

Core Infrastructure Initiative (CII) Best Practices Badge: One Year Later

Dr. David A. Wheeler

2017-02-08

dwheeler @ ida.org

Personal: dwheeler @ dwheeler.com,

Twitter: drdavidawheeler

GitHub & GitLab: david-a-wheeler

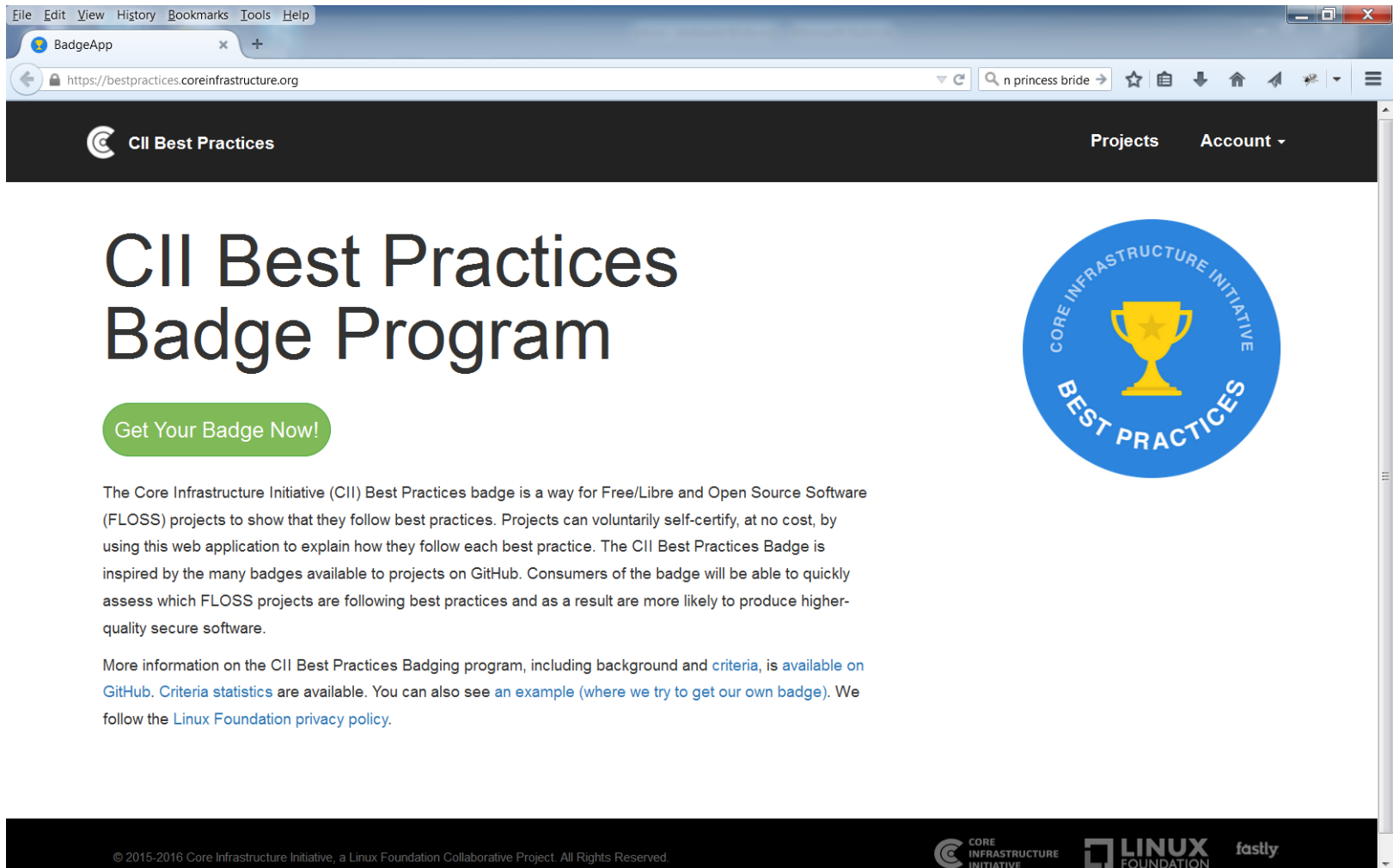
<https://www.dwheeler.com>



- 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”



- 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
 - Best practices is for OSS projects (*production* side)
 - 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



The screenshot shows a web browser window with the address bar displaying <https://bestpractices.coreinfrastructure.org>. The page features a dark header with the CII logo and the text "CII Best Practices" on the left, and "Projects" and "Account" on the right. The main content area has a large heading "CII Best Practices Badge Program" and a green button labeled "Get Your Badge Now!". To the right of the heading is a circular blue badge with a yellow trophy icon and the text "CORE INFRASTRUCTURE INITIATIVE" and "BEST PRACTICES". Below the heading, there is a paragraph explaining the program and a link to the criteria. The footer contains copyright information and logos for the Core Infrastructure Initiative, Linux Foundation, and fastly.

