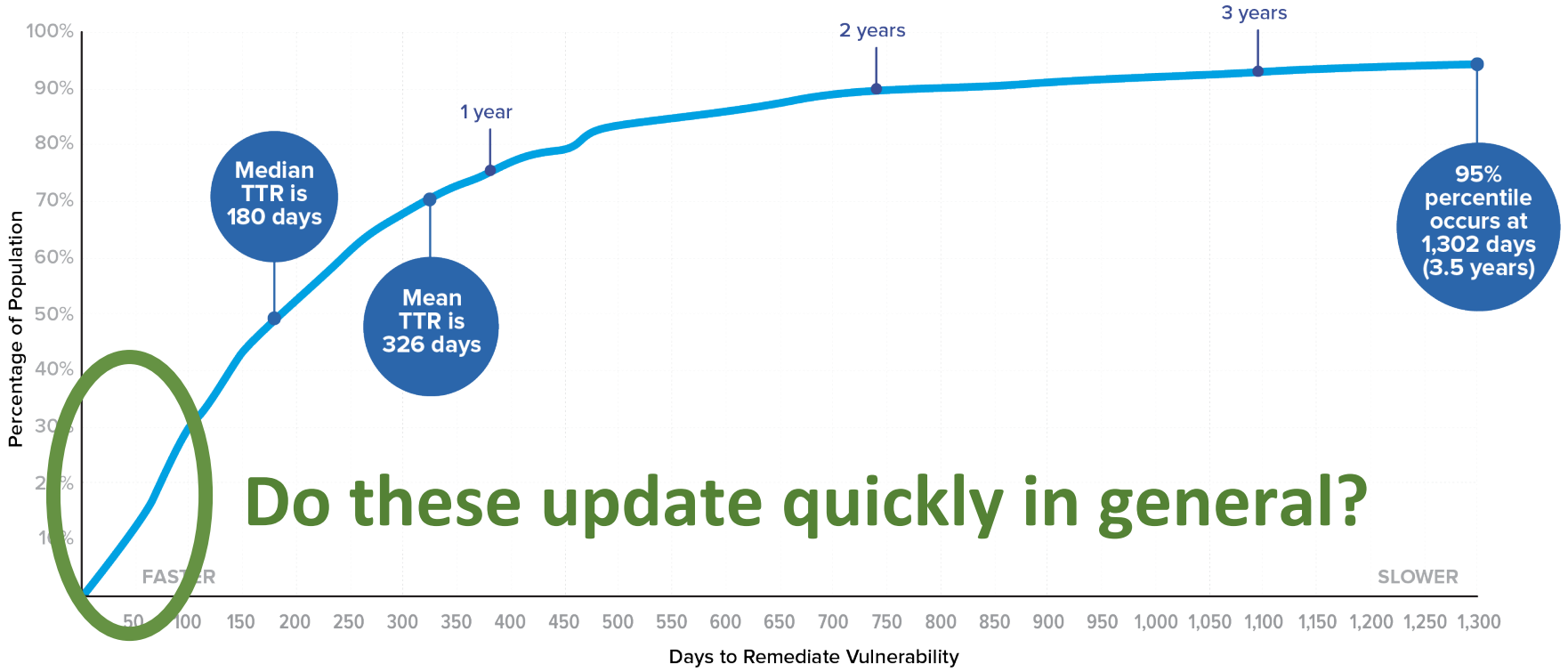
Time To Update

Stale Dependencies

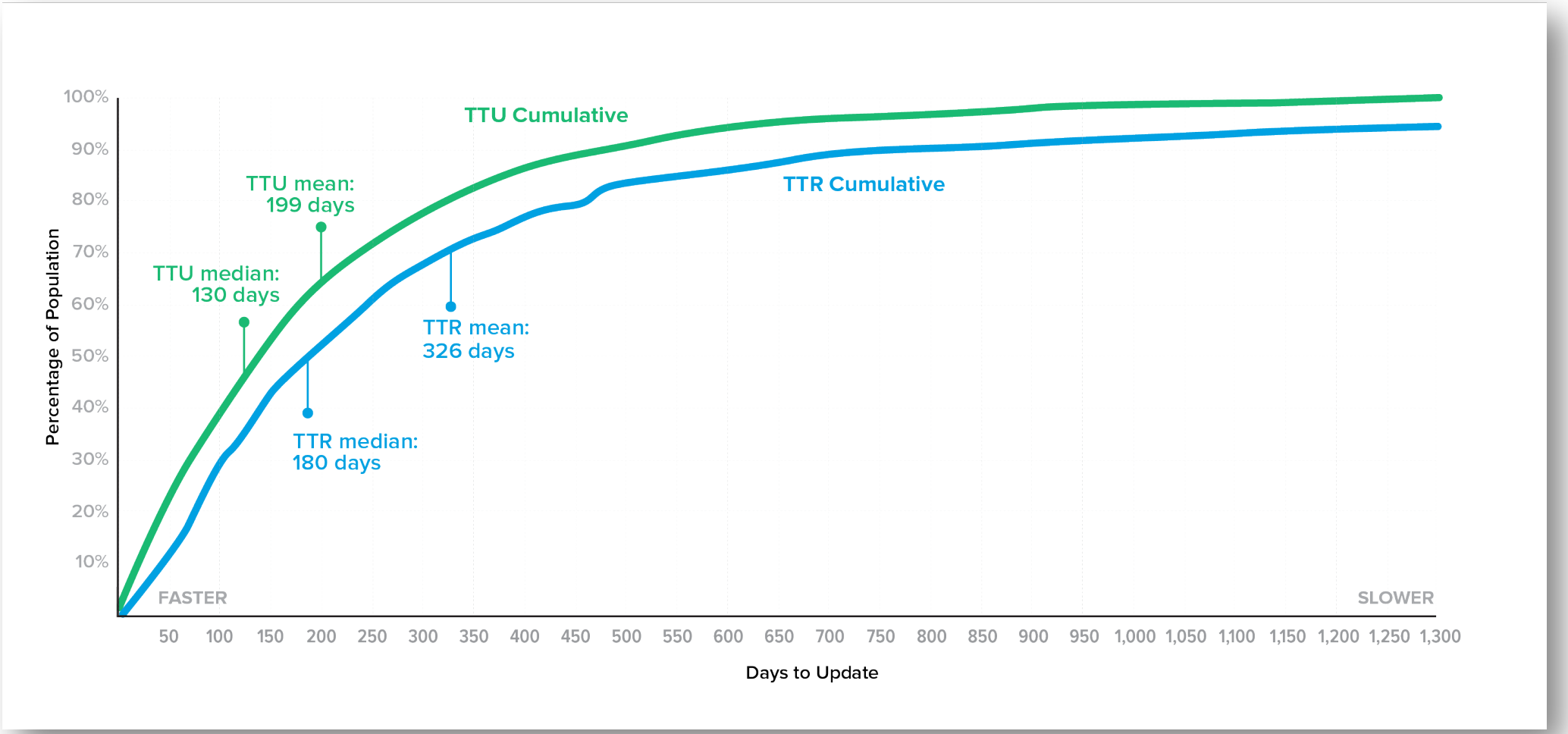
Time to Remediate Vulnerabilities



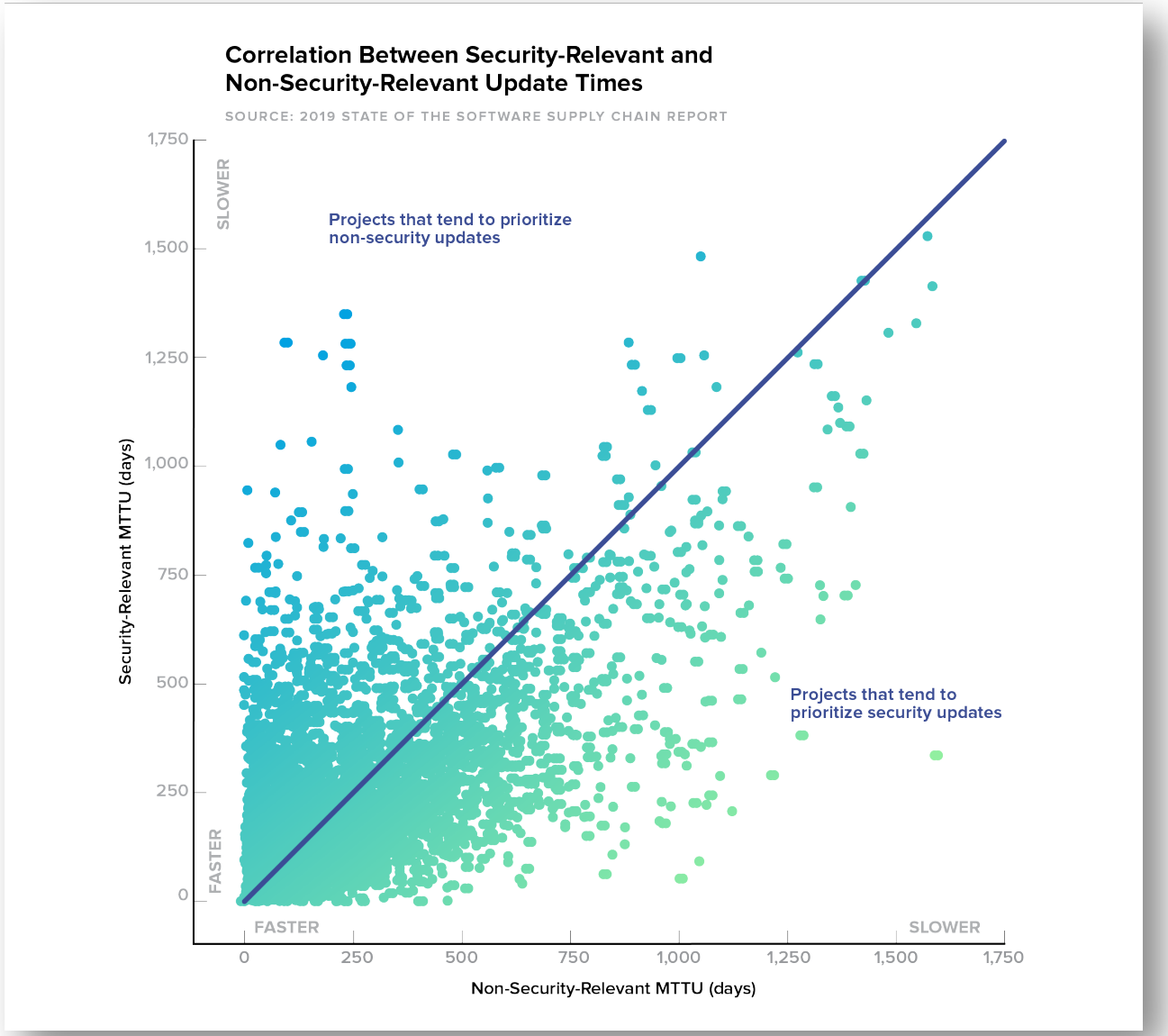
Time to Remediate Vulnerabilities



Time to Remediate (TRR) vs. Time to Update (TTU)



Time to Remediate (TTR) vs. Time to Update (TTU)



Pearson correlation 0.6

@stephenmagill

@weekstweets

Updating for Security

Most projects stay secure by staying up-to-date.

55% have MTTR and MTTU within 20% of each other.

Only 15% maintain better than average MTTR with worse than average MTTU.

Hypotheses

Hypothesis 2:

Projects that update dependencies more frequently are generally more secure.

Hypotheses

Hypothesis 2:

Projects that update dependencies more frequently are generally more secure. (validated)

Hypotheses

Hypothesis 2:

Projects that update dependencies more frequently are generally more secure. (validated)

Hypothesis 3:

Projects with fewer dependencies will stay more up-to-date.

Hypotheses

Hypothesis 2:

Projects that update dependencies more frequently are generally more secure. (validated)

