




It's like flossing your teeth: On the Importance and Challenges of Reproducible Builds for Software Supply Chain Security

Marcel Fourné ^{*}, Dominik Wermke[†], William Enck [‡], Sascha Fahl [†], Yasemin Acar [¶]

^{*}Max Planck Institute for Security and Privacy, Bochum, Germany, marcel.fourne@mpi-sp.org

[†]CISPA Helmholtz Center for Information Security, Germany, [\[first.last\]@cispa.de](mailto:[first.last]@cispa.de)

[‡]North Carolina State University, Raleigh, North Carolina, USA, whenck@ncsu.edu

[¶]Paderborn University, Germany, George Washington University, USA, acar@gwu.edu

Codebook

1. Background
 - a. Professional Experience
 - b. Joining project
 - i. Interests
2. Reasons
 - a. (non-)technical
 - i. Internal and External drivers
 - A. Compilers work like mathematical functions
 - B. Broken expectations
 - C. Want to do/have quality
 - D. Want to build infrastructure
 - E. Wants to work on leaf packages
 - F. Self-guided exploration
 - G. Source code is out in the open, so anybody can see it
 - H. Build speed/caching
 - ii. Community requests
 - A. Snowden leaks/security incident/privacy protection
 - iii. Security for developers
 - b. Threats
 - i. Specific threats
 - ii. Incidents or requirements
 - A. Specific incident found by reproducible builds
 - c. Project decision process
 - i. Started themselves independently
 - ii. Consensus
 - iii. Corporate decision/management
3. Process
 - a. People involved
 - i. Communication
 - ii. Detractors
 - b. Starting topic
 - i. Build process
 - A. CI or other build infrastructure
 - B. Version pinning
 - ii. Upstream interaction
 - A. Community building
 1. Documentation work
 - B. Communication
 1. Patience necessary
 2. Good communication skills more necessary than expected
 - C. Reproducible Builds buy-in
 - D. Common community requests
 - E. Receptive upstream
 - F. Receptive compiler authors
 - G. Patch polishing was necessary
 - H. Upstream rewrote patches themselves
 - iii. Decision criterion for reproducibility
4. Tooling
 - a. Helpful tools used
 - i. Integration into build process
 - b. Other resources used
5. Obstacles
 - a. Technical
 - i. Build date included, SOURCE_DATE_EPOCH
 - ii. Build directory included in full, BUILD_PATH_PREFIX_MAP
 - iii. Compiler included randomness (symbols, ...)
 - iv. Profile Guided Optimization (PGO)
 - v. Cryptographic signatures included in binaries
6. Helpful factors
 - a. Self-effective participants
7. Target changes
8. Changes on starting over
 - a. Regret not doing more outreach
9. How specific
 - a. Direct port of procedures and tools
10. Misc
 - a. Bootstrappable Builds

Questionnaire

Intro

- **Thanks:** Thank you very much for offering your valuable time for this interview. We are very grateful for your contribution.
- **Ready:** Are you ready to start the interview?
- **Structure:** First off, I am going to talk about the context and data handling, and if you agree with everything, we would then start with the interview.

Context

- **We:** We are researchers at [anonymized for submission]
- **Our research:** focuses on the area “Security impact of and experiences with reproducible builds”.
- This interview is a start/exploration of internal processes and decisions often not visible at the technical level.
- For this interview:
 - We are not judging security or technological decisions of a project, we are just interested in the underlying structures and processes.
 - Projects are often very complex, if you don’t know the answer, or cannot speak about a question for any reason, just say “next”.
 - We are not just interested in structures, but also your personal opinions and experiences.
- **Questions?** Any questions about the interview context so far?

Consent

- **Voluntary:** Your responses in this interview are entirely voluntary, and you may refuse to answer any or all of the questions in this interview.
- **Duration:** Duration of the interview depends a bit on the duration of your answers, in our experience so far about X to Y minutes.
- We will de-identify you and your projects in any publication and only include short quotes.
- We will send you a preprint before a potential publication, if you want.
- **Recording:** We would like to record this interview so that we can transcribe the answers later
 - The recording will be destroyed when we transcribed the interview
- **Questions?** Any more questions about data handling or recording?
- I will now start the recording
- “The recording is now on” **SWITCH ON RECORDING**
- Restate consent question

Section 1 - Intro [Personal / General / Project]

1. To start, we are interested in your background and that of [project we are interested in re: reproducible builds]. Please tell us a little bit about how you got involved?
 - a. Coursework?
 - b. Professional experience?
 - c. How did you get into [project]?

- i. What did you find interesting about [project]?
 - ii. And how long have you been working on [project]?
 - iii. Which programming languages are commonly used in [project]?
- d. [if [project] uses more than on PL:] Specific programming language background/experience?
 - i. Do you have experience with some/all of the programming languages used in [project]?
2. Could you please elaborate a bit about your role in [project]?
 - a. What packages do you work on, what do they do?
 - b. How did you get from working generally on [project] to working on reproducible builds?
 3. Could you explain what reproducible builds mean to you?
 - a. In the context of project
 - b. In general

In the context of our research, we’ll be talking about reproducible builds in this interview.

When we say reproducible builds, we mean that for each version of the project, anyone can take the source artifacts and in the best case build a package that is bit-for-bit identical at any point now or in the future.

Comparison that two packages are built from the same source should, at the very least, be possible via human inspection.