File Edit View History Bookmarks Tools Help

BadgeApp

<https://bestpractices.coreinfrastructure.org>

n princess bride

CII Best Practices

Projects Account

CII Best Practices Badge Program

Get Your Badge Now!

The Core Infrastructure Initiative (CII) Best Practices badge is a way for Free/Libre and Open Source Software (FLOSS) projects to show that they follow best practices. Projects can voluntarily self-certify, at no cost, by using this web application to explain how they follow each best practice. The CII Best Practices Badge is inspired by the many badges available to projects on GitHub. Consumers of the badge will be able to quickly assess which FLOSS projects are following best practices and as a result are more likely to produce higher-quality secure software.

More information on the CII Best Practices Badging program, including background and [criteria](#), is [available on GitHub](#). [Criteria statistics](#) are available. You can also see [an example \(where we try to get our own badge\)](#). We follow the [Linux Foundation privacy policy](#).

© 2015-2016 Core Infrastructure Initiative, a Linux Foundation Collaborative Project. All Rights Reserved.

CORE INFRASTRUCTURE INITIATIVE

LINUX FOUNDATION

fastly

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

- Currently one 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

Source:

<https://github.com/linuxfoundation/cii-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)

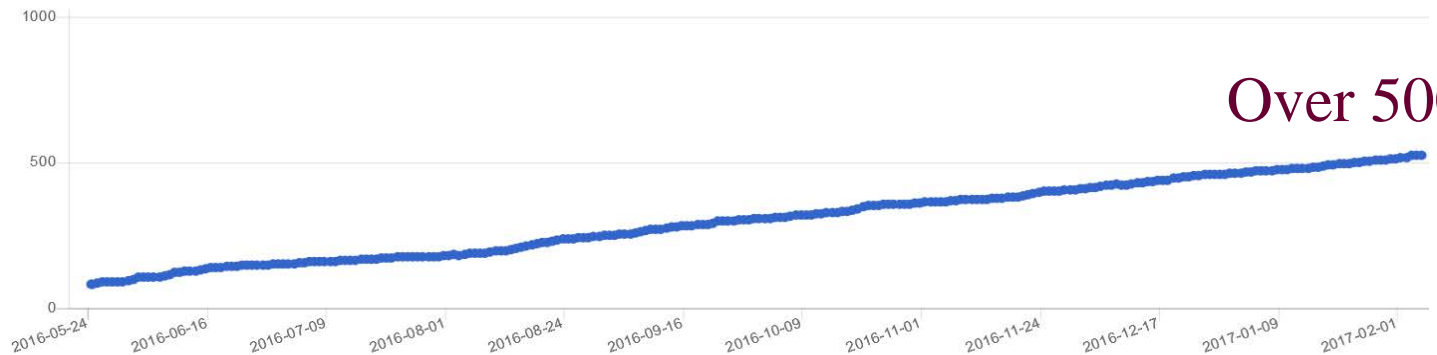
IDA | Initial announcement

- General availability announced May 2016
- Early badge holders:
 - BadgeApp (itself!)
 - Node.js
 - Linux kernel
 - curl
 - GitLab
 - OpenSSL (pre-Heartbleed missed 1/3 criteria)
 - Zephyr project