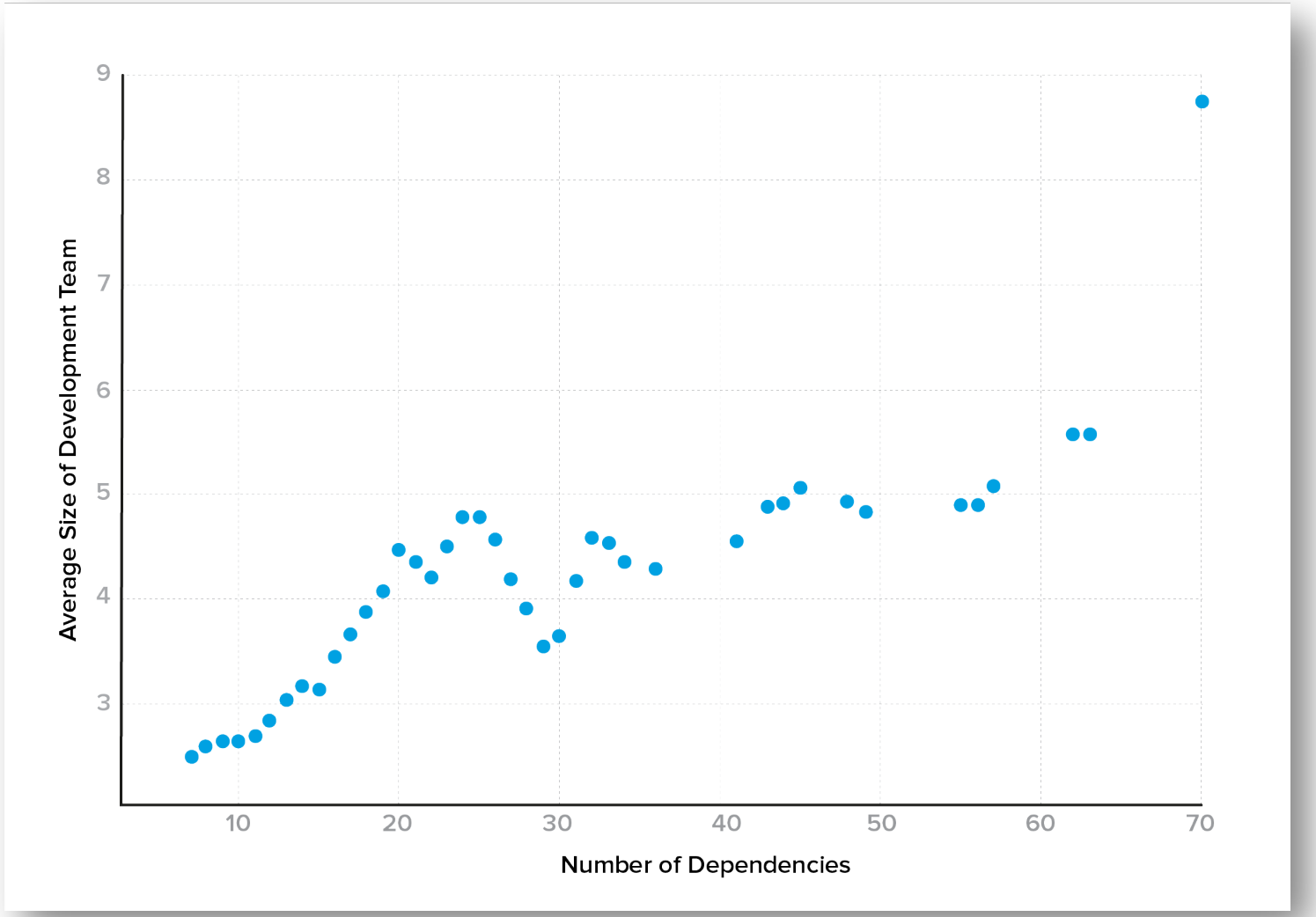
Hypothesis 3:

Projects with fewer dependencies will stay more up-to-date. (rejected)

Components with more dependencies actually have **better** MTTU.

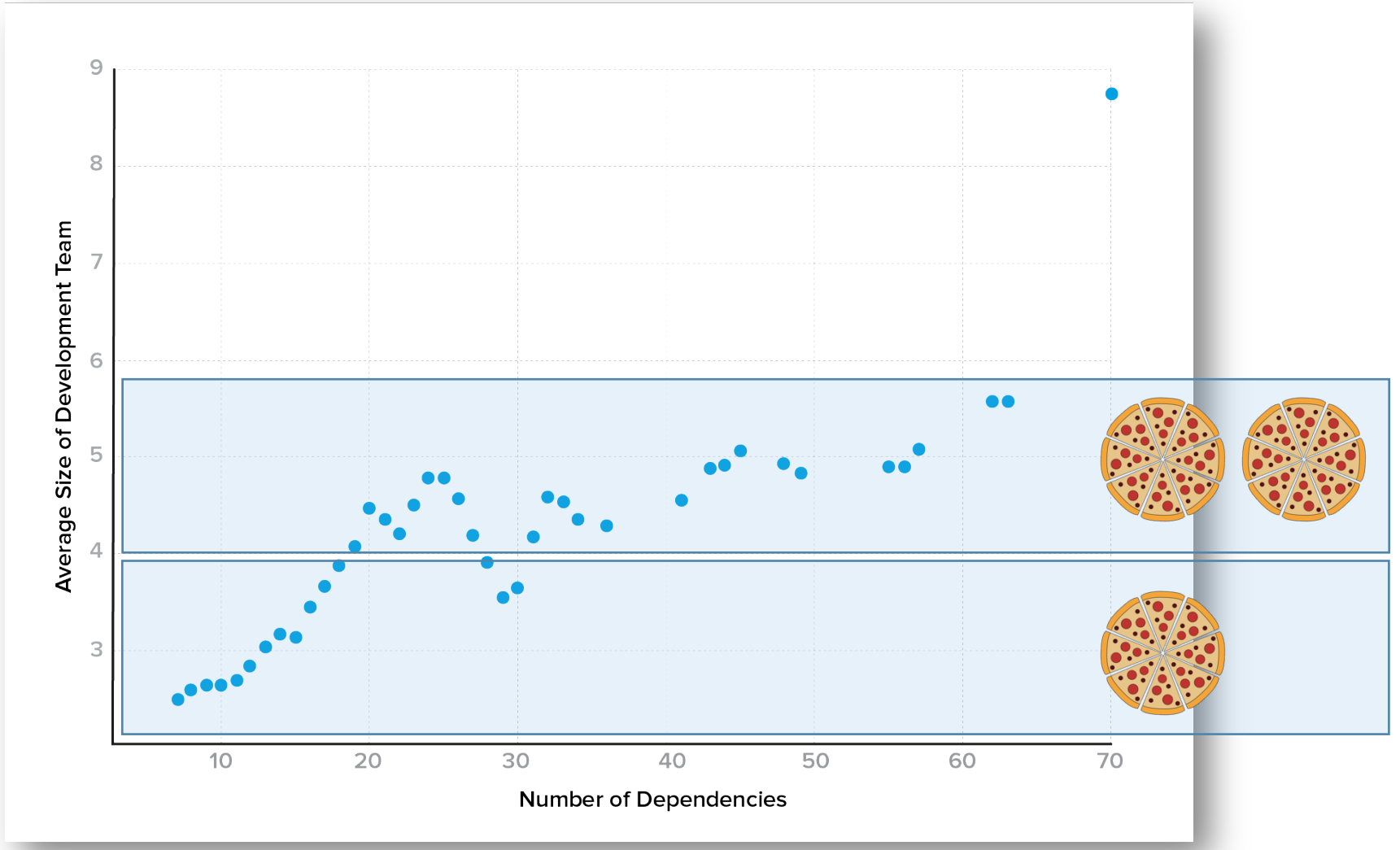
More dependencies correlate with larger development teams

Larger development teams have **50% faster MTTU** and release **2.6x more frequently**.



More dependencies correlate with larger development teams

Larger development teams have **50% faster MTTU** and release **2.6x more frequently**.



Hypotheses

Hypothesis 4:

More popular projects will be better about staying up-to-date.

Hypotheses

Hypothesis 4:

More popular projects will be better about staying up-to-date. (rejected)

Plenty of popular components with poor MTTU.

Popularity does not correlate with MTTU.

