

Core Infrastructure Initiative (CII) Best Practices Badge in 2019

Dr. David A. Wheeler
2019-03-14
dwheeler @ ida.org



Personal: dwheeler @ dwheeler.com,
Twitter: drdavidawheeler
GitHub & GitLab: david-a-wheeler
<https://dwheeler.com>

This presentation released under CC-BY-3.0+ license

- It is *not* the case that “all OSS* is insecure” ... or that “all OSS is secure”
 - Just like all other software, some OSS is (relatively) secure.. and some is not
- Heartbleed vulnerability in OpenSSL
 - Demonstrated in 2014 that some widely-used OSS didn't follow commonly-accepted practices & needed investment for security
- Linux Foundation created Core Infrastructure Initiative (CII) in 2014
 - “to fund and support critical elements of the global information infrastructure”
 - “CII is transitioning from point fixes to holistic solutions for open source security”



IDA | CII Best Practices Badge

- OSS tends to be more secure if it follows good security practices, undergoes peer review, etc.
 - How can we encourage good practices?
 - How can anyone know good practices are being followed?
- Badging project approach:
 - Identified a set of best practices for OSS projects
 - For *production* of OSS (for *license compliance*, see OpenChain)
 - Based on existing materials & practices
 - Created web application: OSS projects self-certify
 - If OSS project meets criteria, it gets a badge (scales!)
 - No cost, & independent of size / products / services / programming language
 - Self-certification mitigated by automation, public display of answers (for criticism), LF spot-checks, LF can override




To get your OSS project a badge, go to <https://bestpractices.coreinfrastructure.org/>

IDA | Criteria

cii best practices **passing**

cii best practices silver

cii best practices gold

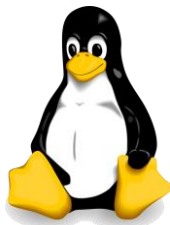
- Three badge levels (passing, silver, gold)
 - For higher levels, must meet previous level
- Passing: 
 - Captures what well-run projects typically already do
 - Not “they should do X, but no one does that”
 - 66 criteria in 6 groups:
 - Basics, Change Control, Reporting, Quality, Security, Analysis
- Silver: Harder but possible for 1-person projects
- Gold requires multiple developers
 - bus factor > 1*, 2-person review

Source: <https://github.com/coreinfrastructure/best-practices-badge/blob/master/doc/criteria.md>

IDA | Badge scoring system

- To obtain a badge, all:
 - MUST and MUST NOT criteria (42/66) must be met
 - SHOULD (10/66) met, OR unmet with justification
 - Users can see those justifications & decide if that's enough
 - SUGGESTED (14/66) considered (met or unmet)
 - People don't like admitting they didn't do something
 - In some cases, URL required in justification (to point to evidence; 8/66 require this)

IDA | Some major projects with a best practice badge

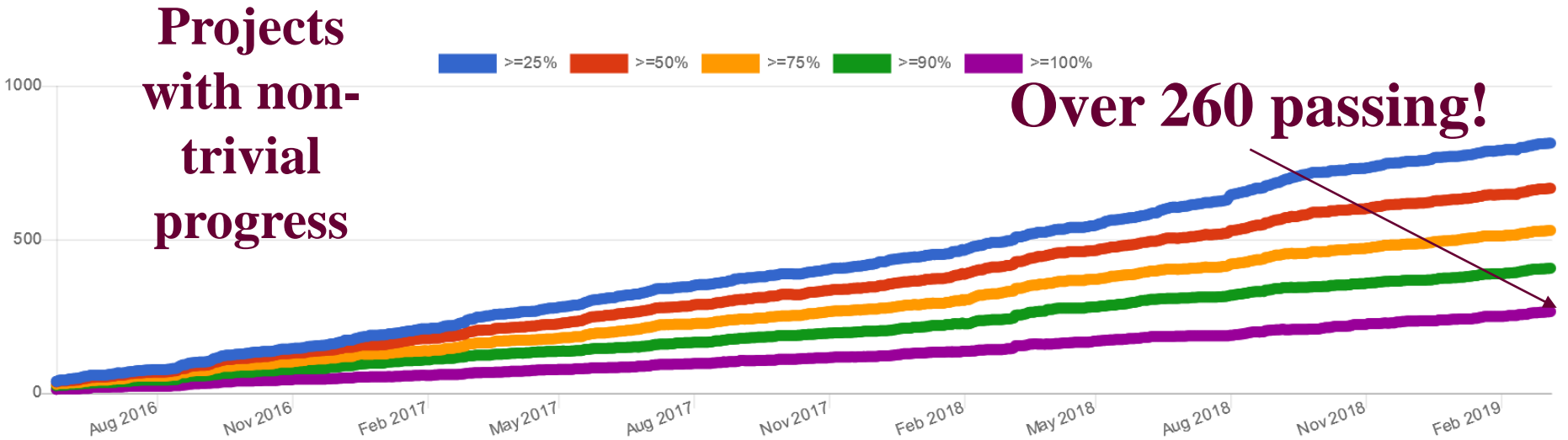
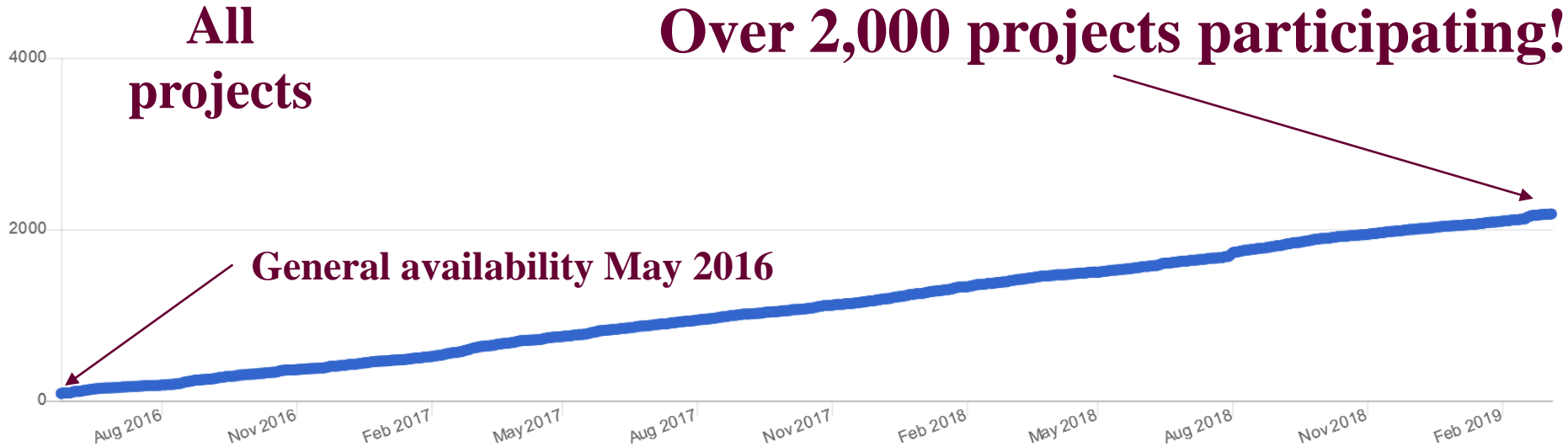