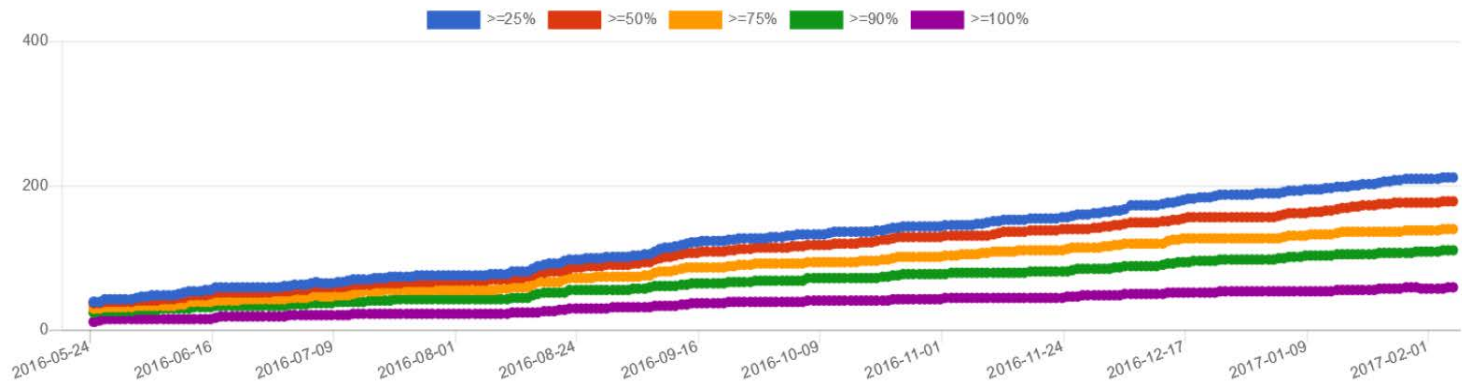
IDA | CII badges are getting adopted!

All
projects

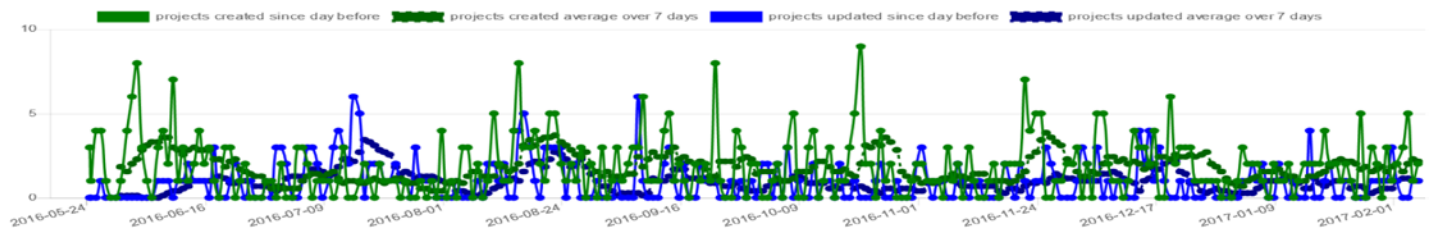
Over 500!



Projects
with non-
trivial
progress



Daily
activity



Source: https://bestpractices.coreinfrastructure.org/project_stats
as of 2017-02-06

IDA | Some additional badge holders

- CommonMark (Markdown in PHP)
- Apache Libcloud
- Apache Syncope
- GnuPG
- phpMyAdmin
- pkgsrc
- openstack
- OWASP ZAP (web app scanner)
- OPNFV (open network functions virtualization)
- JSON for Modern C++
- NTPsec
- LibreOffice
- OpenUnison
- sql-server-base
- Blender
- dpkg
- libseccomp

60 “passing” badges as of 2017-02-08


IDA | Sample impacts of CII badge process


- 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 | Biggest challenges today for getting a badge


- Looked at all projects 90%+ but not passing
 - 52 projects. MUST with Unmet or “?” => Top 10 challenges:


#	Criterion	%missed
1	tests_are_added	25%
2	vulnerability_report_process	23%
3	sites_https	17%
4	test_policy	15%
5	static_analysis	15%
6	dynamic_analysis_fixed	15%
7	vulnerability_report_private	13%
8	know_common_errors	12%
9	know_secure_design	10%
10	documentation_interface	8%


 Tests

 Analysis

 Know secure development

 Vulnerability reporting

 HTTPS

 Documentation

This data is as of 2017-02-06 12:20ET

Changing to 75%+ (81 projects) top 10 list has a slightly different order but the set is the same, except that 75%+ adds warnings_fixed as its #10 & know_common_errors moves #8→#11



■ 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



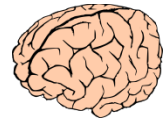
- Criteria
 - #2 “The project MUST publish the process for reporting vulnerabilities on the project site.” [vulnerability_report_process]
 - #8 “If private vulnerability reports are supported, the project MUST include how to send the information in a way that is kept private.” [vulnerability_report_private]
- 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

