



Sandia
National
Laboratories

Application of the OpenSSF Best Practices Badge Program to Scientific Software

Roscoe A. Bartlett, PhD (Sandia National Labs)
Yanfei Guo (Argonne National Laboratory)
Pratik Nayak (Technical University of Munich)

September 24, 2025

HPC Best Practices Webinar Series



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2023-04819O

OpenSSF Best Practices Badge Program: BSSW.io Article



Information For ▾ Contribute to BSSw Receive Our Email Digest Contact BSSw

better scientific software

Resources ▾ Blog Events About ▾ 🔎

HOME > RESOURCES > OpenSSF Best Practices Badge...

in f t o

OpenSSF Best Practices Badge Program

MAY 05, 2023 AUTHOR ROSCOE A. BARTLETT

The Linux Foundation's OpenSSF Best Practices Badge Program represents an impressive collection of the open source community's knowledge base for creating, maintaining, and sustaining robust, high quality, and (most importantly) secure open source software. At its foundation is a featureful "Badge App" website, which provides a database of projects that document what best practices they have adopted and supporting evidence. This set of best practices (along with the detailed documentation and supporting justifications for each item) also serves as an incremental learning tool and as a foundation for incremental software process and quality improvements efforts.

TOPICS BETTER DEVELOPMENT ↗ REVISION CONTROL
↗ DEVELOPMENT TOOLS

Resources by Topic

- + Better Planning
- + Better Development
- + Better Performance
- + Better Reliability
- + Better Collaboration
- + Better Skills

> View All Resources

<https://bssw.io/items/openssf-best-practices-badge-program>

Security Issues in Open Source Software



DR Dark Reading

Open Source Vulnerabilities Still Pose a Big Challenge for Security Teams

Open source software continues to pose a challenge for companies. With the proper security practices, you can reduce your open source risk...

Mar 23, 2023



W The New Stack

Why So Much Open Source Software Is Vulnerable to Hackers

A recent Open Source Security and Risk Analysis (OSSRA) study indicates that 84% of codebases contained at least one known open source...

Mar 9, 2023



LF Lawfare Blog

Open-Source Security: How Digital Infrastructure Is Built on a House of Cards

Log4Shell remains a national concern because the open-source community cannot continue to shoulder the responsibility of securing this...

Jul 25, 2022



CSO CSO Online

The Heartbleed bug: How a flaw in OpenSSL caused a security crisis

Heartbleed can be traced to a single line of code in OpenSSL, an open source code library. Here's how Heartbleed works and how to fix it.

Sep 6, 2022



How does software security impact Computational Science & Engineering (CSE) / High Performance Computing (HPC) Communities?

- Incautious usage of software systems can create problems for our institutions.
- Software installs can create vulnerabilities on our systems and customer systems.
- Some of our CSE/HPC software may be run in environments that can open up security vulnerabilities (e.g., **AI agents!**).
- Workflows, pipelines, and containers increasingly used for CSE/HPC bring with them possible security vulnerabilities.

NOTE: OpenSSL Heartbleed bug was exposed in 2014 and is still generating news articles 11 years later!

Open Source Security Foundation (OpenSSF)



Open Source Security Foundation (OpenSSF)

- A Linux Foundation Project
- Cross-industry organization
- Bringing together the industries open source security initiatives & individuals
- Vision: *“... a future where participants in the open source ecosystem use and share high quality software, with security handled proactively, by default, and as a matter of course”*
- Technical vision:
 - Developers learn secure practices, guided by tools.
 - Security policies created, distributed, enforced automatically.
 - Security issues identified, flow back to chain for rapid response.
 - Community members provide info, notifications, flow forward to users.

Source: <https://openssf.org/about/>



OpenSSF Members - Premier (22)
[Provide funding for OpenSSF](#)



OpenSSF Best Practices Badge Program: Overview



- Set of best practices curated from open-source community that have **specific actionable criteria which require supporting evidence**,
- Particularly strong focus on software security which addresses several U.S. Federal Government notices on software security,

NOTE: Most of the OpenSSF best practices are applicable to all software, not just security-critical software!

- Featureful "Badge App" site that enhances the display of each practice, expanded descriptions of the practices, and fields to enter URL and text descriptions of the status of each practice in the project,
- Badge that can be displayed on a project's own hosting site to show that a project follows accepted best practices,
- Learning tool for best practices for developers and projects,
- Roadmap for continual improvement for a project as it incrementally adopts more practices and improves its scores in different areas,
- Standard index into the parts of the projects and how it handles different types of processes, and
- Website template and database implementation that can be forked and customized for more targeted communities. => **Has been utilized at Sandia National labs to create a customized site!**



OpenSSF Best Practices Badge Program Overview

OpenSSF Best Practices Structure



The OpenSSF Best Practices are broken down and organized in several different ways:

- **Required or optional practices:**
 - MUST: Required/not optional (unless 'N/A' is allowed)
 - SHOULD: Required unless a strong argument against can be made
 - SUGGESTED: Not required but suggested
- **Three different badge levels:**
 - Passing: 43 MUST, 10 SHOULD, 14 SUGGESTED
 - Silver: +44 MUST, +10 SHOULD, +1 SUGGESTED
 - Gold: +21 MUST, +2 SHOULD
- **Six different categories in each badge level:**
 - Basic
 - Change Control
 - Reporting
 - Quality
 - Security
 - Analysis
- **Each category broken down into subcategories**

Passing

Basics

Basic project website content

- The project website **MUST** provide information on how to: obtain, provide feedback (as bug reports or enhancements), and contribute to the software. [\[interact\]](#)
- The information on how to contribute **SHOULD** include the requirements for acceptable contributions (...). **{Met URL}** [\[contribution_requirements\]](#)

NOTE: Each practice has a unique [\[short_link_name\]](#)



Basics

Basic project website content

- The project website MUST succinctly describe what the software does (what problem does it solve?). [\[description_good\]](#)
- The project website MUST provide information on how to: obtain, provide feedback (as bug reports or enhancements), and contribute to the software. [\[interact\]](#)
- The information on how to contribute MUST explain the contribution process (e.g., are pull requests used?) {Met URL} [\[contribution\]](#)
- The information on how to contribute SHOULD include the requirements for acceptable contributions (e.g., a reference to any required coding standard). {Met URL} [\[contribution_requirements\]](#)

FLOSS license ...

Documentation ...

Other ...

Change Control

Public version-controlled source repository

- The project MUST have a version-controlled source repository that is publicly readable and has a URL. [\[repo_public\]](#)
- The project's source repository MUST track what changes were made, who made the changes, and when the changes were made. [\[repo_track\]](#)
- To enable collaborative review, the project's source repository MUST include interim versions for review between releases; it MUST NOT include only final releases. [\[repo_interim\]](#)
- It is SUGGESTED that common distributed version control software be used (e.g., git) for the project's source repository. [\[repo_distributed\]](#)

Unique version numbering ...

Release notes ...



Reporting

Bug-reporting process

- The project MUST provide a process for users to submit bug reports (e.g., using an issue tracker or a mailing list). {Met URL} [report process]
- The project SHOULD use an issue tracker for tracking individual issues. [report tracker]
- The project MUST acknowledge a majority of bug reports submitted in the last 2-12 months (inclusive); the response need not include a fix. [report responses]
- The project SHOULD respond to a majority (>50%) of enhancement requests in the last 2-12 months (inclusive). [enhancement responses]
- The project MUST have a publicly available archive for reports and responses for later searching. {Met URL} [report archive]

Vulnerability report process

- The project MUST publish the process for reporting vulnerabilities on the project site. {Met URL} [vulnerability_report_process]
- ...

Quality

Working build system

- If the software produced by the project requires building for use, the project MUST provide a working build system that can automatically rebuild the software from source code. {N/A allowed} [build]
- ...