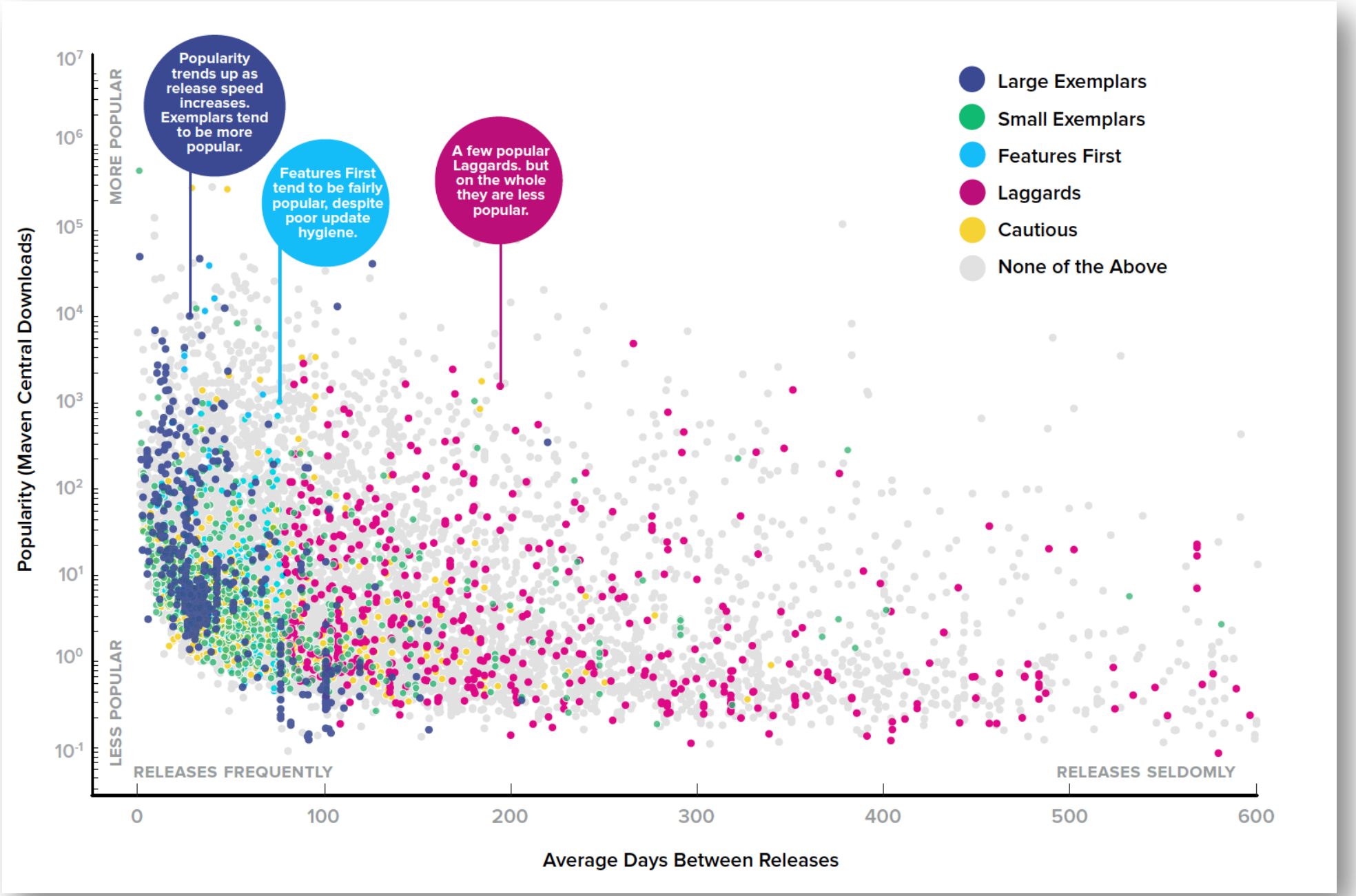
Most popular projects are not statistically different on average from others with respect to MTTU.

Groups

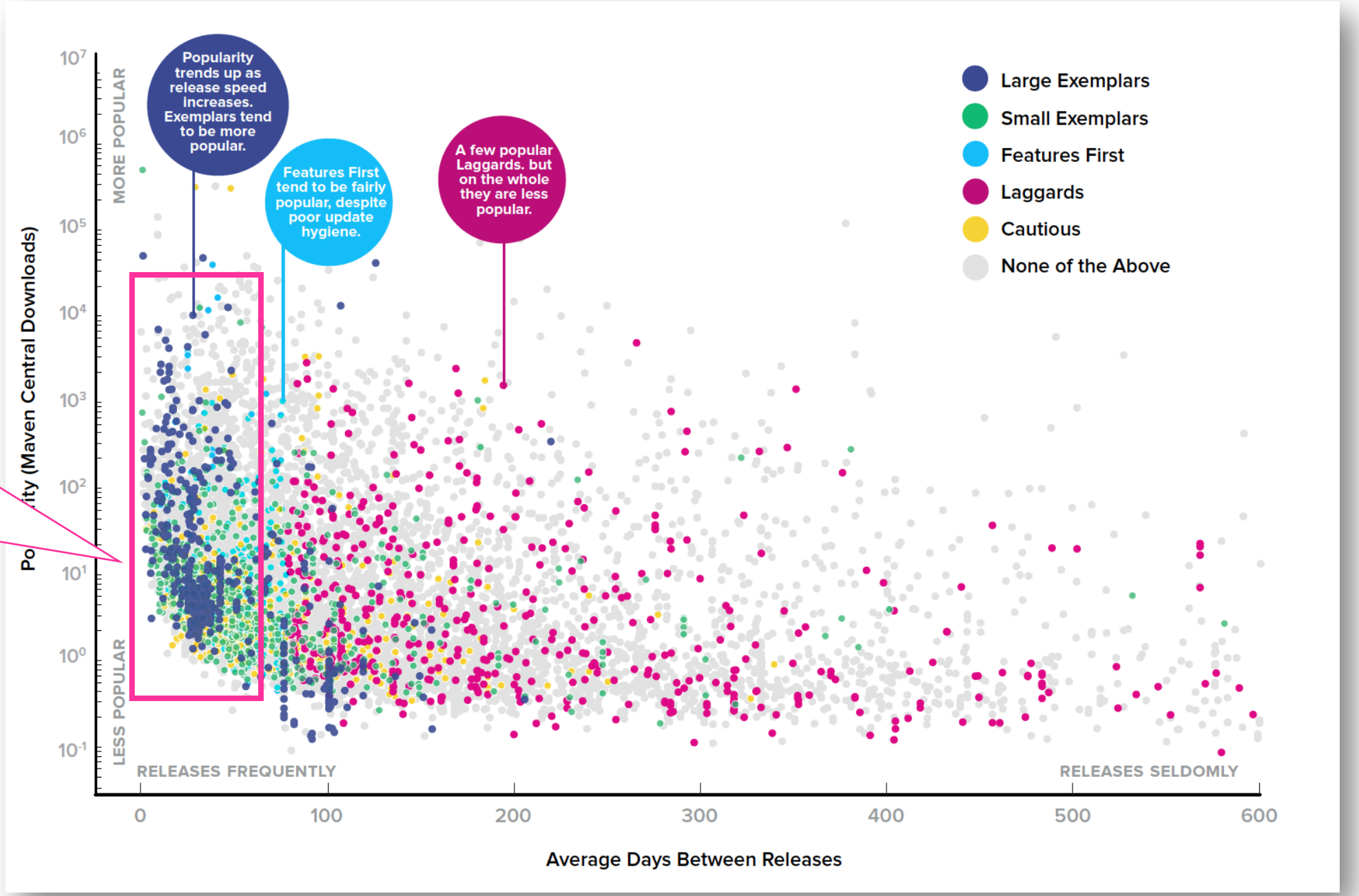
Small Exemplar (606)	Large Exemplar (595)	Laggards (521)	Features First (280)	Cautious (429)
Small development teams (1.6 devs), exemplary MTTU.	Large development teams (8.9 devs), exemplary MTTU, very likely to be foundation supported, 11x more popular.	Poor MTTU, high stale dependency count, more likely to be commercially supported.	Frequent releases, but poor TTU. Still reasonably popular.	Good TTU, but seldom completely up-to-date.