- Many criteria are widely met, e.g.:
 - Use of version control - repo_track
 - Process for submitting bug reports - report_process
 - No unpatched vulnerabilities of medium or high severity publicly known for more than 60 days - vulnerabilities_fixed_60_days

- Have developed draft criteria for higher-level badges
 - Current names: “passing+1” and “passing+2”
 - Passing+2 expected to be harder and *not* necessarily achievable by single-person projects
 - Merged from proposals, NYC 2016 brainstorm, OW2, Apache maturity model
 - Expect to drop/add criteria due to feedback
- **ANNOUNCING:** It’s available for feedback:
 - <https://github.com/linuxfoundation/cii-best-practices-badge/blob/master/doc/other.md>
- We’d love your feedback!

IDA | Some proposed passing+1 criteria

- 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... Individuals who run a FLOSS project **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 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]
- Project releases of the software intended for widespread use **MUST** be cryptographically signed... [signed_releases]
- 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]

IDA | Some proposed passing+2 criteria

- 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. [security_review]



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?
- We'd love your help in improving criteria

IDA | In conclusion: Key URLs

- CII
 - <https://www.coreinfrastructure.org>
- CII best practices badge (get a badge):
 - <https://bestpractices.coreinfrastructure.org/>
- Draft passing+1 & passing+2 criteria
 - <https://github.com/linuxfoundation/cii-best-practices-badge/blob/master/doc/other.md>
- CII best practices badge project:
 - <https://github.com/linuxfoundation/cii-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

- 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)

- Multi-million dollar project
 - Supported by many, e.g., Amazon Web Services, Adobe, Bloomberg, Cisco, Dell, Facebook, Fujitsu, Google, Hitachi, HP, Huawei, IBM, Intel, Microsoft, NetApp, NEC, Qualcomm, RackSpace, salesforce.com, and VMware
- Actions
 - Collaboratively identifies & funds OSS projects in need of assistance
 - Allows developers to continue their work under community norms
 - Transitioning from point fixes to holistic solutions for open source security

IDA | CII-funded projects with multi-project impacts

- The fuzzing project
- OWASP Zed Attack Proxy (ZAP) as a service
- False-Positive-Free Testing with Frama-C
- Reproducible builds
- CII census (project quantitative analysis)
- Best practices badge (focus today)



IDA | Mozilla Open Source Support (MOSS) relation

- Mozilla Open Source Support (MOSS) added Secure Open Source (SOS) track
 - Announced June 9, 2016
 - “supports security audits for open source software projects, and remedial work to rectify the problems found”
 - “support model is different from & complementary to CII. [CII focuses on] deeper-dive investments into core OS security infrastructure, while [SOS targets] OSS projects with lower-hanging fruit security needs.”
- CII complements other efforts like MOSS

IDA | Badge criteria must be...

- Relevant
- Attainable by typical OSS projects
- Clear
- Include security-related criteria
- Consensus of developers & users
 - Criteria & web app developed as OSS project
 - Built on existing work, e.g., Karl Fogel's *Producing Open Source Software*
- Not hypocritical
 - Our web app must get its own badge!

Worked with several projects, including the
Linux kernel & curl, to perform alpha test of criteria

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]