Automated test suite

- The project MUST use at least one automated test suite that is publicly released as FLOSS <...> [test]

New functionality testing

- The project MUST have a general policy (formal or not) that as major new functionality is added to the software produced by the project, tests of that functionality should be added to an automated test suite. [test_policy]
- ...

Warning flags ...



Security

Secure development knowledge

- The project MUST have at least one primary developer who knows how to design secure software. (See 'details' for the exact requirements.) [\[know_secure_design\]](#)
- At least one of the project's 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. [\[know_common_errors\]](#)

Use basic good cryptographic practices ...

Secured delivery against man-in-the-middle (MITM) attacks ...

Publicly known vulnerabilities fixed ...

Other security issues ...

NOTE: Most of the OpenSSF best practices are applicable to all software!

Analysis

Static code analysis

- At least one static code analysis tool (beyond compiler warnings and "safe" language modes) MUST be applied to any proposed major production release of the software before its release, ... {**N/A justification**} {Met justification} [\[static_analysis\]](#)
- It is SUGGESTED that at least one of the static_analysis criterion include rules or approaches to look for common vulnerabilities in the analyzed language or environment. {**N/A allowed**} [\[static_analysis_common_vulnerabilities\]](#)
- All medium and higher severity exploitable vulnerabilities discovered with static code analysis MUST be fixed in a timely way after they are confirmed. {**N/A allowed**} [\[static_analysis_fixed\]](#)
- It is SUGGESTED that static source code analysis occur on every commit or at least daily. {**N/A allowed**} [\[static_analysis_often\]](#)

Dynamic code analysis ...

OpenSSF Best Practices: Criteria Statistics



Table: OpenSSF Best Practice Breakdown

Level	Total active	MUST	SHOULD	SUGGESTED	Security specific	Allow N/A	Met justification or URL required	Require URL
Passing	67	43	10	14	16	27	1	8
Silver	+55	+44	+10	1	+18	40	38	17
Gold	+23	+21	+2	0	+5	9	13	9

Notes/Observations:

- Some practices are relisted in higher levels going from SUGGESTED to SHOULD or SHOULD to REQUIRED
 - Example **SHOULD ... [bus_factor]** at Silver-level is relisted as **MUST ... [bus_factor]** at Gold-level
- **Most of the practices are NOT specific to security!**
- Also, **most of the security-specific practices are “N/A” or are easily met for most CSE/HPC software.**



OpenSSF Best Practices

“Badge App” Site

OpenSSF Best Practices: “Badge App” Site Overview



OpenSSF Best Practices

Projects Sign Up Login A

8908 Projects

Badge status All Exclude passing Text search Name or description text Text search

Add New Project

ID	Name	Description	Website	License	Owner	Last achieved at	Tiered %	Badge
1	BadgeApp	BadgeApp is the web application that allows developers to provide information about their project and (hopefully) get an Open Source Security Foundation...	https://github.com/coreinfrastructure/best-practices-badge	MIT	David A. Wheeler	2023-09-19 06:10:30	300%	
24	Zed Attack Proxy (ZAP)	Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by...	https://www.zaproxy.org	Apache-2.0	Simon Bennetts	2016-08-10 07:03:00	198%	
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	Charlemange		85%	
29	Node.js	Node.js® is a JavaScript	https://nodejs.org	MIT	Rod Vagg	2016-02-	164%	

Some Projects Earning Badges:



Source: <https://www.bestpractices.dev/en/projects>

OpenSSF Best Practices: “Badge App” Site: Gold-Level Projects



56 Projects

Badge status Gold (300%)

Exclude passing

Text search Name or description text

Text search



Add New Project

ID	Name	Description	Website	License	Owner	Last achieved at	Tiered %	Badge
1	BadgeApp	BadgeApp is the web application that allows developers to provide information about their project and (hopefully) get an Open Source Security Foundation...	https://github.com/coreinfrastructure/best-practices-badge	MIT	David A. Wheeler	2023-09-19 06:10:30	300%	openssf best practices gold
34	Linux Kernel	The Linux kernel.	https://www.kernel.org	GPL-2.0	Greg Kroah-Hartman	2018-06-14 16:10:57	300%	openssf best practices gold
63	curl	curl is a command line tool and library for internet transfers	https://curl.se	MIT	Daniel Stenberg	2016-03-24 20:14:00	300%	openssf best practices gold
74	Zephyr Project	The Zephyr Project is a small, scalable real-time operating	https://www.zephyrproject.org	Apache-2.0	Brett Preston	2018-03-10	300%	openssf best practices gold

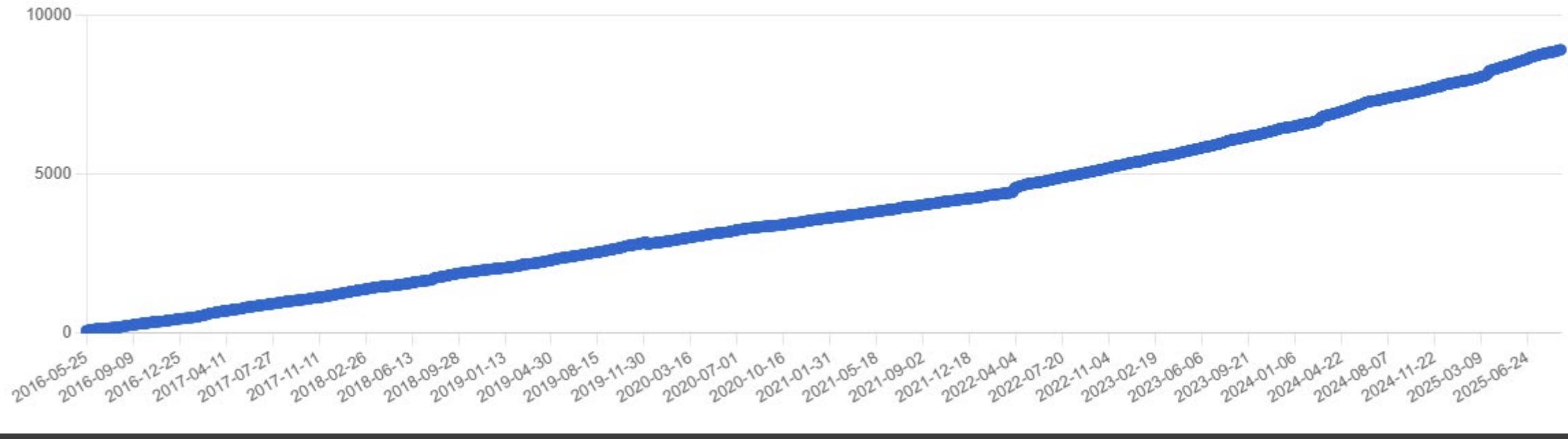
Source: <https://www.bestpractices.dev/en/projects?gteq=300>

OpenSSF Best Practices: Growth in adoption



Project Statistics

All projects



- Number of registered projects growing steadily
- Number of projects registering is continuing to accelerate!
- **Conclusion: Well accepted and adopted badge program and site!**