Rest of the population: 8142

Group popularity and release speed



Exemplars release fast and tend to be more popular.

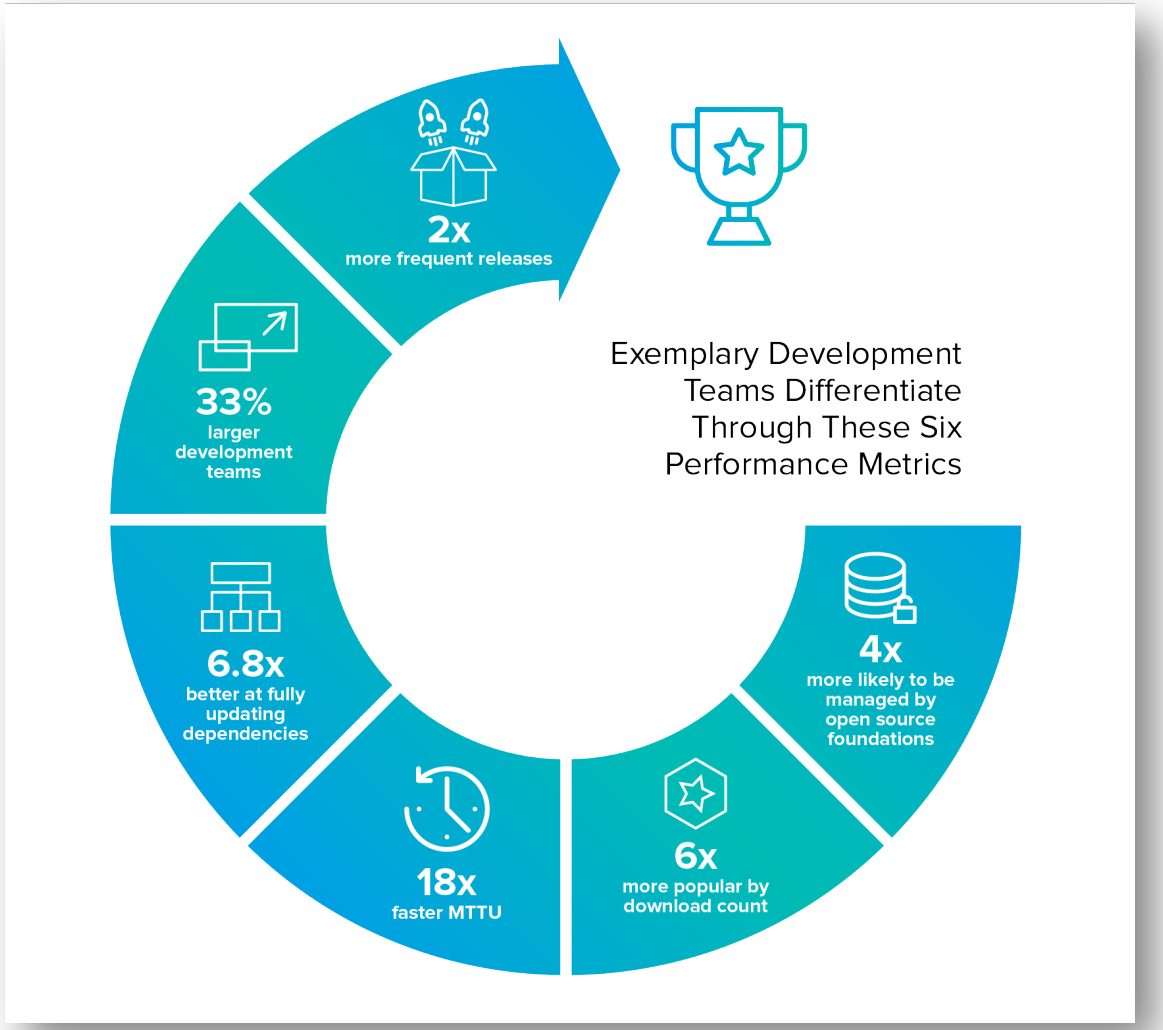


sonatype

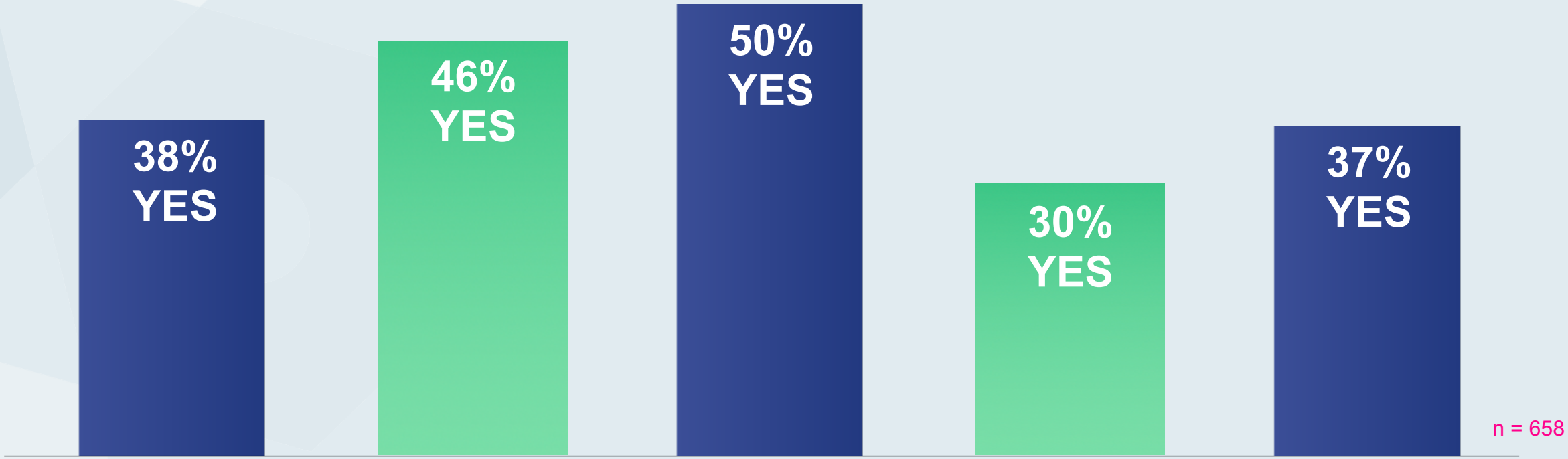
Not all popular projects are exemplary and release fast