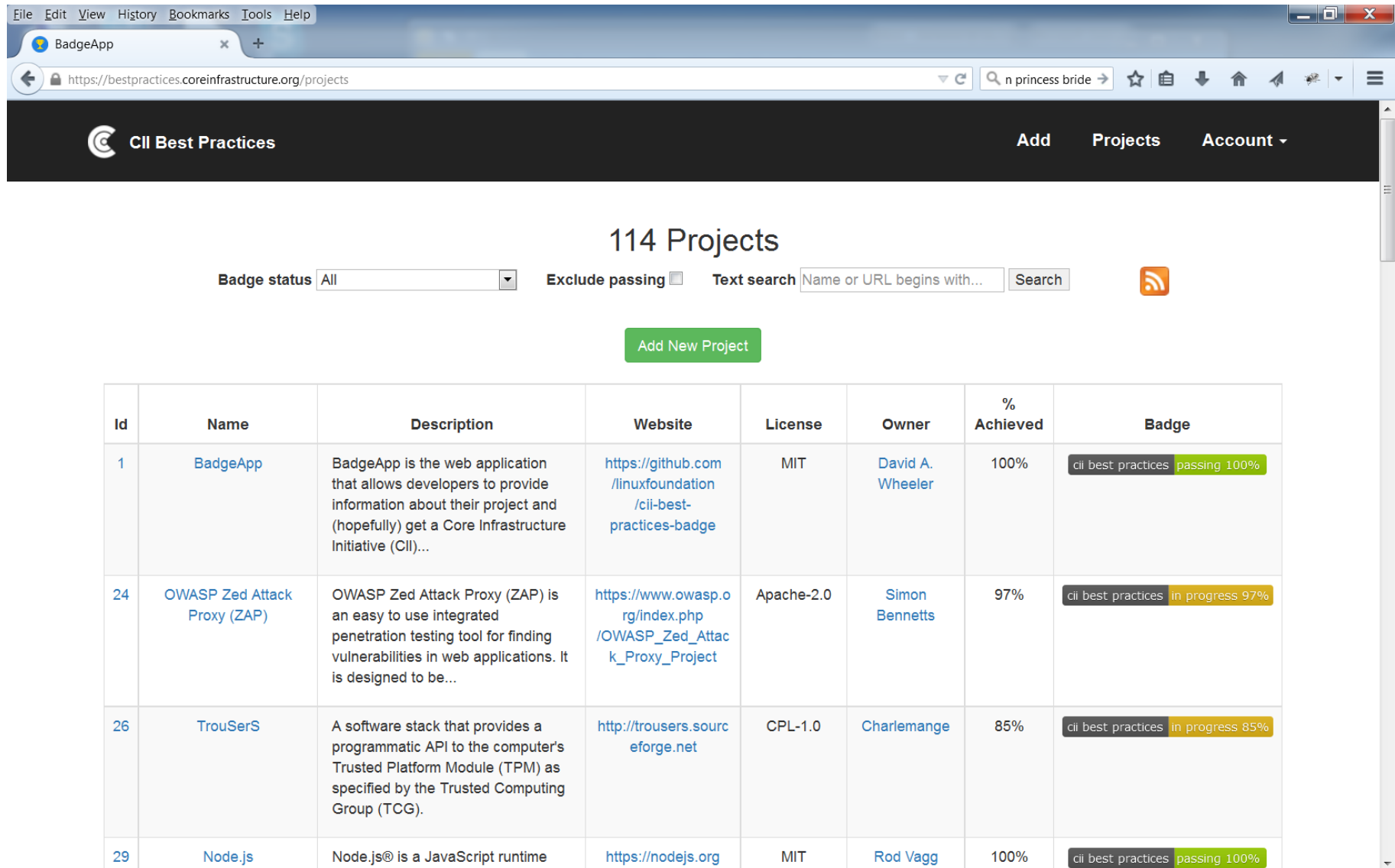
IDA | Badge criteria must NOT be...

- 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





- Criteria have different levels of importance
 - MUST (NOT) – required (42/66)
 - SHOULD (NOT) – sometimes valid to not do (10/66)
 - SUGGESTED – common valid reasons, but at least consider it (14/66)
- Criteria may have “details” (39/66)
 - Give clarifications/examples, e.g., “MAY...”
- Each criterion is named (lowercase underscore)
- For each criterion, users answer:
 - Status: Met, Unmet, Unknown (?), N/A*
 - Justification: Markdown text. Usually optional