Source: https://www.bestpractices.dev/en/project_stats

OpenSSF Best Practices: Badge level and adoption stats

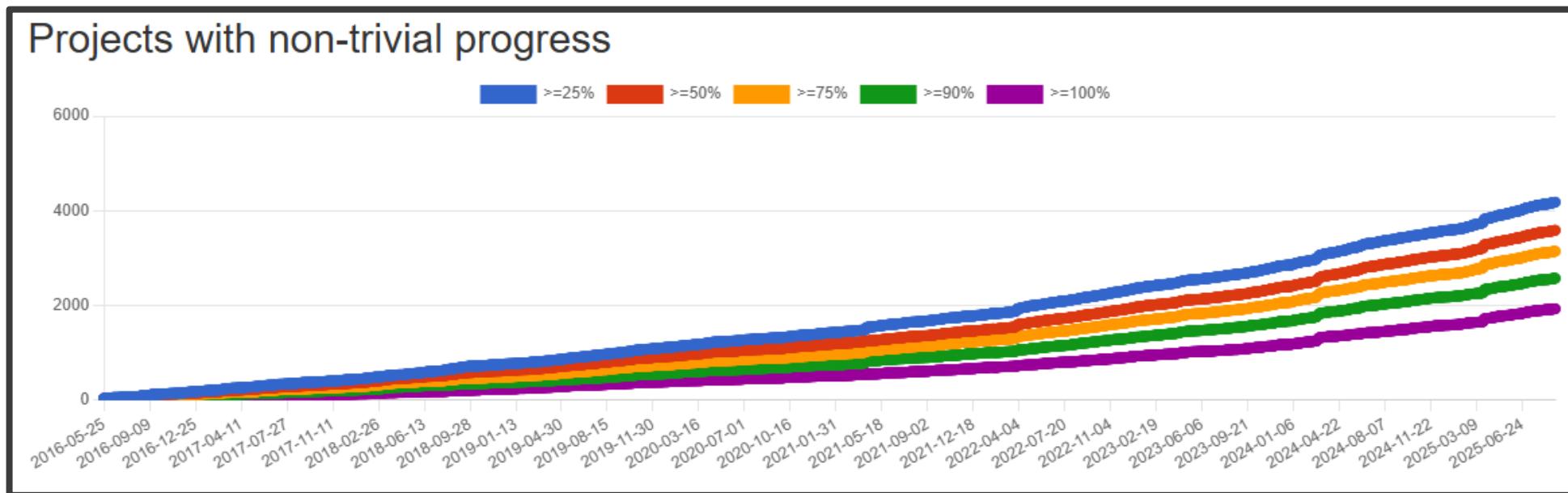


Of 8908 total registered projects (as of 9/9/2025)

- 1031 **Passing-level** projects: **11%**
- 116 **Silver-level** projects: **1.3%**
- 56 **Gold-level** projects: **0.6%**

SIDENOTE: A number of the listed projects are not official entries for those projects

Source: <https://www.bestpractices.dev/en/projects>



Source: https://www.bestpractices.dev/en/project_stats

OpenSSF Best Practices: Project Page



OpenSSF Best Practices

99% Projects Sign Up Login



TriBITS Core

[Expand panels](#) [Show all details](#) [Hide met & N/A](#)

Projects that follow the best practices below can voluntarily self-certify and show that they've achieved an Open Source Security Foundation (OpenSSF) best practices badge. [Show details](#)

If this is your project, please show your badge status on your project page! The badge status looks like this: [openssf best practices](#) in progress 99% Here is how to embed it: [Show details](#)

These are the [passing](#) level criteria. You can also view the [silver](#) or [gold](#) level criteria.

▼ Basics	13/13	•
▼ Change Control	9/9	•
▼ Reporting	8/8	•
▼ Quality	13/13	•
▼ Security	15/16	•
▼ Analysis	8/8	•

Source: <https://www.bestpractices.dev/en/projects/4839#>

OpenSSF Best Practices: Project Practice Entry



▼ Basics

13/13 •

▲ Change Control

9/9 •

•••

Release notes



- Met
- Unmet
- N/A
- ?

The project MUST provide, in each release, release notes that are a human-readable summary of major changes in that release to help users determine if they should upgrade and what the upgrade impact will be. The release notes MUST NOT be the raw output of a version control log (e.g., the "git log" command results are not release notes). Projects whose results are not intended for reuse in multiple locations (such as the software for a single website or service) AND employ continuous delivery MAY select "N/A". (URL required) [release_notes] [Show details](#)

<https://github.com/TriBITSPub/TriBITS/blob/master/tribits/CHANGELOG.md>

Source: <https://www.bestpractices.dev/en/projects/4839#changecontrol>

OpenSSF Best Practices: Project Badge Display



A screenshot of a GitHub repository's README.rst file. At the top left, it says "README.rst". On the right side, there is a small edit icon. Below the file name, there is a badge with the text "openssf best practices" and "in progress 99%". The main content of the file is the heading "TriBITS: Tribal Build, Integrate, and Test System".

A screenshot of a GitHub commit editor for the file "README.rst" in the "master" branch. The top bar shows "TriBITS / README.rst in master" with "Cancel changes" and "Commit changes..." buttons. Below the editor area, there are tabs for "Edit" and "Preview". The code editor shows the following content:

```
1 .. image:: https://bestpractices.coreinfrastructure.org/projects/4839/badge
2 :target: https://bestpractices.coreinfrastructure.org/projects/4839
3 :alt: CII Best Practices
4 =====
5 TriBITS: Tribal Build, Integrate, and Test System
6 =====
```

The "Edit" tab is selected. Above the code editor, there are settings for "Spaces" (set to 2), "2", and "Soft wrap".

Source: <https://github.com/TriBITSPub/TriBITS#readme>

Other Benefits of the OpenSSF Best Practices Badge Program

OpenSSF Best Practices as a Learning Tool



- Listing of practices includes “additional information”
 - <https://www.bestpractices.dev/en/criteria?details=true&rationale=true>
- Each practice on Badge App Project page has a “Show details” button and there is a “Show all details button” at the top of the page.
 - <https://www.bestpractices.dev/en/projects/4839#all>
- Reading though all 129 best practices with detail (and clicking on the links for more info) can take more than ½ a day (or much longer depending on the level of familiarity with each practices).

Dynamic code analysis

- It is SUGGESTED that at least one dynamic analysis tool be applied to any proposed major production release of the software before its release. [\[dynamic_analysis\]](#)

Details:

A dynamic analysis tool examines the software by executing it with specific inputs. For example, the project MAY use a fuzzing tool (e.g., [American Fuzzy Lop](#)) or a web application scanner (e.g., [OWASP ZAP](#) or [w3af](#)). In some cases the [OSS-Fuzz](#) project may be willing to apply fuzz testing to your project. For purposes of this criterion the dynamic analysis tool needs to vary the inputs in some way to look for various kinds of problems or be an automated test suite with at least 80% branch coverage. The [Wikipedia page on dynamic analysis](#) and the [OWASP page on fuzzing](#) identify some dynamic analysis tools. The analysis tool(s) MAY be focused on looking for security vulnerabilities, but this is not required.

Rationale:

Static source code analysis and dynamic analysis tend to find different kinds of defects (including defects that lead to vulnerabilities), so combining them is more likely to be effective. For example, [Linus Torvalds' "Linux 4.14-rc5" announcement \(October 15, 2017\)](#) notes that "(people are doing) random fuzzing... and it's finding things... Very nice to see."



FLOSS license

- The software produced by the project MUST be released as FLOSS. [[floss_license](#)]

Details:

FLOSS is software released in a way that meets the [Open Source Definition](#) or [Free Software Definition](#). Examples of such licenses include the [CC0](#), [MIT](#), [BSD 2-clause](#), [BSD 3-clause revised](#), [Apache 2.0](#), [Lesser GNU General Public License \(LGPL\)](#), and the [GNU General Public License \(GPL\)](#). For our purposes, this means that the license MUST be:

- [an approved license by the Open Source Initiative \(OSI\)](#), or
- [a free license as approved by the Free Software Foundation \(FSF\)](#), or
- [a free license acceptable to Debian main](#), or
- [a "good" license according to Fedora](#).

The software MAY also be licensed other ways (e.g., "GPLv2 or proprietary" is acceptable).

Rationale:

These criteria are designed for FLOSS projects, so we need to ensure that they're only used where they apply. Some projects may be mistakenly considered FLOSS even though they are not (e.g., they might not have any license, in which case the defaults of the country's legal system apply, or they might use a non-FLOSS license). We've added "produced by the project" as a clarification - many projects use non-FLOSS software/services in the process of creating software, or depend on them to run, and that is allowed.