IDA | Lots of projects participating & getting badges!



- 2,178 participating projects (1,016 on 2017-09-19)
- 265 passing projects (105 on 2017-09-19)

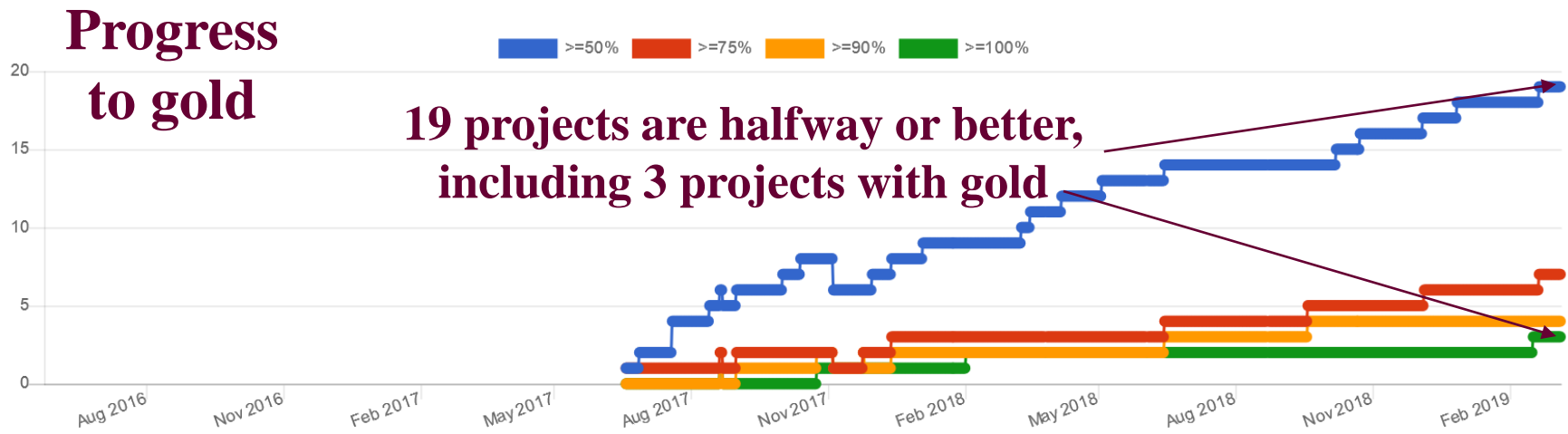
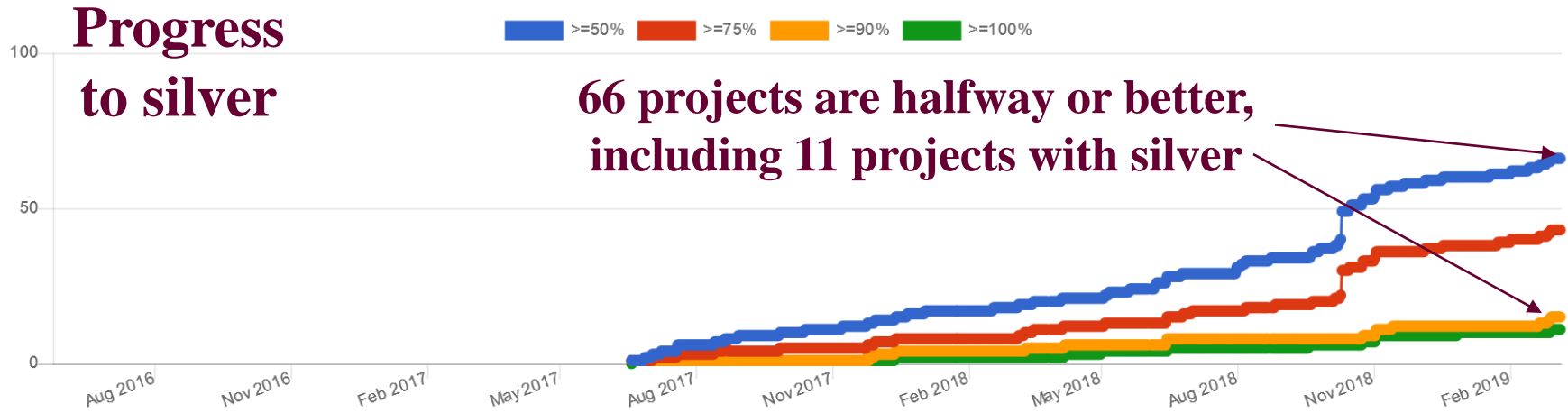
Data as of 2019-03-06



IDA | What about silver & gold?