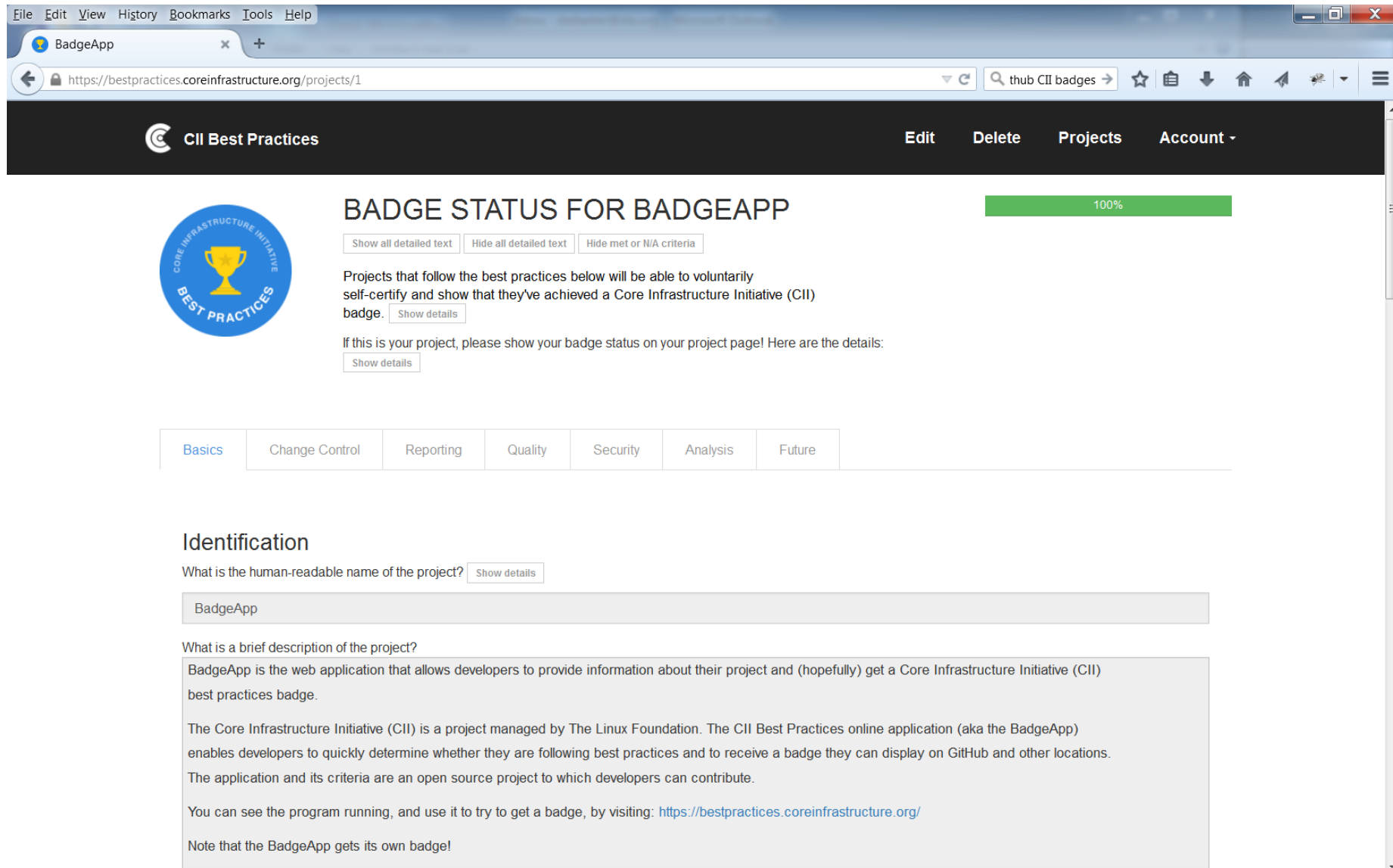
IDA | BadgeApp security

- File “security.md” describes how we secure the web app
- Report vulnerabilities to designated people
- Requirements – simple, most data public
 - Passwords stored in database only as iterated salted hashes
- Design: Showed that we applied design principles
 - Simple, filter inputs
- Implementation
 - Checked that it counters all of OWASP top 10
 - Applied “Ruby on Rails Security Guide”
 - Hardened (e.g., hardening HTTP headers)
- Verification
 - Source code quality analyzer (rubocop, rails_best_practices), [static] source code weakness analyzer (brakeman), web application scanner (OWASP ZAP), 98% test coverage, OSS enables multi-person review
- Supply chain (reuse)
 - Consider before use, bundle-audit (check known vulns), license_finder
 - Strive to minimize/simplify transitive dependencies & size
- People



The screenshot shows a web browser window with the address bar displaying `https://bestpractices.coreinfrastructure.org/projects`. The page header features the "CII Best Practices" logo and navigation links for "Add", "Projects", and "Account". The main content area displays "114 Projects" and includes filters for "Badge status" (set to "All"), "Exclude passing" (unchecked), and a "Text search" field. A green "Add New Project" button is also visible. Below these elements is a table listing four projects: BadgeApp, OWASP Zed Attack Proxy (ZAP), TrouSerS, and Node.js. Each row in the table provides details such as the project ID, name, description, website, license, owner, and achievement percentage, along with a visual badge status indicator.

