HPC-BP Webinar Q&A

# Application of the OpenSSF Best Practices Badge Program to Scientific Software

Date: 24 September 2025
Presented by: Roscoe A. Bartlett (Sandia National Laboratories), Yanfei Guo (Argonne National Laboratory), and Pratik Nayak (Technical University of Munich)

(The slides are available via the link in the page's sidebar.)

---

**Q:** Is the reminder thing [from the OpsenSSF badging program website] new? I've never gotten one

**A:** Looking at the code that implements email notifications, it seems that it sends out notifications only to the badge app project owner, and only if your badge app account has notifications enabled, and the project does not yet have a passing badge, and it has been more than 60 days since the the edit to the project or the last reminder email was sent out.

**Q:** For the [know_secure_design] criteria, are there specific requirements for the courses that satisfy, or are they just meant to be examples, representing the level of knowledge expected?

**A:** Reading over know_secure_design it says "requires understanding the following design principles, including the 8 principles from Saltzer and Schroeder". But what does it mean to "know" these things? It mentions "Many books and courses are available to help you understand how to develop more secure software and discuss design" and it gives, as the only specific example, Secure Software Development Fundamentals. That course says it takes 14-18-hours to complete and must be retaken every 2 years to maintain certification. I think this is left somewhat ambiguous on purpose, as it allows different projects and communities to define this for themselves. But it does define a high bar in general. I think we, as an HPC community, should define different levels of knowledge and training for our projects to "know secure design" for our types of software. That is likely the responsible thing to do to provide some uniformity between projects and some confidence to our stakeholders.

---

*A relevant discussion thread from the Zoom chat, lightly edited. Chat participants denoted Participant A, B, C, D. Speaker comments added later denoted by their names..*

**Participant A:** The OpenSSF course... is not well suited to scientific software development.

**Participant B:** Why not?

**Participant A:** It focuses a lot on security for web-facing software. I talked to the OpenSSF people about trying to work with us to develop a course for "unix user-land software" and they were not interested at all.

**Ross Bartlett:** But nearly all of the security items that don't apply to your project can be marked as "N.A." So what is the problem other than the requirement to "know secure design"? (And for that, we need to define what that means for different levels of projects in our space.)

**Participant C:** The NSF Trusted CI center has done a fair amount of training, and although that also leans towards web-facing and other situations which are "real" security concerns, they have a better understanding of "scientific" software.

**Participant B:** I think scientific software may increasingly need to do web- (or other connection-facing) stuff… and listening on ports, securing connections, etc. is where a lot of the issues are for securing software. What would your user land-only security course cover? If it's supplemental that could be helpful, but I think it's hard to say that scientific software developers shouldn't be aware of "web" vulnerabilities — even the CI we use, and internal services at HPC centers like Jupyter notebooks, use REST APIs and listen on ports. We've had unknowledgeable scientific software developers introduce security issues with their github actions that could have allowed their projects to be compromised.

**Ross Bartlett:** Exactly. If everyone was just writing and running numerical software on their local machine behind a firewall, security would not be much of a concern. But that is no longer the case (and has not been so for many years).

**Participant B:** This may be off base but I worry about scientific software constantly deeming itself "special". We use web applications in scientific software, and not all scientific software is "unix user-land software", nor is that a sufficiently narrow domain to cover the threat vectors we're likely to face. Scientific software increasingly relies on "mainstream" software and shouldn't rule out threats because they think their environment will always look like it does today.

**Ross Bartlett:** Agreed

**Participant A:** Except the OpenSSF course doesn't really cover any of that stuff either.

**Participant B:** The other major vector I suspect isn't fully addressed here is software supply chain… which is even harder :)

**Participant D:** Yes, but only by paying more people (or trusting ai curated feedstock). The comment about docker made me twitch.. right now most people absolutely don't vet what comes in with their containers, and supply chain is exactly that problem. Dataex from a container used to be thought difficult but is now eminently feasible. Github has just upgraded to deal with the current node fiasco, but people are still not convinced. I use a hardware crypto cert for software signing .. some people think that may be the only way.