- Silver & gold level badges intentionally harder to get (more demanding)
- For now we've focused on getting projects participating & passing, not silver/gold
 - *We want* projects to earn silver/gold
 - Non-passing projects appear to be in especially bad shape - focus on the bigger problem!
- Currently only 3 gold projects & 11 projects with silver (including gold earners)
 - But this measure hides the steady progress made by many projects...

IDA | Many projects working towards silver & gold



IDA | Some communities encouraging badges

- Cloud Native Computing Foundation (CNCF)*
 - Maturity levels: Sandbox → incubating → graduated
 - For graduated level must “have achieved and maintained a CII Best Practices Badge.”
 - Containerd recently graduated, has passing badge
- R community discussing recommending badges
 - 2018 survey:
 - 90% believe badge will provide value to the R community’s package developers or package users
 - 77% saying it has benefit for both developers and users
 - 74% would be willing to try it
 - Multiple R packages tried it out & began working towards badges as part of discussion
 - DBI passing
 - Close to passing include ggplot2, covr, dodgr, netReg

Sources: CNCF Graduation Criteria v1.2

https://github.com/cncf/toc/blob/master/process/graduation_criteria.adoc

“Should R Consortium Recommend CII Best Practices Badge for R Packages: Latest Survey Results” <https://www.r-consortium.org/blog/2018/07/26/should-r-consortium-recommend-cii-best-practices-badge-for-r-packages-latest-survey-results>

- Can easily embed current badge image
 - ``
 - Easily shows *current* state on GitHub, etc.
- REST API enables easy JSON data access
 - Including project database download for analysis
 - See <https://github.com/coreinfrastructure/best-practices-badge/blob/master/doc/api.md>
- Cross Origin Resource Sharing (CORS)
 - Enables data access from client-side JavaScript
 - E.g., for fancy client-side dashboards

IDA | Example: CNCF landscape

- CNCF landscape <<https://landscape.cncf.io/>> easily accesses badge data




Cloud Native **Graduated**

Open Source Software

License **Apache License**

CII Best Practices **passing**

 **Tweet** 500

containerd

Cloud Native Computing Foundation (CNCF)

Runtime · Container Runtime

An open and reliable container runtime

Website	https://containerd.io/		
Repository	https://github.com/containerd/containerd 🔄 ★ 3,637		
Crunchbase	https://www.crunchbase.com/organization/cloud-native-computing-found...		
LinkedIn	https://www.linkedin.com/company/cloud-native-computing-foundation		
Twitter	@containerd	Latest Tweet	this week
First Commit	3 years ago	Latest Commit	this week
Contributors	167	Headcount	11-50
Headquarters	San Francisco, California		

IDA | Sample clarifications

- vulnerabilities_fixed_60_days (PR #1188)
 - “There MUST be no unpatched vulnerabilities of medium or high severity that have been publicly known for more than 60 days.”
 - Added: “... this badge criterion, like other criteria, applies to the individual project. Some projects are part of larger umbrella... An individual project often cannot control the rest, but an individual project can work to release a vulnerability patch in a timely way.”
- hardened_site (PR #1187)
 - “The project website, repository (if accessible via the web), and download site (if separate) MUST include key hardening headers... [GitHub is known to meet this]”
 - Added: “Static web sites with no ability to log in via the web pages may omit the CSP and X-XSS-Protection HTTP hardening headers, because in that situation those headers are less effective.”

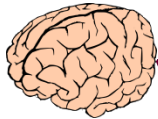
IDA | Most common challenges for getting a badge

- All projects 90%+ but not passing (2019-03-07)
 - 265 projects. MUST with Unmet or “?” => Top 10 challenges:

#	Criterion	%miss	Old rank#
1	vulnerability_report_process	21%	1
2	tests_are_added	17%	3
3	vulnerability_report_private	15%	4
4	know_secure_design	13%	9
5	vulnerabilities_fixed_60_days	13%	24
6	test_policy	13%	5
7	know_common_errors	13%	7
8	static_analysis	11%	8
9	static_analysis_fixed	11%	21
10	sites_https	9%	2



Tests



Know
secure
development



Analysis



Vulnerability
reporting



HTTPS

Fixing

This data is as of
2019-03-07,
old rank from
2017-09-06

Mostly same challenges as 2017-09-06. HTTPS becoming less of a problem, dropped from #2 to #10. Unclear why fixing things has become bigger problem..!

IDA | BadgeApp dependencies and security

- Tiny amount of new code in our system...
- Because almost all code is reused
 - Direct dependencies = 75 gems
 - Direct AND indirect dependencies = 197 gems
 - Plus OS, language runtime, RDBMS, etc.
- Today a key security concern for most projects is vulnerabilities through their dependencies
 - Minimize dependencies, ask them to minimize their run-time dependencies, sanity check of direct dependencies
 - Package manager: Track what we have, trivially update packages
 - Dependency tools*: detect & report packages with known vulnerabilities (GitHub + bundle audit)
 - Thorough automated tests: enable quick update, test, & ship to production (we have 100% coverage)
 - Other measures, esp. hardening (such as CSP), reduce risk in meantime