Id	Name	Description	Website	License	Owner	% Achieved	Badge
1	BadgeApp	BadgeApp is the web application that allows developers to provide information about their project and (hopefully) get a Core Infrastructure Initiative (CII)...	https://github.com/linuxfoundation/cii-best-practices-badge	MIT	David A. Wheeler	100%	
24	OWASP Zed Attack Proxy (ZAP)	OWASP Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be...	https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project	Apache-2.0	Simon Bennetts	97%	
26	TrouSerS	A software stack that provides a programmatic API to the computer's Trusted Platform Module (TPM) as specified by the Trusted Computing Group (TCG).	http://trousers.sourceforge.net	CPL-1.0	Charlemagne	85%	
29	Node.js	Node.js® is a JavaScript runtime	https://nodejs.org	MIT	Rod Vagg	100%	



The screenshot shows a web browser window with the address bar displaying `https://bestpractices.coreinfrastructure.org/projects/1`. The page title is "BadgeApp". The main header of the application is "CII Best Practices" with navigation links for "Edit", "Delete", "Projects", and "Account".

The main content area is titled "BADGE STATUS FOR BADGEAPP" and features a green progress bar indicating 100% completion. Below the title, there are three buttons: "Show all detailed text", "Hide all detailed text", and "Hide met or N/A criteria".

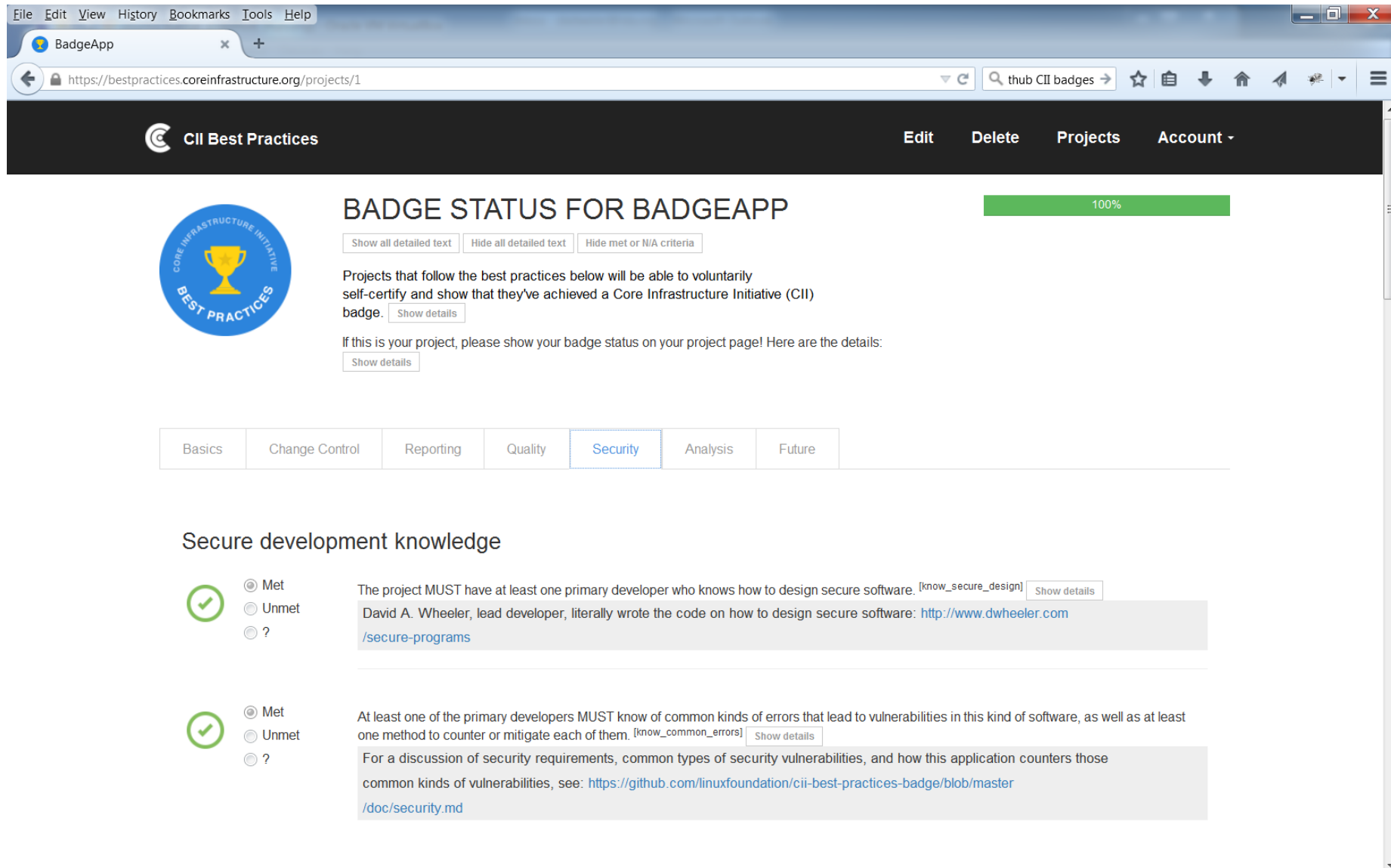
The text explains that projects following best practices can voluntarily self-certify and show they've achieved a Core Infrastructure Initiative (CII) badge. A "Show details" button is provided.