sonatype

Exemplar Teams



Enterprise Devs Manage Dependencies



We **schedule updating** dependencies as part of our daily work

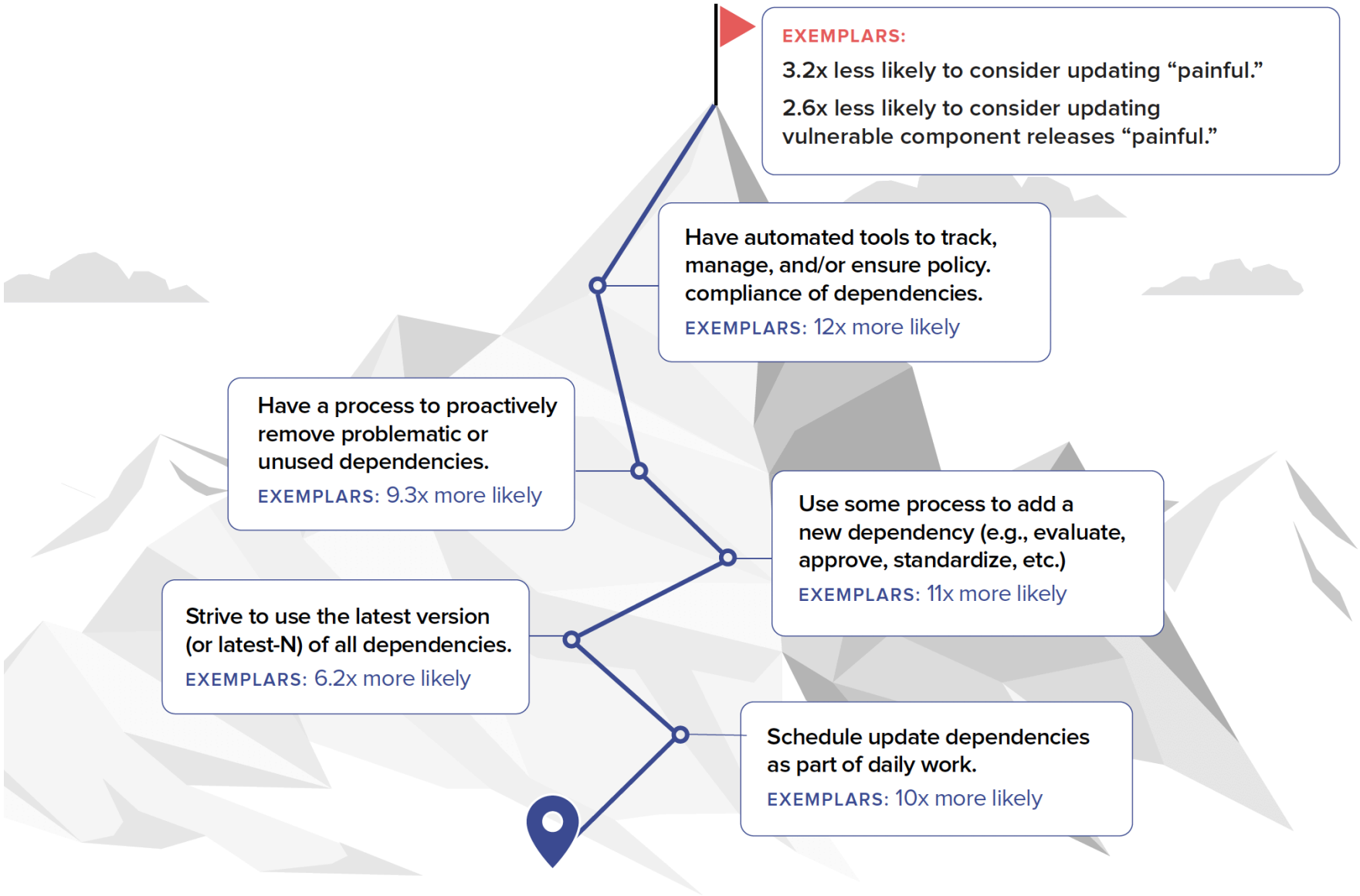
We **strive to use** the latest version (or latest-N) of all our dependencies

We **use some process** to add a new dependency (e.g., evaluate, approve, standardize, etc.)