Source: <https://www.bestpractices.dev/en/criteria?details=true&rationale=true>



FLOSS license

What license(s) is the project released under? [Show details](#)

BSD-3-Clause



- Met
- Unmet
- ?

The software produced by the project MUST be released as FLOSS. [\[floss_license\]](#) [Hide details](#)

FLOSS is software released in a way that meets the [Open Source Definition](#) or [Free Software Definition](#). Examples of such licenses include the [CC0](#), [MIT](#), [BSD 2-clause](#), [BSD 3-clause revised](#), [Apache 2.0](#), [Lesser GNU General Public License \(LGPL\)](#), and the [GNU General Public License \(GPL\)](#). For our purposes, this means that the license MUST be:

- an approved license by the Open Source Initiative (OSI), or
- a free license as approved by the Free Software Foundation (FSF), or
- a free license acceptable to Debian main, or
- a "good" license according to Fedora.

The software MAY also be licensed other ways (e.g., "GPLv2 or proprietary" is acceptable).

<https://github.com/TriBITSPub/TriBITS/blob/master/tribits/Copyright.txt> The BSD-3-Clause license is approved by the Open Source Initiative (OSI).

Source: <https://www.bestpractices.dev/en/projects/4839#basics>

OpenSSF Best Practices as Continuous Process Improvement tool



- Badge App Project page **Show unmet criteria mode** (i.e. “Expand panels” and “Hide met & N/A”) :
 - <https://www.bestpractices.dev/en/projects/4839#all>

The screenshot shows a section titled "TriBITS Core" with three buttons at the top: "Expand panels", "Show all details", and "Hide met & N/A". The "Expand panels" button is highlighted with a red box. Below these buttons is a paragraph of text: "Projects that follow the best practices below can voluntarily self-certify and show that they've achieved an Open Source Security Foundation (OpenSSF) best practices badge." A "Show details" button is shown to the right. Further down, there's another "Show details" button next to a badge status message: "openssf best practices in progress 99%". At the bottom, it says "These are the passing level criteria. You can also view the silver or gold level criteria."

- Badge App **sends out periodic email reminders** about the status for your projects and where to look for improvements.

OpenSSF Best Practices Project Page: Showing unmet criteria



TriBITS Core

[Collapse panels](#)[Show all details](#)[Show met & N/A](#)

Projects that follow the best practices below can voluntarily self-certify and show that they've achieved an Open Source Security Foundation (OpenSSF) best practices badge. [Show details](#)

If this is your project, please show your badge status on your project page! The badge status looks like this: `openssf best practices in progress 99%` Here is how to embed it: [Show details](#)

These are the `passing` level criteria. You can also view the `silver` or `gold` level criteria.

• • •

Change Control

9/9 ●

Public version-controlled source repository

Unique version numbering

 Met Unmet ?

It is SUGGESTED that the [Semantic Versioning \(SemVer\)](#) or [Calendar Versioning \(CalVer\)](#) version numbering format be used for releases. It is SUGGESTED that those who use CalVer include a micro level value. `[version_semver]` [Show details](#)

TriBITS has not put out any official releases. Customers instead use almost-continuous integration to get updates of TriBITS. In the future, official releases may be put out.

Source: <https://www.bestpractices.dev/en/projects/4839#all>

OpenSSF Best Practices Project Page: Showing unmet criteria



▲ Quality

13/13 •

Working build system

Automated test suite

New functionality testing

Warning flags

▲ Security

15/16 •

Secure development knowledge



- Met
- Unmet
- ?

The project MUST have at least one primary developer who knows how to design secure software. (See 'details' for the exact requirements.) [\[know_secure_design\]](#) [Show details](#)

The main developer does not claim to "know how to design secure software" so I can't check this. But it is hard to see how a framework for CMake build systems could open up these types of security vulnerabilities.

OpenSSF Best Practices Badge App: Standard Index into projects



- Badge App Project Page for a given project provides a standard list of practices and make it easy to find how a given project addresses various issues and how to access those.
- Example: **How to report an issue for the project?**
 - [report_process] Passing-level, Reporting, Bug reporting process

Bug-reporting process



- Met
- Unmet
- ?

The project MUST provide a process for users to submit bug reports (e.g., using an issue tracker or a mailing list). (URL required) [report_process]

Yes, either GitHub issue tracker or mailing list. See: <https://github.com/linuxfoundation/cii-best-practices-badge/blob/master/README.md>

- Example: **Documentation of the internal implementation?**
 - [documentation_architecture] Silver-level, Basics, Documentation



- Met
- Unmet
- N/A
- ?

The project MUST include documentation of the architecture (aka high-level design) of the software produced by the project. If the project does not produce software, select "not applicable" (N/A). (URL required) [documentation_architecture]

Show details

The design is documented in [doc/implementation.md](#)



Software Security Practices

OpenSSF Best Practices: OpenSSL (Heartbleed bug)



Id	Name	Description	Website	License	Owner	Last achieved at	Tiered %	Badge
54	OpenSSL	OpenSSL is an open source project that provides a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure...	https://www.openssl.org	OpenSSL	Rich Salz	2016-02-12 22:09:00	105%	openssl best practices passing
87	OpenSSL before Heartbleed	This is a <i>historical</i> badge entry for the OpenSSL project before the Heartbleed vulnerability was reported, circa February 2014. Please note that the...	http://www.openssl.org	OpenSSL	Rich Salz		63%	openssl best practices in progress 63%

Source: <https://www.bestpractices.dev/en/projects?q=openssl>

NOTE: OpenSSL Heartbleed bug was exposed and fixed in **2014** and was **only 63% of a passing** OpenSSF best practices badge!

NOTE: Even today, OpenSSL only achieves a passing badge!

Source: <https://en.wikipedia.org/wiki/Heartbleed>

OpenSSF Best Practices: Security Focus?



Software Security Practices:

- 16 of 67 **Passing-level** practices (9 of 16 allow N/A)
- +18 of +55 **Silver-level** practices (11 of 18 allow N/A)
- +5 of +23 **Gold-level** practices (3 of 5 allow N/A)

Until recently, software security is barely mentioned in most software engineering books:

Classic Example: "Code Complete: 2nd Edition", 800+ pages, 2004: **Exactly one paragraph** is devoted to the area of software security in section 3.5 "Architecture Prerequisite":

The architecture should describe the approach to design level and code level security. If a thread model has not previously been built, it should be built at architecture time. Coding guidelines should be developed with security implications in mind, including approaches to handling buffers, rules for handling untrusted data (data input from users, cookies, configuration data, and other external interfaces), encryption, level of detail contained in error messages, protected secrete data that's in memory, and other issues.

More Recent Example: "The Pragmatic Programmer: 20th Anniversary Edition", 300+ pages, 2020: **Devotes 7 pages** to software security in Topic 42 "Stay Safe Out There".

Awareness and visibility of software security is increasing => **Bigger issue due to AI Agents!**



Passing-level

Security

Secure development knowledge

- The project MUST have at least one primary developer who knows how to design secure software. (See 'details' for the exact requirements.) [[know_secure_design](#)]
- At least one of the project's 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](#)]

What does it mean to “**knows how to design secure software**” and “**know of common kinds of errors that lead to vulnerabilities in this kind of software**”?

NOTE: These are not even practices!