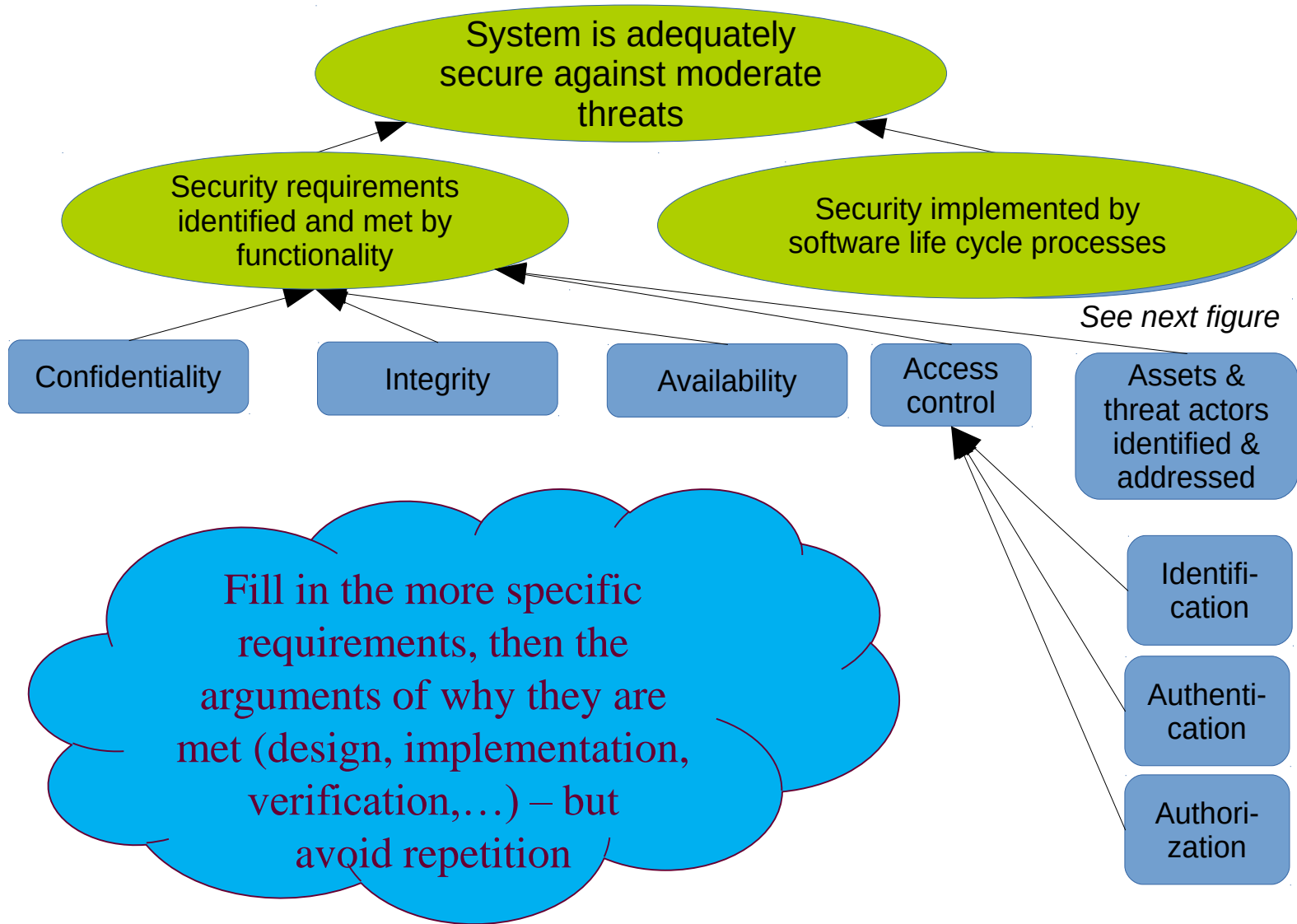
* Origin analysis / software composition analysis tools

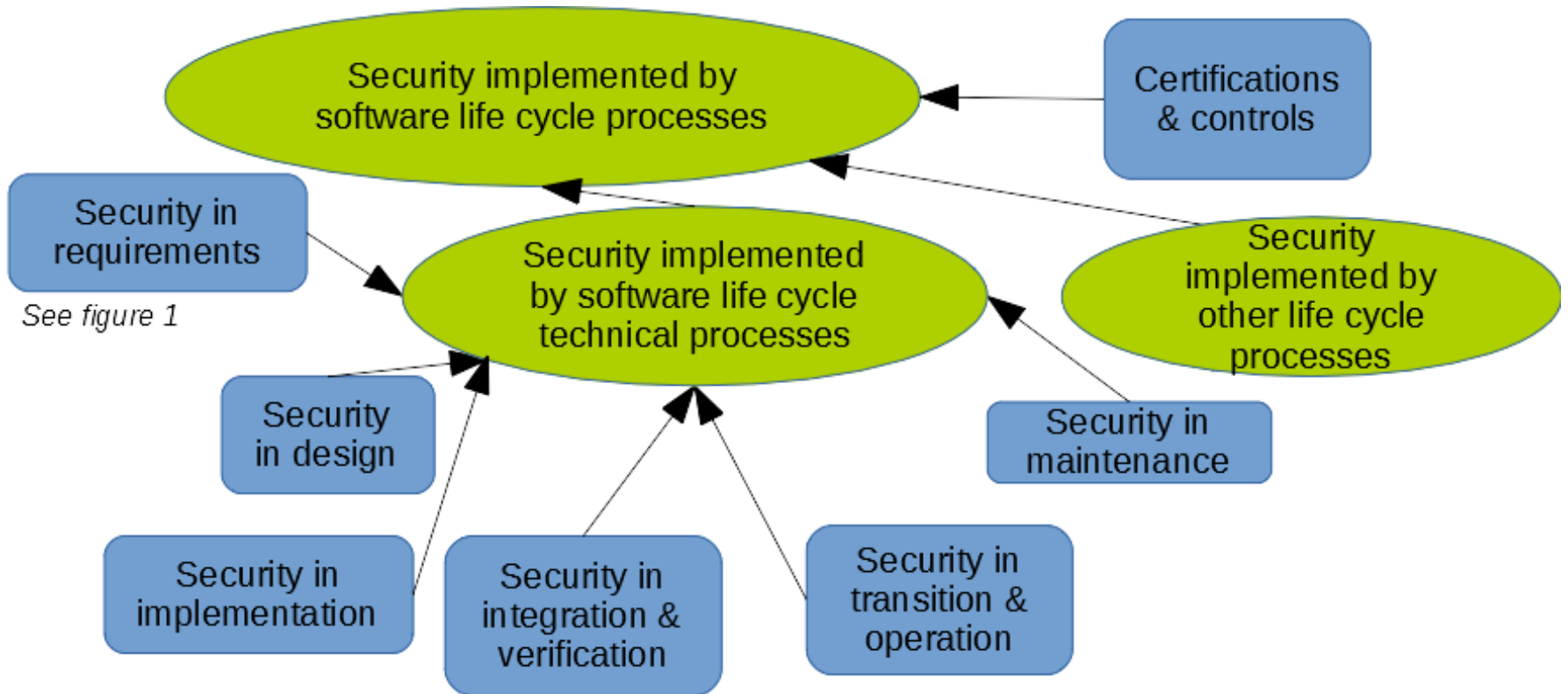
IDA | Application security: Using an assurance case