Below this, a message states: "If this is your project, please show your badge status on your project page! Here are the details:" followed by another "Show details" button.

A horizontal navigation bar contains the following tabs: "Basics", "Change Control", "Reporting", "Quality", "Security", "Analysis", and "Future". The "Basics" tab is currently selected.

The "Identification" section is titled "Identification" and contains the following information:

- What is the human-readable name of the project? [Show details](#)
BadgeApp
- What is a brief description of the project?
BadgeApp is the web application that allows developers to provide information about their project and (hopefully) get a Core Infrastructure Initiative (CII) best practices badge.
- The Core Infrastructure Initiative (CII) is a project managed by The Linux Foundation. The CII Best Practices online application (aka the BadgeApp) enables developers to quickly determine whether they are following best practices and to receive a badge they can display on GitHub and other locations. The application and its criteria are an open source project to which developers can contribute.
- You can see the program running, and use it to try to get a badge, by visiting: <https://bestpractices.coreinfrastructure.org/>
- Note that the BadgeApp gets its own badge!



File Edit View History Bookmarks Tools Help


BadgeApp

https://bestpractices.coreinfrastructure.org/projects/1

thub CII badges

CII Best Practices

Edit Delete Projects Account -

 **BADGE STATUS FOR BADGEAPP** 100%

Show all detailed text Hide all detailed text Hide met or N/A criteria

Projects that follow the best practices below will be able to voluntarily self-certify and show that they've achieved a Core Infrastructure Initiative (CII) badge. [Show details](#)

If this is your project, please show your badge status on your project page! Here are the details: [Show details](#)

Basics Change Control Reporting Quality **Security** Analysis Future

Secure development knowledge

☒ Met ☐ Unmet ☐ ?

The project **MUST** have at least one primary developer who knows how to design secure software. [\[know_secure_design\]](#) [Show details](#)

David A. Wheeler, lead developer, literally wrote the code on how to design secure software: <http://www.dwheeler.com/secure-programs>

☒ Met ☐ Unmet ☐ ?

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. [\[know_common_errors\]](#) [Show details](#)

For a discussion of security requirements, common types of security vulnerabilities, and how this application counters those common kinds of vulnerabilities, see: <https://github.com/linuxfoundation/cii-best-practices-badge/blob/master/doc/security.md>

IDA | EU-FOSSA project interactions with CII Badge

- EU-FOSSA = EU-Free and Open Source Software Auditing
 - 1M Euro project initiated by 2 Members of European Parliament
 - Executed by European Commission (the European Union's executive body)
 - Goal: invest into commonly used OSS which might need support in the security domain
- Intends to define a complete process to properly perform code reviews within the European Institutions
 - To execute one sample code review during Q3-Q4/2016
 - Sample results will determine if activity could become a continuous action of the European Institutions in the future
- FOSSA project exchanging experiences with CII
- FOSSA looking closely at the CII Badge criteria
 - During definition of metrics to analyze sustainability and security

See: <https://joinup.ec.europa.eu/community/eu-fossa/description> and <https://fosdem.org/2016/schedule/event/fossa/>

IDA | A few notes on the BadgeApp

- “BadgeApp” is simple web application that implements the criteria (fill in form)
 - OSS (MIT license)
 - All libraries OSS & legal to add (checked with license_finder)
 - Simple Ruby on Rails app
 - Criteria info (text, category, etc.) in YAML
- Overall approach: Proactively counter mistakes
 - Mistakes happen; we use a variety of tools, automated test suite, processes to counter them
- Please contribute!
 - See its CONTRIBUTING.md for more