Ideally, the comparison should be as automated as possible.

We want to allow different parties to determine if a binary package matches its source code, eliminating possible backdoors during the build process.

Section 2 - Reasons, Decisions

1. [project] has/[has not yet] made progress towards reproducible builds. We’ll be very interested in the process in a moment. Before we start on that, we are interested in your reasons for making [project] reproducible?
 - a. Technical and also non-technical?
 - i. Internal/external drivers?
 - ii. Community requests (users vs. developers)?
 - iii. Security for developers? (by getting the software out of your sole control, making the build environment on your machine less of a target for attackers)
 - b. What threats are you protecting against?
 - i. Are there any specific threats?
 - ii. Specific incidents/requirements?
 - c. What were the reasons against making [project] or individual packages fully reproducible as of now? (not yet, or not at all)

- d. How did the project decide? Who made these decisions, what roles did they have?

Section 3 - Process, Tools

1. What [was/is/would be/will be] your process of making [project] or individual packets reproducible?
 - a. When did you start?
 - b. Who were the people and roles involved in this effort?
 - i. How did they communicate with each other? (*e.g., mailing lists, conferences/Bug Squashing Parties, issues, calls...*)
 - ii. Were there detractors?
 - c. What is your estimate of your time invested into it?
 - i. Did that change over time?
 - d. Where did you start?
 - i. What was the strategy for choosing which packages to work on first?
(*e.g., easy leaf packages first vs. important upstream dependencies first*)

1. a. i. What is your build process like?
 - A. Do you use some form of CI or other build infrastructure?
 - B. What is the level of version pinning that you do for releases?
 1. Upstream compiler version subreleases?
 2. Transitive dependencies?
- ii. How do you interact with upstream projects used in [project]?
 - A. Was there active community building,
 - B. communication,
 - C. buy-in into reproducible builds,
 - D. any insights on successful / unsuccessful communication
 - E. (common) community requests [rb or upstream]?
- iii. How did you decide that a package is reproducible?

(prompt: 100% reproducible artifacts vs. specific test for reproducibility; explainable differences in certain data fields)

1. [for libraries] Do you (want to) use other programming languages in the library and if so, why?
 - a. Can you tell us about any interactions between programming language changes and reproducibility efforts? Did one of them impact the other?
2. What is your tech setup/specific tooling or other resources to help you make [project] reproducible?
(*e.g., prompt for diffoscope*)
 1. a. Please tell us about the tools or libraries you built or used to help make [project] reproducible?

- i. Did you integrate any of them into the CI or packaging utilities?

- A. How did that go?
- b. Were there any approaches, tools, that you abandoned on your way to make [project] reproducible?
 - i. *Example: Binary diffing died*
- c. What type of tool – whether it exists or not – would you have liked to have to help with making your builds reproducible, or checking for reproducibility?
- d. Were there any other resources you used?
(*e.g., documentation, knowledge bases, web-sites*)

Section 4 - Obstacles/Challenges, Facilitators

1. If there were any, please tell us about the obstacles involved in making [project] reproducible?
 - a. Organizational (buy-in from different developers, not being able to do systemic changes in different parts of the project)
 - b. Technical (hard dependencies on timestamps for identifiers etc.)
 - c. Dependencies (upstream not being willing to take patches, marking change requests as invalid/WorksForMe)
 - d. Of the differences between different programming languages and their communities, which influenced your effort in [project]'s reproducibility? And was that influence positive or negative? (bug reporting, error culture towards compiler changes, community responses to questions)
2. Which factors were particularly helpful in making your progress more manageable?
3. Did your target change due to difficulties in getting [project] to be reproducible?
(*e.g. any compromises*)

Section 5 - Generalization / Lessons Learned

1. “If you had to start over, what would you do differently?”
 - a. Would using current tools solve a lot of the problems you encountered?
 - b. Are there any programming language specific things you wish you could have used?
2. What worked well, what didn't work?
 - a. Can you tell us about the role of upstream patches and how they help or hinder reproducibility?
 - i. How did your effort change over time while other packages worked on their reproducibility?
3. How specific would you say your process was to your project? We are interested in what can be generalized or already benefits other projects, and what's specific to yours.

- a. In your personal opinion, do you think that other projects could follow the same or similar procedures? (direct port of technical measures or organizational structures, similarities between biggest hurdles)
4. What would you recommend to other developers and projects?

Outro

1. Is there anything else you would like to tell us about reproducible builds, within or outside the scope of [project]?
2. Is there something that we did not cover during the interview but you would like to talk about?

Debrief

- **SWITCH OFF RECORDING** “The recording is now off”
- Could you recommend other projects or persons we could invite for an interview? They should have attempted to make their project reproducible.
- Thank the participant again for their valuable time
- Do you want to get a preprint of our paper?
 - If interested: what is your preferred contact data?
 - * We will be in contact for a preprint

Motivational Matrix

	Time	Money	Reputation	Results	
Research Group	Caching		Scientific Reproduction		Introspection
University	Minimizing manual retesting	Deduplication		Cheaper Builds	
Development Corporation				Ability to build in the future	
Security Organization	Build debug	Smaller binary differences			
Open Source Project	Increased development speed		OpenSSF scorecard		
End User			Quality	Chain of security	
Government					

Figure 1. Motivational matrix, joint work from a discussion session with attendees of the Reproducible Builds Summit 2022.