- We want applications to be generally secure
- However, security:
 - Can't be directly measured (“how many kilograms”)
 - Is an emergent property (totality of components)
 - Is often a negative property (“never does X”)
- How can you know “we’ve done enough”?
 - “Did long list of things” doesn’t provide confidence
 - How do you know those were the *right* things?
 - Must be able to justify & refine later
 - Must avoid breaking the bank
- Useful approach: an “assurance case”
 - Start with overall goal, repeatedly break into smaller parts
 - Not complicated – keeps track of what needs to be done
 - Pattern we’ve used may be useful to you too!

*See: A Sample Security Assurance Case Pattern by David A. Wheeler,
December 2018, IDA Paper P-9278*

Assurance case: Top level (figure 1)



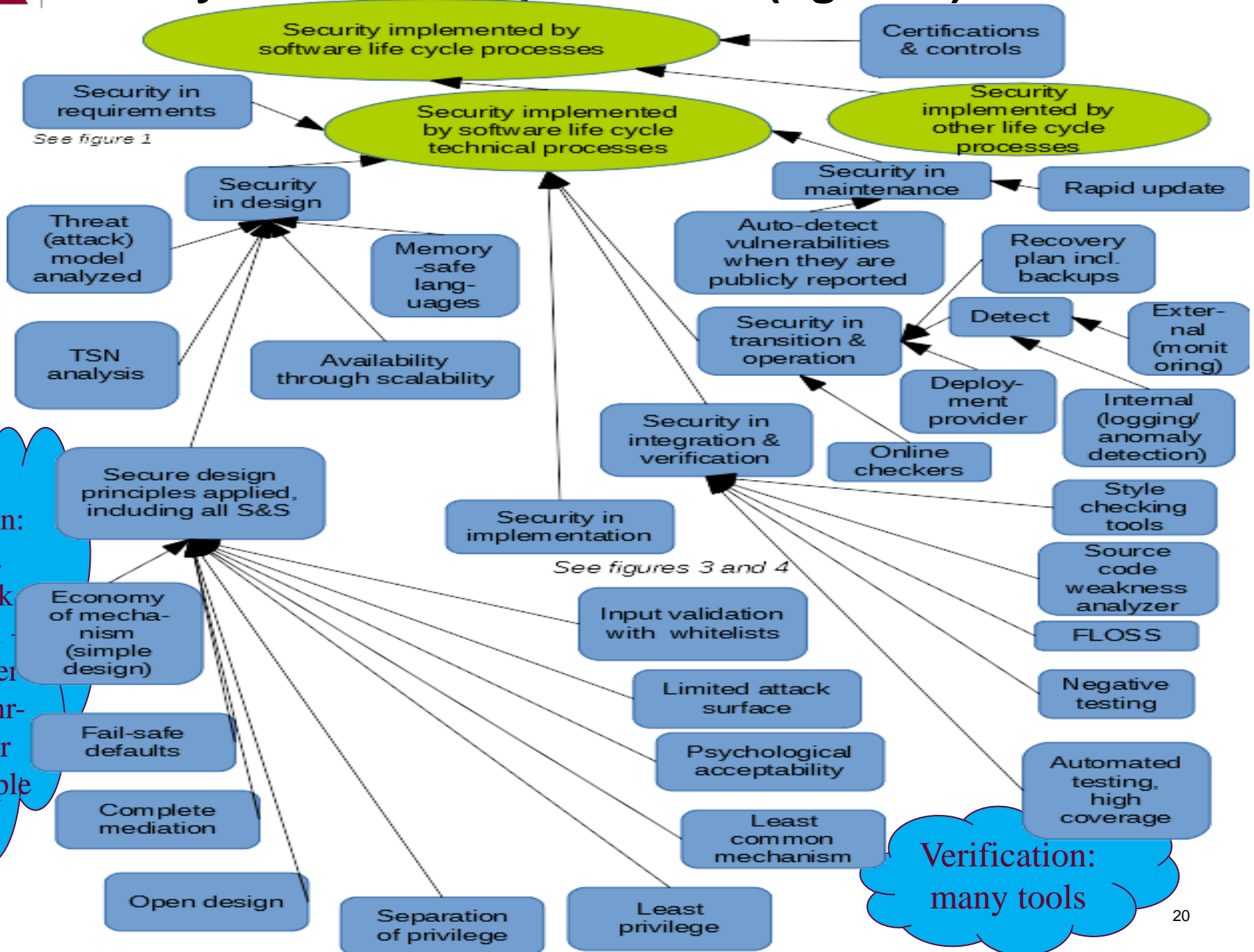


See figure 1

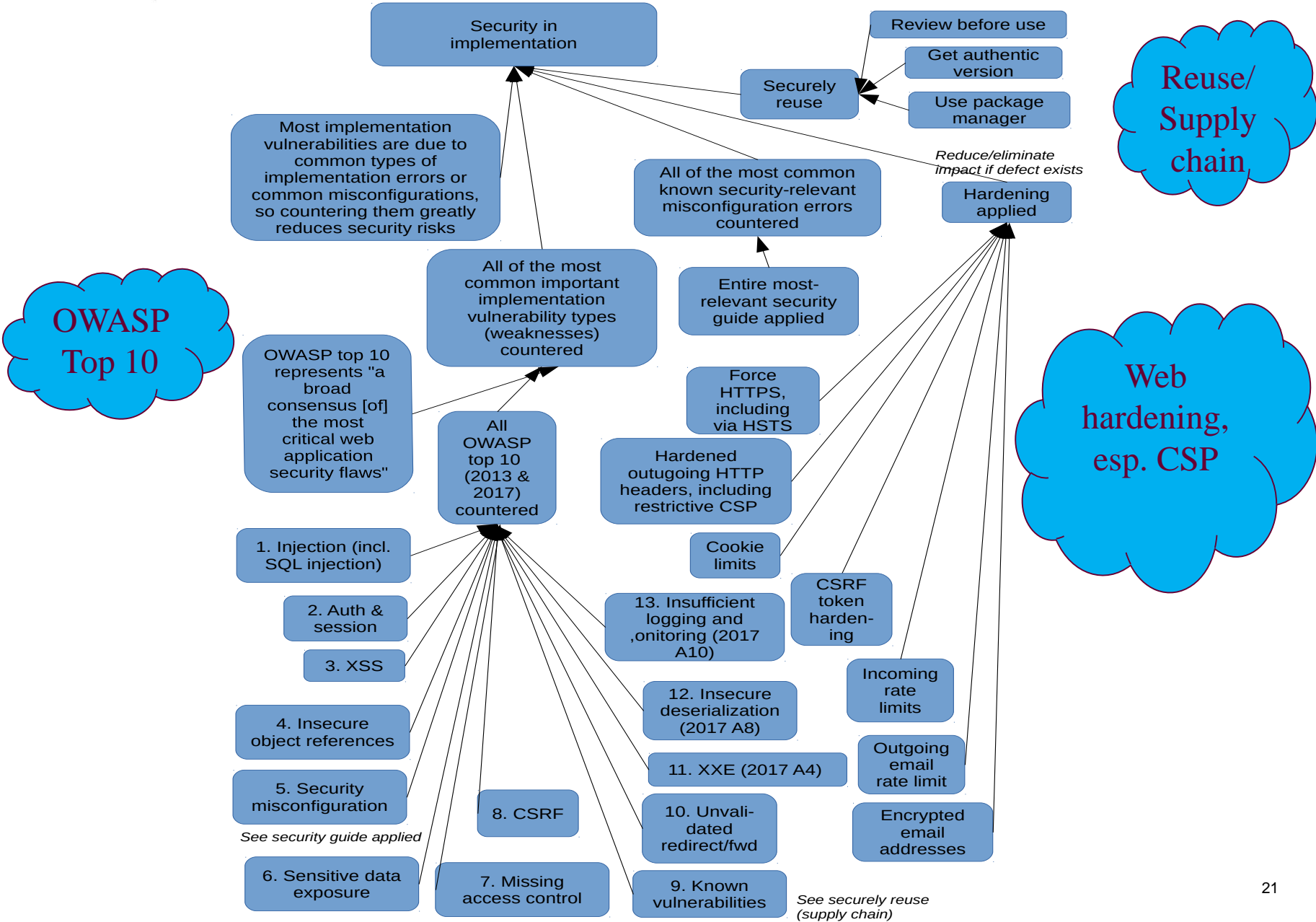


Not a waterfall-
These are
processes, not
phases

IDA | Life cycle technical processes (figure 2)



Security in implementation (figure 3)



IDA | Got on Hacker News (HN)!

- Badge-related post got on Hacker News front page on 2018-10-06
 - “Certainly not knocking on the badge or the practices...I just found it amusing that PHP often gets a bad rap, but then shows up at the top of the listed projects for objectively good development practices.” - reindeer
 - “I just found and read through the criteria list. **It's mind-bogglingly exhaustive, but in a very good way**, and an excellent catalyst for maintainable, secure software. I'd regard it as **universally applicable** to any and all code.” – exikyut