=> These are asking the question “**Do you know what you are doing?**”



Security

Secure development knowledge

- The project MUST have at least one primary developer who knows how to design secure software. (See 'details' for the exact requirements.) [know secure design](#)

Details:

This requires understanding the following design principles, including the 8 principles from [Saltzer and Schroeder](#):

- economy of mechanism (<...>), fail-safe defaults (<...>), complete mediation (<...>), open design (<...>), separation of privilege (<...>), least privilege (<...>), least common mechanism (<...>), psychological acceptability (<...>), limited attack surface (<...>), input validation with allowlists (<...>)

A "primary developer" in a project is anyone who is familiar with the project's code base, <....> If there is only one developer, that individual is the primary developer. Many books and courses are available to help you understand how to develop more secure software and discuss design. For example, the [Secure Software Development Fundamentals](#) course is a free set of three courses that explain how to develop more secure software (it's free if you audit it; for an extra fee you can earn a certificate to prove you learned the material).

What is needed to check this box? [OpenSSF: Concise Guide for Developing More Secure Software](#):

- **Learn about secure software development.** Take, e.g., the [free OpenSSF course](#) or the hands-on [Security Knowledge Framework](#) course. [SAFECode's Fundamental Practices for Secure Software Development](#) provides a helpful summary.
 - [Free OpenSSF course](#) : 18 hours every 2 years (least time consuming option?)



Security

Secure development knowledge

- At least one of the project's 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. [\[know_common_errors\]](#)

Details:

- Examples (depending on the type of software) include SQL injection, OS injection, **classic buffer overflow**, cross-site scripting, missing authentication, and missing authorization. See the [CWE/SANS top 25](#) or [OWASP Top 10](#) for commonly used lists. Many books and courses are available to help you understand how to develop more secure software and discuss common implementation errors that lead to vulnerabilities. For example, the [Secure Software Development Fundamentals](#) course is a free set of three courses that explain how to develop more secure software (it's free if you audit it; for an extra fee you can earn a certificate to prove you learned the material).

Common Security Vulnerabilities? 2024 CWE Top 25



Security vulnerabilities related to CSE/HPC Software (11 of 25)

Rank	ID	Name	Score	CVEs in KEV	Rank Change vs. 2023
2	CWE-787	Out-of-bounds Write	45.20	18	-1
6	CWE-125	Out-of-bounds Read	11.42	3	+1
8	CWE-416	Use After Free	10.19	5	-4
11	CWE-94	Improper Control of Generation of Code ('Code Injection')	7.13	7	+12
12	CWE-20	Improper Input Validation	6.78	1	-6
16	CWE-502	Deserialization of Untrusted Data	5.07	5	-1
20	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	3.69	2	-3
21	CWE-476	NULL Pointer Dereference	3.58	0	-9
23	CWE-190	Integer Overflow or Wraparound	3.37	3	-9
24	CWE-400	Uncontrolled Resource Consumption	3.23	0	+13

- 5 of the top 25 are very common in all C/C++/Fortran/CUDA/HIP/... CSE/HPC codes!
- Many of these are major causes of software bugs impacting software correctness in CSE/HPC codes

Conclusion: Software correctness & quality and software security are best friends

Source: https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html#top25list

Application of the OpenSSF Best Practices Badge Program to Scientific Software: Challenges?



Are the **burdens of the Security Criteria** too much for CSE/HPC software?

- **Most security-related criteria can be marked “N.A.”** (e.g., [crypto_published])

Met
 Unmet
 N/A
 ?

The software produced by the project MUST use, by default, only cryptographic protocols and algorithms that are publicly published and reviewed by experts (if cryptographic protocols and algorithms are used). [\[crypto_published\]](#) [Show details](#)

There is no encryption associated with TriBITS code.

- **What about “The project **MUST** have at least one primary developer who knows how to design secure software”** [\[know_secure_design\]](#)?

⇒ We could **define different categories of CSE/HPC software** and software development and **minimal training criteria for each level**:

- ⇒ **Level 1:** Basic CSE/HPC Numerical Software => 4 hours, every 5 years?
- ⇒ **Level 2:** CSE/HPC DevOps Software => 10 hours, every 3 years?
- ⇒ **Level 3:** System Software/User Management => [Free OpenSSF course](#) : 18 hours, every 2 years?



Ginkgo

**OpenSSF Best Practices
Badge Program Experience**

OpenSSF Badge efforts: Ginkgo

HPC-BP webinar, Sep 24, 2025

Pratik Nayak, Computational Mathematics, Technical University of Munich, Germany.

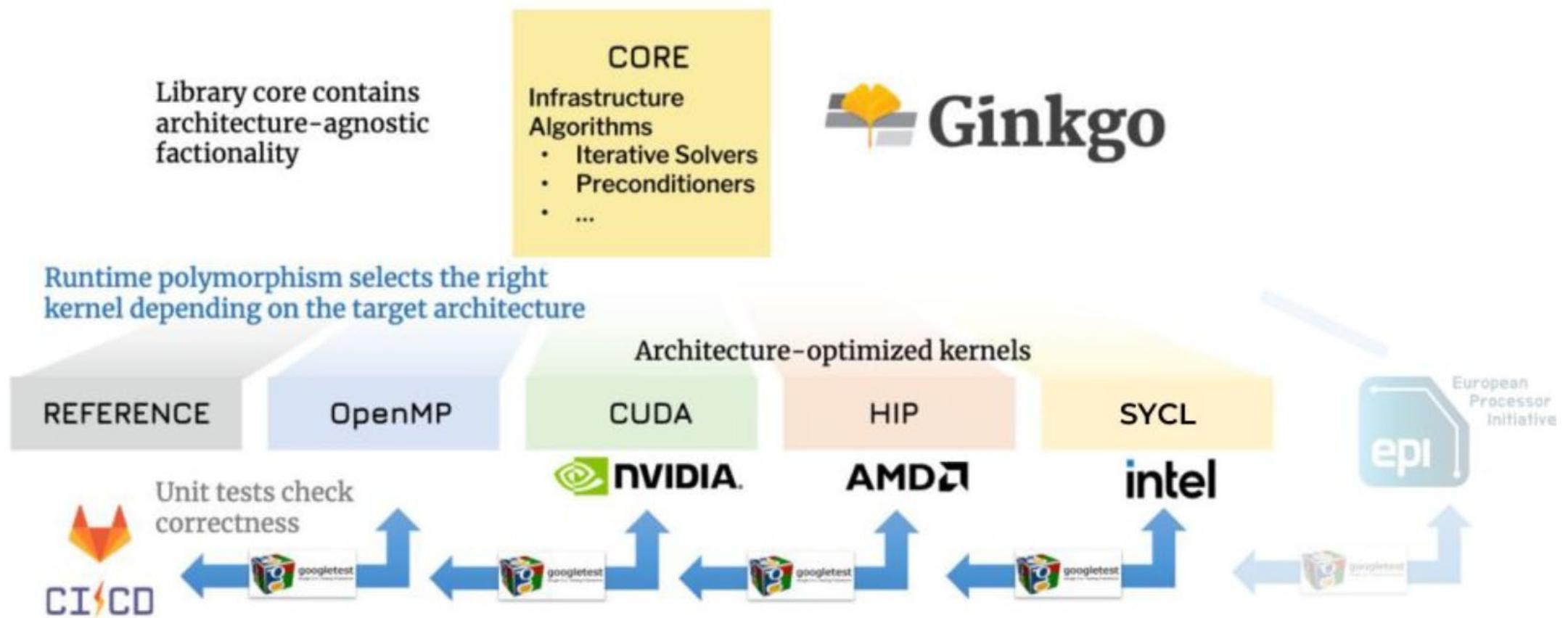


Ginkgo: A high performance numerical linear algebra library

- Open-source, Modern C++, robust with comprehensive unit tests, benchmarks and examples.
- Heavily focussed on user-friendly interface, and low-cost integration for applications.
- Support for various hardware backends:
 - NVIDIA, AMD and Intel GPUs,
 - Multi-core CPUs with OpenMP
 - Distributed support with MPI.
- Main features: Sparse matrices, Krylov solvers, preconditioners, etc
- Integrated into various applications: MFEM, SUNDIALS, deal.II, NekRS, OpenFOAM, OpenCARP.