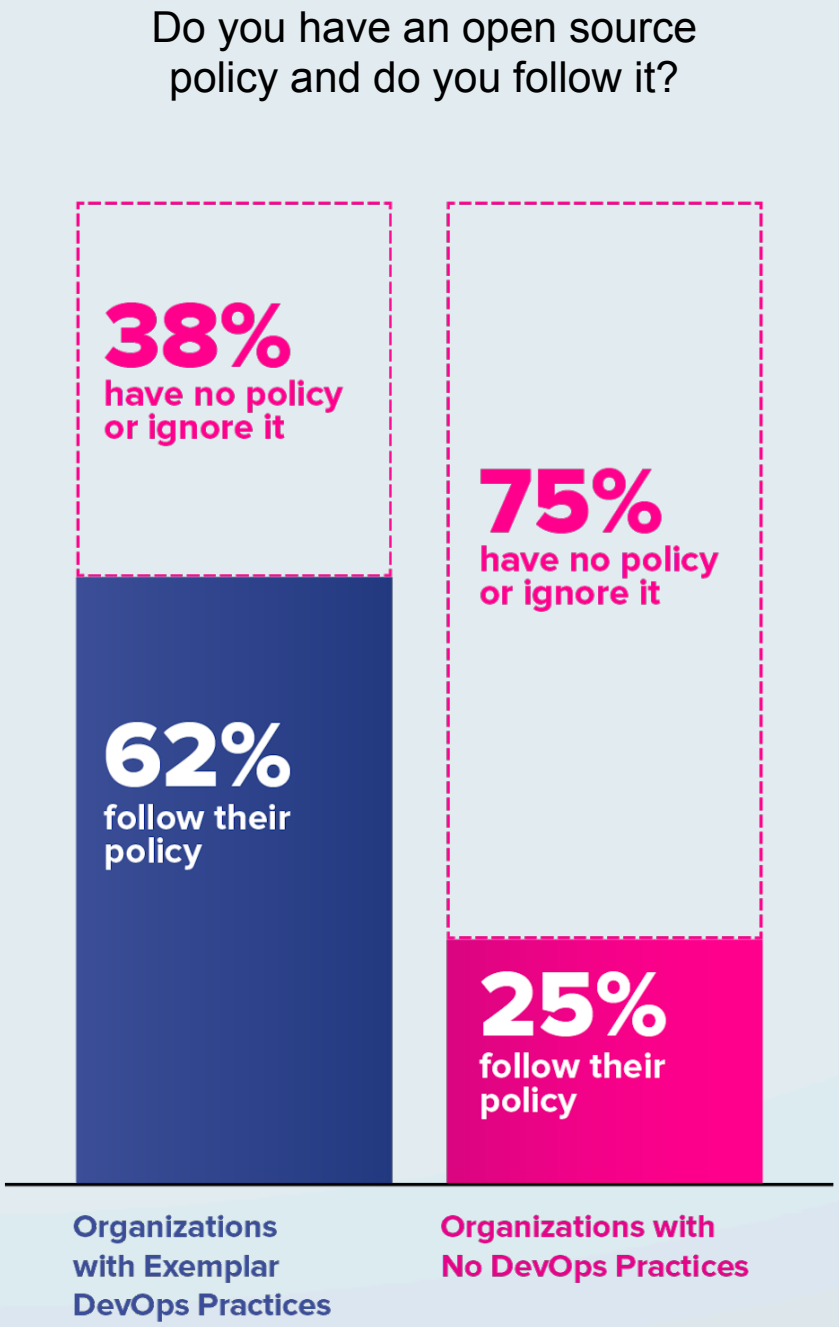
We have a process to **proactively remove** problematic or unused dependencies

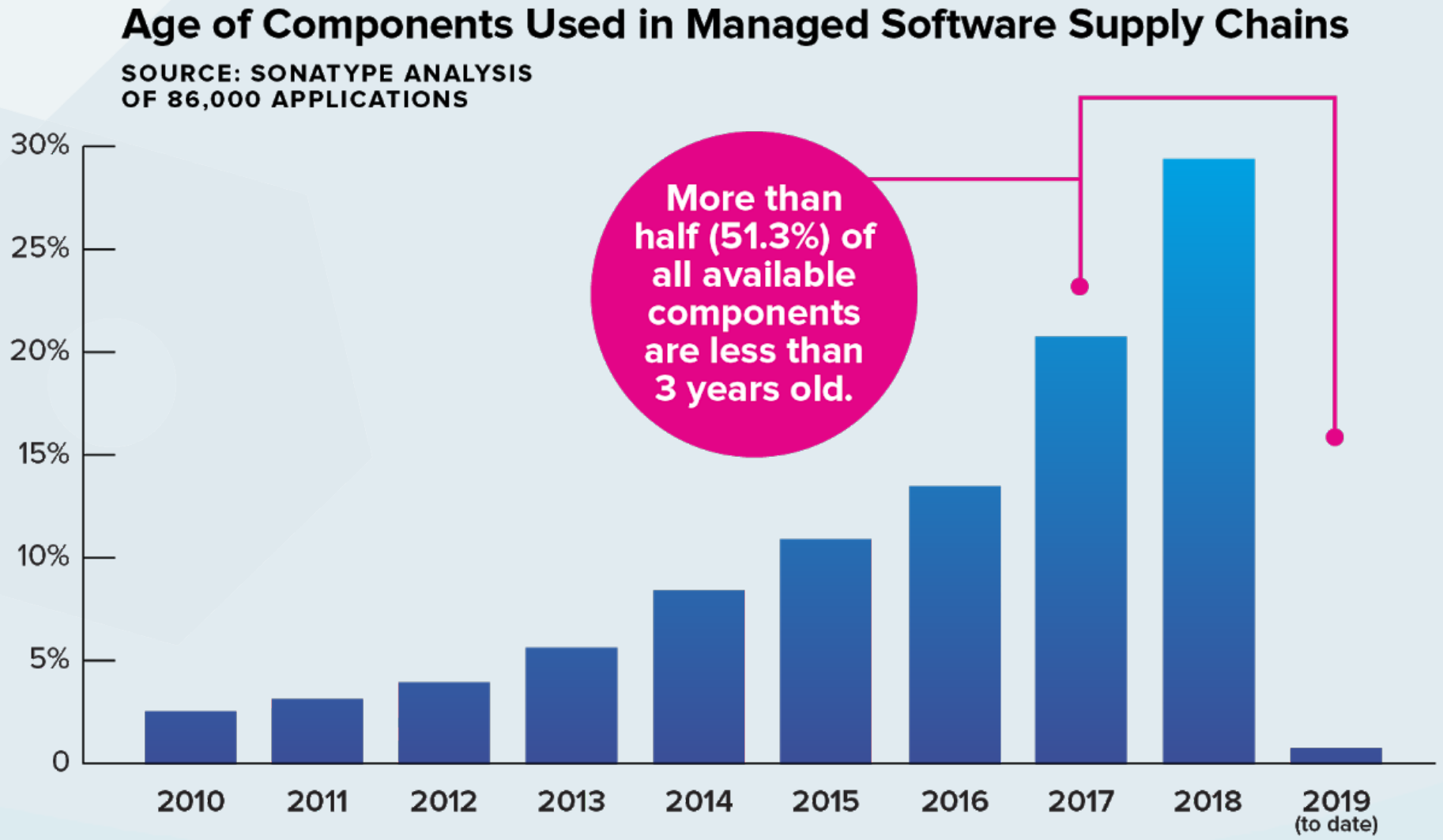
We have **automated tools** to track, manage, and/or ensure policy compliance of our dependencies