 - “Lots of self-proclaimed ‘experts’ love to say ‘do X and Y and Z and you will be successful because these are best practices’, but it's all a bunch of snake oil... ‘Best practices are best not practiced.’” – userbinator, dissenting, but then downvoted & replied to...
 - “Best practices are a bit like good genes. [They’re] by no means a guarantee of success, fame, glory and riches, but damn if they don't make things easier.” - reindeer
 - “I see absolutely nothing dogmatic or cargo cult about the recommendations they make. **They are completely sensible**, and a decent guideline for improving the technical support infrastructure of a project.” - throwaway2048

- CII best practices badge (get a badge):
 - <https://bestpractices.coreinfrastructure.org/>
- CII best practices badge project:
 - <https://github.com/coreinfrastructure/best-practices-badge>

My thanks to the *many* who reviewed or helped develop the badging criteria and/or the software to implement it. This includes: Mark Atwood, Tod Beardsley, Doug Birdwell, Alton(ius) Blom, Hanno Böck, enos-dandrea, Jason Dossett, David Drysdale, Karl Fogel, Alex Jordan (strugee), Sam Khakimov, Greg Kroah-Hartman, Dan Kohn, Charles Neill (cneill), Mark Rader, Emily Ratliff, Tom Ritter, Nicko van Someren, Daniel Stenberg (curl), Marcus Streets, Trevor Vaughan, Dale Visser, Florian Weimer

IDA | Involved in OSS?