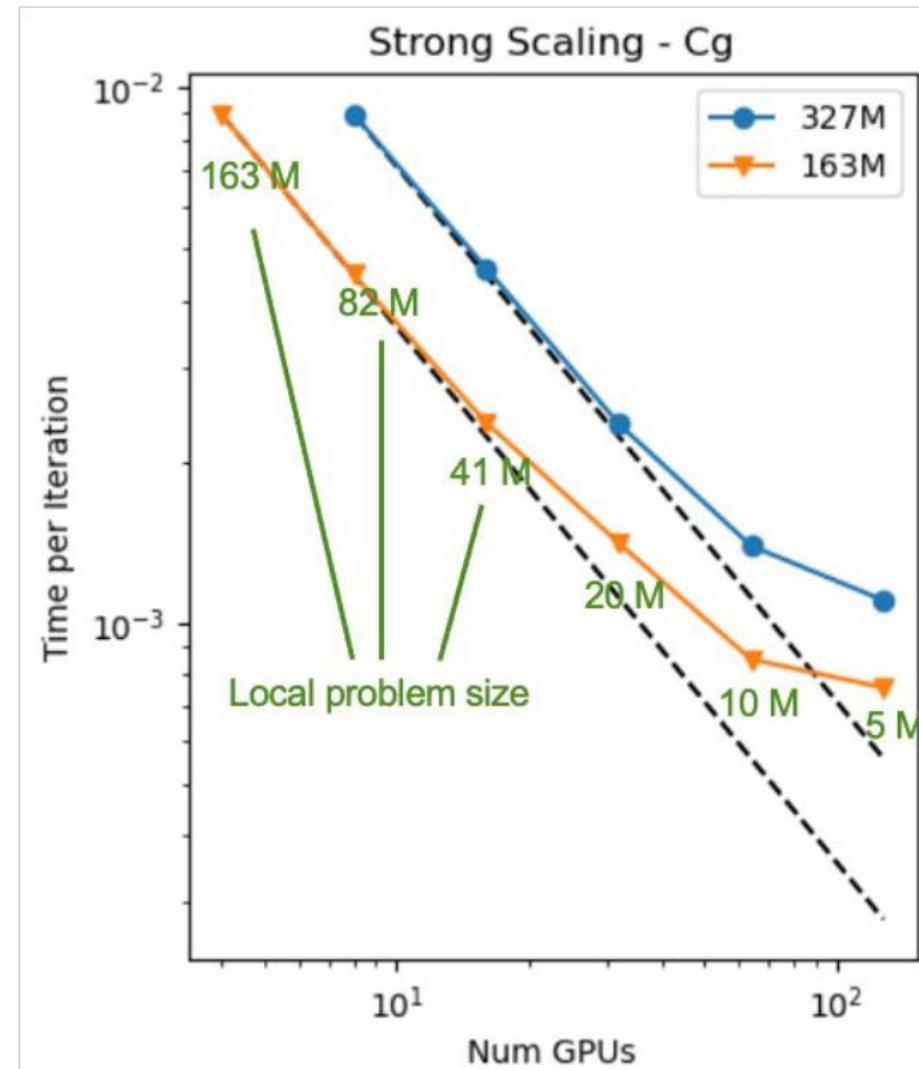
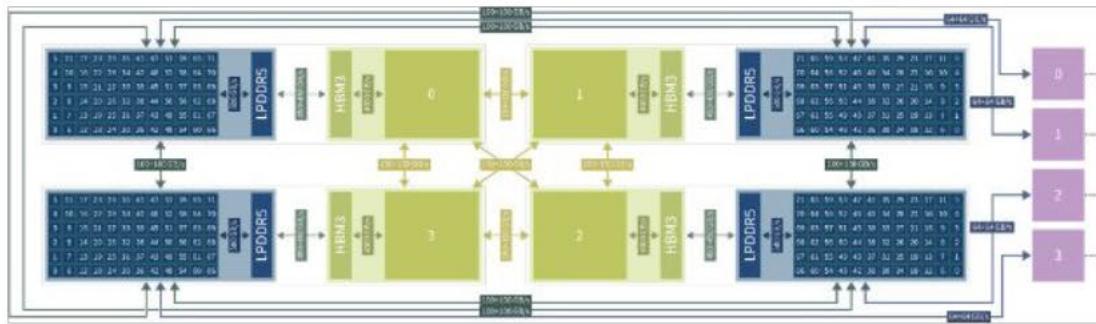
The Ginkgo software library



Scalability of Ginkgo



JUPITER



Strong scaling on Jupiter.

Ginkgo functionality

	OMP	CUDA	HIP	SYCL
Basic				
SpMV	✓	✓	✓	✓
SpMM	✓	✓	✓	✓
SpGeMM	✓	✓	✓	✓
BiCG	✓	✓	✓	✓
BiCGSTAB	✓	✓	✓	✓
CG	✓	✓	✓	✓
CGS	✓	✓	✓	✓
GCR	✓	✓	✓	✓
GMRES	✓	✓	✓	✓
FCG	✓	✓	✓	✓
FGMRES	✓	✓	✓	✓
IR	✓	✓	✓	✓
IDR	✓	✓	✓	✓
Chebyshev	✓	✓	✓	✓
MinRES	✓	✓	✓	✓
Krylov solvers				
Block-Jacobi	✓	✓	✓	✓
Gauss-Seidel/SSOR	✓	✓	✓	✓
ILU/IC	✓	✓	✓	✓
Parallel ILU/IC	✓	✓	✓	✓
Parallel ILUT/ICT	✓	✓	✓	✓
ISAI	✓	✓	✓	✓
Preconditioners				
Batched				
Batched BiCGSTAB	✓	✓	✓	✓
Batched CG	✓	✓	✓	✓
Batched GMRES	✓	✓	✓	✓
Batched ILU	✓	✓	✓	✓
Batched ISAI	✓	✓	✓	✓
Batched Block-Jacobi	✓	✓	✓	✓
AMG				
AMG preconditioner	✓	✓	✓	✓
AMG solver	✓	✓	✓	✓
Sparse direct				
Parallel Graph Match	✓	✓	✓	✓
Symbolic Cholesky	✓	✓	✓	✓
Numeric Cholesky	✓	✓	✓	✓
Symbolic LU	✓	✓	✓	✓
Numeric LU	✓	✓	✓	✓
Utilities				
Sparse TRSV	✓	✓	✓	✓
On-Device Matrix Assembly	✓	✓	✓	✓
MC64/RCM reordering	✓	✓	✓	✓
Wrapping user data	✓	✓	✓	✓
Logging	✓	✓	✓	✓
PAPI counters (SDE)	✓	✓	✓	✓
Config file support	✓	✓	✓	✓