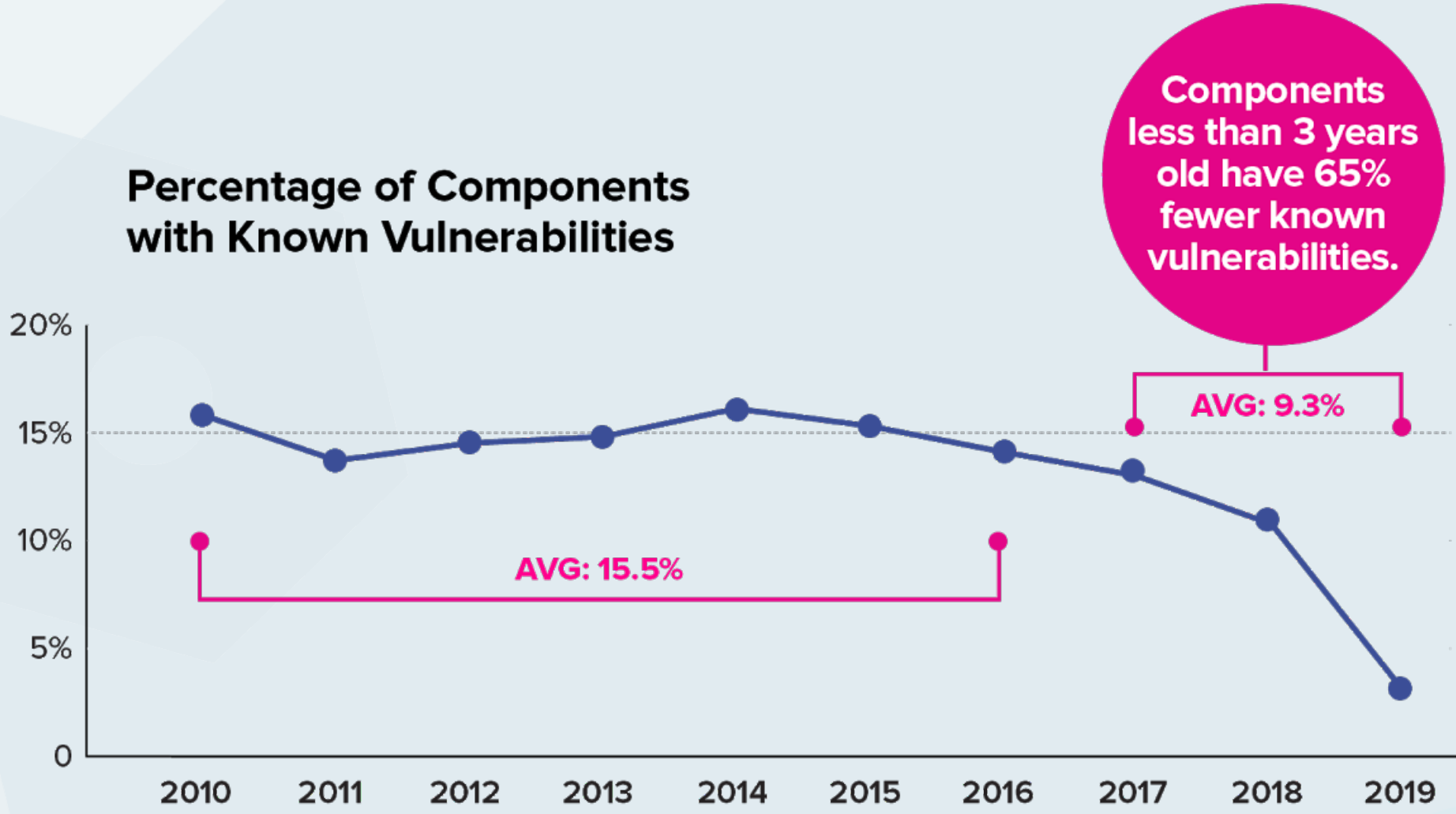
When Devs climb the mountain every day, it's easier.



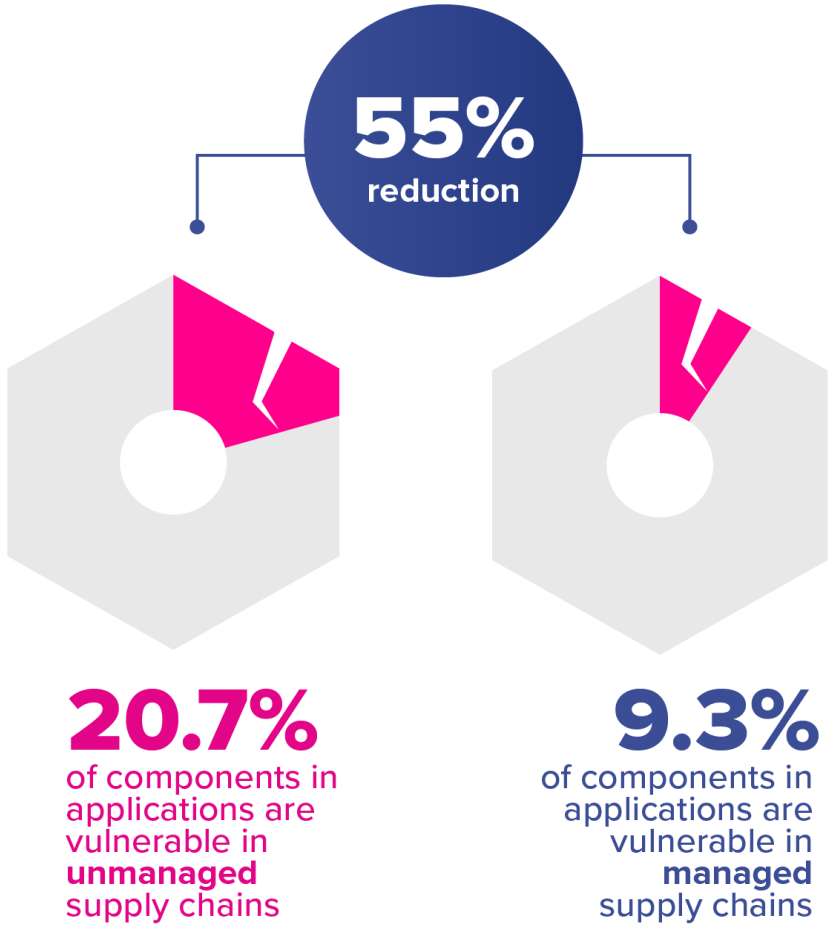
Automation continues to prove difficult to ignore




Age of Components Used in Managed Software Supply Chains
SOURCE: SONATYPE ANALYSIS
OF 86,000 APPLICATIONS
30%
25%
20%
15%
10%
5%
0
More than half (51.3%) of all available components are less than 3 years old.
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
(to date)


Percentage of Components with Known Vulnerabilities
Components less than 3 years old have 65% fewer known vulnerabilities.
20%
15%
10%
5%
0
AVG: 9.3%
AVG: 15.5%
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019

For organizations who tamed their supply chains, the rewards were impressive.



3 Quick Takeaways

- Start with observability
- Introduce criteria for component selection (your software suppliers)
- Aim for a minimum of 4 releases with 80% of dependencies updated

sonatype

weeks@sonatype.com

54,000 Tales: A Data-driven Discussion of Exemplar Development and DevSecOps Practices

Aurora 6 & 7
1140am – 1210pm

Jayne Groll
DevOps Institute

Dr. Stephen Magill
Galois

David Jones
Credit Suisse