- If you lead an OSS project, what you do matters!
 - People depend on the software you create
 - The practices you apply affect the result
 - Secure or quality software is not an accident
 - Please try to get a badge, & show when you have it
- If you're considering using an OSS project
 - Check on the project – should you use it?

- CII Best Practices badge use continues to (quietly) grow
 - 2,178 participating projects & 265 passing
 - Fewer silver & gold, but steady progress
- APIs enable many uses of its data
- Modern software is mostly third party code
 - Prepare for their inevitable vulnerabilities
- Assurance cases can help make secure software
- OSS projects: Work on getting a badge!

IDA | Sample impacts of CII badge process (1 of 2)

- OWASP ZAP (web app scanner)
 - Simon Bennetts: “[it] helped us improve ZAP quality... [it] helped us focus on [areas] that needed most improvement.”
 - Change: Significantly improved automated testing
- CommonMark (Markdown in PHP) changes:
 - TLS for the website (& links from repository to it)
 - Publishing the process for reporting vulnerabilities
- OPNFV (open network functions virtualization)
 - Change: Replaced no-longer-secure crypto algorithms
- JSON for Modern C++
 - “I really appreciate some formalized quality assurance which even hobby projects can follow.”
 - Change: Added explicit mention how to privately report errors
 - Change: Added a static analysis check to continuous integration script

IDA | Sample impacts of CII badge process (2 of 2)

- BRL-CAD
 - Probably would have taken an hour uninterrupted, getting to 100% passing was relatively easy
 - Website certificate didn't match our domain, fixed
- POCO C++ Libraries
 - "... thank you for setting up the best practices site. It was really helpful for me in assessing the status..."
 - Updated the CONTRIBUTING.md file to include a statement on reporting security issues
 - Updated the instructions for preparing a release in the Wiki to include running clang-analyzer
 - Enabled HTTPS for the project website
- GNU Make
 - HTTPS. Convinced Savannah to support HTTPS for repositories (it supported HTTPS for project home pages)

- BadgeApp
 - BadgeApp is the web application that allows developers to provide information about their project and (hopefully) get a Core Infrastructure Initiative (CII)...
- Zephyr Project
 - The Zephyr Project is a small, scalable real-time operating system for use on resource-constrained systems supporting multiple architectures. Developers are...
- league/commonmark
 - Markdown parser for PHP based on the CommonMark spec.



- Criteria
 - #1 The project **MUST** have evidence that such tests are being added in the most recent major changes to the project. [tests_are_added]
 - #4 The project **MUST** have a general policy (formal or not) that as major new functionality is added, tests of that functionality **SHOULD** be added to an automated test suite. [test_policy]
- Automated testing is important
 - Quality, supports rapid change, supports updating dependencies when vulnerability found
 - No coverage level required – just get started



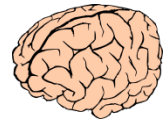
- Criteria
 - #2 “The project **MUST** publish the process for reporting vulnerabilities on the project site.” [vulnerability_report_process]
 - #8 “If private vulnerability reports are supported, the project **MUST** include how to send the information in a way that is kept private.” [vulnerability_report_private]
- Just tell people how to report!
 - In principle *easy* to do – but often omitted
 - Projects need to *decide* how



- #3 “The project sites (website, repository, and download URLs) MUST support HTTPS using TLS.” [sites_https]
- Details:
 - You can get free certificates from Let's Encrypt.
 - Projects MAY implement this criterion using (for example) GitHub pages, GitLab pages, or SourceForge project pages.
 - If you are using GitHub pages with custom domains, you MAY use a content delivery network (CDN) as a proxy to support HTTPS.
- We’ve been encouraging hosting systems to support HTTPS



- #5 “At least one static code analysis tool **MUST** be applied to any proposed major production release of the software before its release, if there is at least one FLOSS tool that implements this criterion in the selected language.” [static_analysis]
 - A static code analysis tool examines the software code (as source code, intermediate code, or executable) without executing it with specific inputs.
- #6 “All medium and high severity exploitable vulnerabilities discovered with dynamic code analysis **MUST** be fixed in a timely way after they are confirmed.” [dynamic_analysis_fixed]
 - Early versions didn’t allow “N/A”; this has been fixed.



- Criteria
 - #8 “The project **MUST** have at least one primary developer who knows how to design secure software.” [know_secure_design]
 - #9 “At least one of the primary developers **MUST** know of common kinds of errors that lead to vulnerabilities in this kind of software, as well as at least one method to counter or mitigate each of them.” [know_common_errors]
- Specific list of requirements given – doesn’t require “know everything”
- Perhaps need short “intro” course material?



- #10 “The project MUST include reference documentation that describes its external interface (both input and output).”
[documentation_interface]
- Some OSS projects have good documentation – but some do not

- The project **MUST** clearly define and document its project governance model (the way it makes decisions, including key roles). [governance]
- The project **MUST** be able to continue with minimal interruption if any one person is incapacitated or killed... [you] **MAY** do this by providing keys in a lockbox and a will providing any needed legal rights (e.g., for DNS names). [access_continuity]
- The project **MUST** have FLOSS automated test suite(s) that provide at least 80% statement coverage if there is at least one FLOSS tool that can measure this criterion in the selected language. [test_statement_coverage80]
- The project **MUST** automatically enforce its selected coding style(s) if there is at least one FLOSS tool that can do so in the selected language(s). [coding_standards_enforced]
- The project **MUST** implement secure design principles (from "know_secure_design"), where applicable... [implement_secure_design]