✓ MPI Support

✓ Single-GPU Support

pyGinkgo: A python interface for Ginkgo

```

import pyGinkgo as pg
import numpy as np

fn = 'm1 mtx'
dev = pg.device("cuda")
mtx = pg.read(device=dev, path=fn, dtype="double", format="Csr")
n_rows = mtx.size[0]

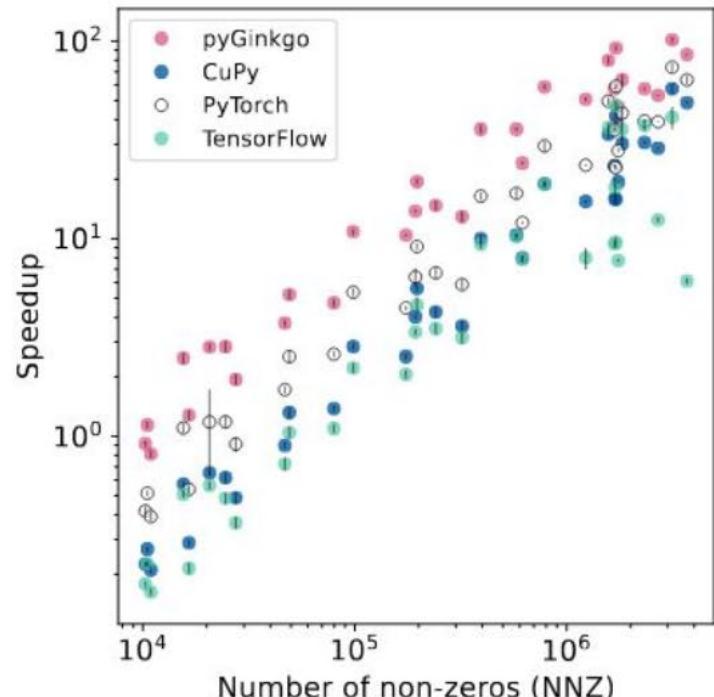
b = pg.as_tensor(
    device=dev, dim=(n_rows,1), dtype="double", fill=1.0
)
x = pg.as_tensor(
    device=dev, dim=(n_rows,1), dtype="double", fill=0.0
)

# Create ILU preconditioner
preconditioner = pg.preconditioner.Ilu(dev, mtx)

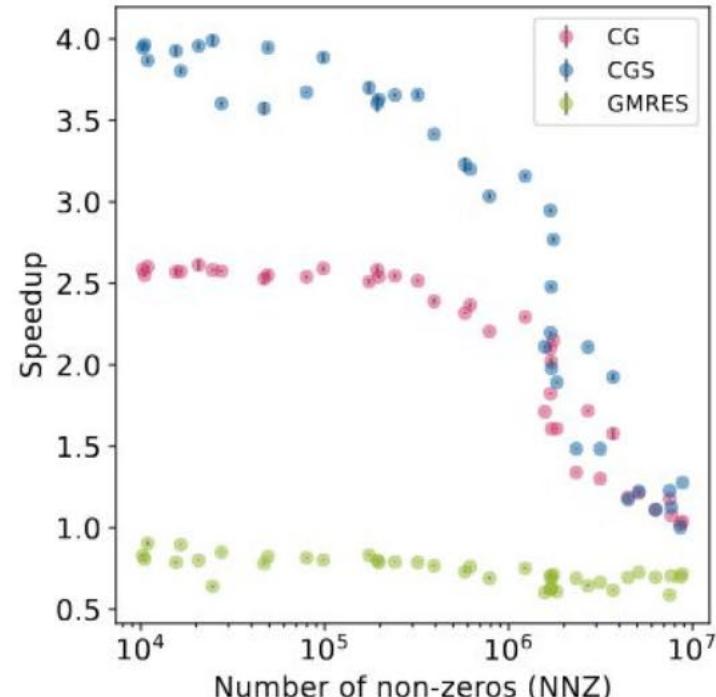
#Setup GMRES solver
solver = pg.solver.gmres(dev, mtx, preconditioner,
    max_iters=1000, krylov_dim=30, reduction_factor=1e-06
)

#Apply
logger, result = solver.apply(b, x)

```



(a) SpMV on NVIDIA A100 GPU: Variation of speedup relative to SciPy with number of non-zeros



(c) Solver on NVIDIA A100: Variation of speedup relative to CuPy with number of non-zeros for 1000 solver iterations

xSDK: An Exascale Computing Project effort

- Provide a toolkit with set of scientific software that are interoperable.
- Each software has to follow a set of community package policies¹.
 - Some Mandatory policies that ensure interoperability, and ensure best practices.
 - M1: Spack installation, M2: Comprehensive test suite, M4: Platform portability, M5: Contact person
 - Some Recommended policies that encourage software sustainability, extensibility and security.
 - R1: Smoke tests, R2: Valgrind checks in CI, R5: Provide list of dependencies.
- Ginkgo is a part of xSDK, and was developed with software sustainability in mind.



¹ <https://x sdk.info/policies/>

Getting the OpenSSF badge

- Badge details available here: <https://www.bestpractices.dev/en/projects/8037>
- Achieved **passing** badge on 4th Nov, 2023 before our 1.7.0 release (current release 1.10.0)



Challenges in obtaining the badge

- Basics, Change control, Reporting were easy to obtain as they are common practice.
- Quality (build, tests, warnings) requires some attention, but as we were part of xSDK, it made things easier for us.
- Security was the more involved part for us.
 - Checking design principles were followed, some were applicable, some were only loosely applicable.
 - <https://openssf.org/training/courses/> provides a course, but only some topics are relevant (but still useful in general).
 - We ended up selectively taking parts of the course that were relevant for us.
 - Input validation, error handling, code scanners, code analysis tools

Badge extensions: Silver and Gold level criteria

- OpenSSF also provides silver and gold criteria
- Ginkgo already fulfills some of them:
 - Static and dynamic analysis
 - Coding standards.
 - Copyright requirements
- But not all of them:
 - Upgrade paths, and support for older versions of the product (easy to fix).

Benefits of the badge

- Security issues are also usability and performance issues.
 - Out of bounds reads/writes, memory leaks, use after free etc are common issues in most HPC software written in C/C++.
 - Conscious checks done when obtaining the badge can help identify these issues.
 - Automated static/dynamic analysis checks help identify and fix performance/usability issues much earlier making the code base more robust.

Benefits of the badge

- Many HPC codes are still written in C/C++, these seem to have higher number of security vulnerabilities.
- Having an OpenSSF badge shows that basic requirements for secure software are met
 - Builds trust in users.
 - Makes the software more sustainable.
 - Can be advertised when applying for grants/funding.



MPICH

OpenSSF Best Practices Badge Program Experience

OpenSSF Best Practices in MPICH

Yanfei Guo

Computer Scientist

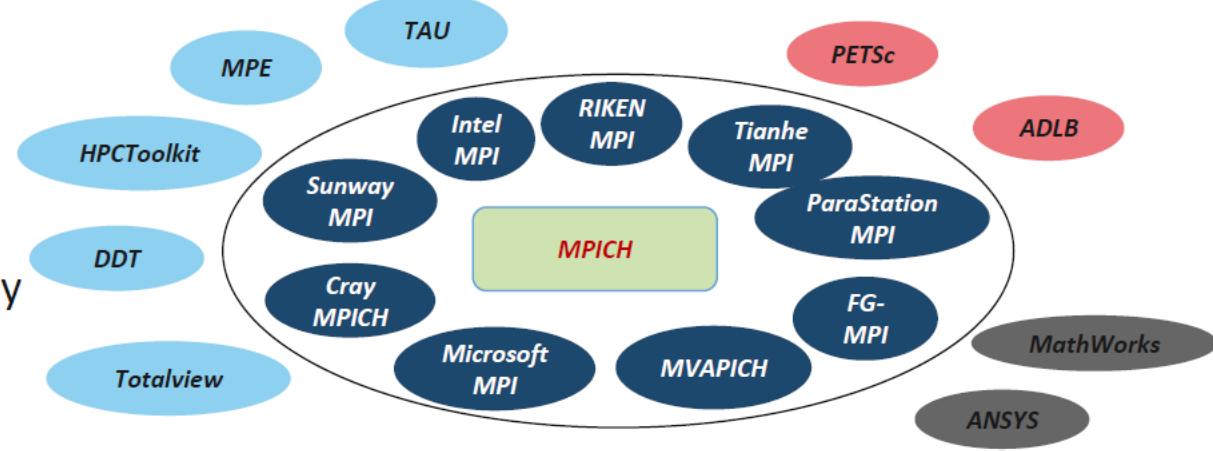
Argonne National Laboratory



U.S. DEPARTMENT OF
ENERGY

The MPICH Project

- Funded by DOE for 33 years
- Has been a key influencer in the adoption of MPI
 - First/most comprehensive implementation of every MPI standard
 - Allows supercomputing centers to not compromise on what features they demand from vendors
- DOE R&D100 award in 2005 for MPICH
- DOE R&D100 award in 2019 for UCX (MPICH internal comm. layer)
- MPICH and its derivatives are the world's most widely used MPI implementations (7 of Top 10 Supercomputers)



***MPICH is not just a software
It's an Ecosystem***

Totals grouped by language (dominant language first):	
ansic:	615297 (88.98%)
f90:	52018 (7.52%)
sh:	12904 (1.87%)
perl:	5412 (0.78%)
python:	2349 (0.34%)
cpp:	1818 (0.26%)
java:	1673 (0.24%)
php:	8 (0.00%)
sed:	4 (0.00%)

generated using David A. Wheeler's 'SLOCCount'.