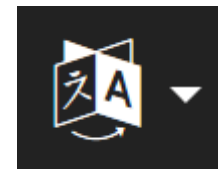
- The project results **MUST** check all inputs from potentially untrusted sources to ensure they are valid (a whitelist), and reject invalid inputs, if there are any restrictions on the data at all. [input_validation]
- The project **MUST** cryptographically sign releases of the project results intended for widespread use, and there **MUST** be a documented process explaining [how to] obtain the public signing keys and verify the signature(s)... [signed_releases]
- The project **MUST** provide an assurance case that justifies why its security requirements are met. [It **MUST**...] [assurance_case]
- The project **MUST** use at least one static analysis tool ... to look for common vulnerabilities... , if there is at least one FLOSS tool that can... [static_analysis_common_vulnerabilities]
- Projects **MUST** monitor or periodically check their external dependencies (including convenience copies) to detect known vulnerabilities, and fix exploitable vulnerabilities or verify them as unexploitable. [dependency_monitoring]

- The project **MUST** require two-factor authentication (2FA) for developers for changing a central repository or accessing sensitive data (such as private vulnerability reports)... [require_2FA]
- The project **MUST** have at least 50% of all proposed modifications reviewed before release by a person other than the author... [two_person_review]
- The project **MUST** have a "bus factor" of 2 or more. [bus_factor]
- The project **MUST** have a reproducible build... [build_reproducible]
- The project **MUST** apply at least one dynamic analysis tool to any proposed major production release of the software before its release. [dynamic_analysis]
- The project **MUST** have performed a security review within the last 5 years. This review **MUST** consider the security requirements and security boundary. [security_review]
- Hardening mechanisms **MUST** be used in the software produced by the project so that software defects are less likely to result in security vulnerabilities. [hardening]

Level	Total active	MUST	SHOULD	SUGGESTED	Allow N/A	Met justification or URL required	Includes details	New at this level
Passing	66	42	10	14	27	9	48	66
Silver	55	44	10	1	39	54	38	48
Gold	23	21	2	0	9	21	15	14

There are not a *lot* of gold criteria, but they're challenging.

Source: <https://bestpractices.coreinfrastructure.org/criteria>
as of 2017-09-10



- English (en)
- Chinese (Simplified) / 简体中文 (zh-CN)
- French / Français (fr)
- German / Deutsch (de)
- Japanese / 日本語 (ja)
- Russian / Русский (ru)

Our sincere
thanks to all
the hard-working
translators!!

Even if you can't understand the detailed justifications,
you can see the criteria & claimed answers

- OSS: software licensed to users with these freedoms:
 - to *run* the program for any purpose,
 - to *study* and *modify* the program, and
 - to freely *redistribute* copies of either the original or modified program (without royalties to original author, etc.)
- Original term: “Free software” (confused with no-price)
- Other synonyms: libre sw, free-libre sw, FOSS, FLOSS
- Antonyms: proprietary software, closed software
- Widely used; OSS #1 or #2 in many markets
 - “... plays a more critical role in the DoD than has generally been recognized.” [MITRE 2003]
- OSS almost always *commercial* by law & regulation
 - Software licensed to general public & has non-government use
→ commercial software (in US law, per 41 USC 403)

IDA | Criteria categories and examples (1)

1. Basics

- The software **MUST** be released as FLOSS*. [floss_license]
- It is **SUGGESTED** that any required license(s) be approved by the Open Source Initiative (OSI). [floss_license_osi]

2. Change Control

- The project **MUST** have a version-controlled source repository that is publicly readable and has a URL. [repo_public]
 - Details: The URL **MAY** be the same as the project URL. The project **MAY** use private (non-public) branches in specific cases while the change is not publicly released (e.g., for fixing a vulnerability before it is revealed to the public).

3. Reporting

- The project **MUST** publish the process for reporting vulnerabilities on the project site. [vulnerability_report_process]

4. Quality

- If the software requires building for use, the project **MUST** provide a working build system that can automatically rebuild the software from source code. [build]
- The project **MUST** have at least one automated test suite that is publicly released as FLOSS (this test suite may be maintained as a separate FLOSS project). [test]
- The project **MUST** have a general policy (formal or not) that as major new functionality is added, tests of that functionality **SHOULD** be added to an automated test suite. [test_policy]
- The project **MUST** enable one or more compiler warning flags, a "safe" language mode, or use a separate "linter" tool to look for code quality errors or common simple mistakes, if there is at least one FLOSS tool that can implement this criterion in the selected language. [warnings]

5. Security

- At least one of the primary developers **MUST** know of common kinds of errors that lead to vulnerabilities in this kind of software, as well as at least one method to counter or mitigate each of them. [know_common_errors]
- The project's cryptographic software **MUST** use only cryptographic protocols and algorithms that are publicly published and reviewed by experts. [crypto_published]
- The project **MUST** use a delivery mechanism that counters MITM attacks. Using https or ssh+scp is acceptable. [delivery_mitm]
- There **MUST** be no unpatched vulnerabilities of medium or high severity that have been publicly known for more than 60 days. [vulnerabilities_fixed_60_days]

6. Analysis

- At least one static code analysis tool **MUST** be applied to any proposed major production release of the software before its release, if there is at least one FLOSS tool that implements this criterion in the selected language... [static_analysis]
- It is **SUGGESTED** that the {static code analysis} tool include rules or approaches to look for common vulnerabilities in the analyzed language or environment.
[static_analysis_common_vulnerabilities]
- It is **SUGGESTED** that at least one dynamic analysis tool be applied to any proposed major production release of the software before its release. [dynamic_analysis]

IDA | Badge criteria must NOT be...

- Will NOT require any specific products or services (especially proprietary ones)
 - We intentionally don't require git or GitHub
 - That said, will automate many things if project *does* use GitHub
- Will NOT require or forbid any particular programming language