MPICH Project Info

- Source code hosted on Github <https://github.com/pmodels/mpich>
 - Library, dependencies, tests
- Issues, Pull Requests, Wiki and Discussion using Github
- Self-hosted website <https://www.mpich.org> and mailing lists



- Self-hosted Jenkins CI system <https://jenkins-pmrs.cels.anl.gov/>
 - Testing for development and release, triggered by Github
- ALCF CI for testing with Aurora
- Weekly development calls and monthly meetings with vendor partners

Development Cycle

- Creating PR to MPICH
- Basic Checks
 - Authorship, Warning, Whitespace, Spelling
- Functional Tests
 - Communication library, GPU, compilers
- Merging
- Nightly, Weekly Tests
 - Updates in main branch triggers nightly tests

Warning flags



- Met
- Unmet
- N/A
- ?

The project MUST enable one or more compiler warning flags for code quality errors or common simple mistakes, if the selected language. [warnings] [Show details](#)



- Met
- Unmet
- N/A
- ?

The project MUST address warnings. [warnings_fixed] [Show details](#)

MPICH Testing

■ Test Suite and Benchmarks

- MPICH Test Suite (3000+ test jobs), OSU Benchmarks, and Custom Benchmarks
- Various compilers (gcc, clang, intel, nvidia, cray)
- Various system images (x86_64, 32bit, macos, freebsd, ARM)

Totals grouped by language (dominant language first):
ansic: 163001 (78.82%)
f90: 20184 (9.76%)
fortran: 7988 (3.86%)
sh: 6236 (3.02%)
cpp: 5758 (2.78%)
perl: 3151 (1.52%)
python: 437 (0.21%)
ruby: 53 (0.03%)

generated using David A. Wheeler's 'SLOCCount'.

Automated test suite



- Met
 Unmet
 ?

The project MUST use at least one automated test suite that is publicly released as FLOSS (this test suite may be maintained as a separate FLOSS project). The project MUST clearly show or document how to run the test suite(s) (e.g., via a continuous integration (CI) script or via documentation in files such as BUILD.md, README.md, or CONTRIBUTING.md). [test] Show details



- Met
 Unmet
 ?

A test suite SHOULD be invocable in a standard way for that



- Met
 Unmet
 ?

It is SUGGESTED that the test suite cover most (or ideally all)



- Met
 Unmet
 ?

It is SUGGESTED that the project implement continuous integration, using a central code repository and automated tests are run on the

Development Cycle

- Creating PR to MPICH
- Basic Checks
 - Autotools, Warning, Whitespace, Spelling
- Functional Tests
 - Communication libraries, MPI, compilers
- Merging
- Nightly, Weekly Tests
 - Updates in main branch trigger nightly tests



Screenshot copied to clipboard

Automatically saved to screenshots folder.

Markup and share

MPICH Jenkins CI

Jenkins

- Dashboard >
- Build History
- Project Relationship
- Check File Fingerprint
- Disk Usage
- Job Import Plugin

Build Queue
No builds in the queue.

Build Executor Status

Node	Status	Idle
aws	Idle	0
pmrs-freebsd64-240-01	Idle	1

S	W	Name ↓
		mpich-authorship-check
		mpich-review-arm
		mpich-review-async
		mpich-review-ch3-compilers
		mpich-review-ch3-sock
		mpich-review-ch3-tcp
		mpich-review-ch4-armci-mpi
		mpich-review-ch4-ccl
		mpich-review-ch4-gpu-ofi
		mpich-review-ch4-gpu-ucx
		mpich-review-ch4-ofi
		mpich-review-ch4-ofi-more
		mpich-review-ch4-thread-package
		mpich-review-ch4-ucx
		mpich-review-ch4-xpmem
		mpich-review-compilers

test:mpich/ch4/most
test:mpich/ch3/most

Review required
At least 1 approving review is required by reviewers with write access.

Some checks were not successful
2 failing, 6 successful checks

2 failing checks ▾

- mpich-review-ch4-ofi — Build finished. 15970 tests run, 85 skipped, 1 failed.
- mpich-review-ch4-ucx — Build finished. 25485 tests run, 182 skipped, 204 failed.

6 successful checks ▾

- contribution-checker — Build finished. Required
- mpich-review-ch3-sock — Build finished. 9324 tests run, 48 skipped, 0 failed.
- mpich-review-ch3-tcp — Build finished. 12460 tests run, 70 skipped, 0 failed.
- mpich-warnings — Build finished. Required

Merging is blocked
At least 1 approving review is required by reviewers with write access.

Merge without waiting for requirements to be met (bypass rules)

Merge pull request You can also merge this with the command line. [View command line instructions.](#)

Unique Challenge for Testing HPC Software

- We have to self-host the Jenkins CI (and the testing cluster)
 - Need to test on HPC hardware
 - Cannot integrate lab clusters with Github Actions directly
- MPICH's Jenkins setup took 3 major revisions in more than 10 years
 - Initial setup was straight forward, tests were triggered manually
 - Second setup using Github hooks to trigger jobs from PR
 - Third setup focus on codesign the CI and the test suite to improve efficiency

Meeting Security Requirements

- Security development knowledge
 - Developing Secure Software (LFD121) Course
 - Mostly target developer for software and service in untrusted environment
 - Many of the knowledge is less relevant to HPC applications
- Use basic good cryptographic practices
- Secured delivery against MITM
 - https
- Publicly known vulnerabilities fixed
- Code repo not leaking private credentials
 - No credential included in code repo

Analysis Tools Used in MPICH Development

- Static Analysis Tools
 - Coverity (<https://scan.coverity.com/>)
 - Initially reported a few thousands issues for MPICH. Bad codes, false positives
- Dynamic Analysis Tools and Sanitizer
 - Clang Address Sanitizer and Undefined Behavior Sanitizer
 - Valgrind
 - More reliable reports.
- Significant efforts to cleanup those issues
 - Much easier to keep it clean afterward
 - Highly recommend for both security and quality

What we needed to do to get OpenSSF Badge

- Easy to achieve



- Limited scope



- LFD121 is somewhat heavyweight for HPC software developers. A lightweight course could reduce the barrier

- Constant effort



- High initial cost, but easy afterward



OpenSSF Best Practices Badge Program: Summary



- OpenSSF Best Practices provides **actionable collection of open source communities best practices**.
- OpenSSF Best Practices **Badge App site codifies best practices** and provides blanks for projects to fill in.
- **Badge can be displayed on project's hosting site** when a given badge level is earned (or showing progress towards a badge).
- Badge App site contains features to **identify areas of improvement**.
- Badge App site can send out **regular reminders** to keep making progress.
- Badge App site provide a ready implementation **foundation for a customized best practices sites**. => **Has been utilized at Sandia National labs to create a customized site!**
- OpenSSF Best Practices Badge Program elevates security to a first-level concern for everyone developing software? (**What the minimum sufficient level of knowledge?**)

Create an OpenSSF Best Practices entry for one of your software projects?

<https://www.bestpractices.dev/en/projects>

Comments or Questions?