Interview Guide v2.5

Metadata	**Interview**
Created: 2021-08-10	Date:
Updated: 2021-10-17	Participant:
Language: **English**	Interviewer:
	Duration (est.):

Changelog

- v2.5 | Added some additional instructions for better replicability.
- v2.4 | Moved changelog to interview guide.
- v2.3 | Extended intro slightly with some additional motivation.
- v2.2 | Extended metadata entries with interview data.
- v2.1 | Extended intro with full list of institutions.
- v2.0 | Added German version.
- v1.2 | Added preamble and outro.
- v1.1 | Moved interview guide to Nextcloud.
- v1.0 | First interview-ready version.

[General Greeting of Participant]

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 of, I am going to talk about the context and data handling, and if you agree with everything, we would then start with the actual interview.

Context

- **We**: We are a researchers at the CISPA Helmholtz Center for Information Security (short intro).
- **Our research**: focuses on the area Usable Security for Developers (short overview).
- **In the past**: Open Source Projects as data source.
- Open Source code and projects are at the foundation of many software ecosystems. Open Source also has unique challenges, such as changing contributors and trust requirements.
- **Now**: "How can we empower open source contributors to build more secure projects."
- This interview as a start/exploration of internal processes and decisions often not visible at the repository level.
- For this interview:
    - We are not judging security or privacy 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, just say "next".
    - We are not just interested in structures, but also your personal opinions and experiences.
- **Questions?** Any questions about this interview context?

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 30 to 45 minutes.
- We will fully anonymize you and your projects in any publication and only include short quotes.
- We will send you a preprint before a potential publication, so you can veto quote usage etc.
- **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 answers (a few days)
    - The transcripts will be destroyed after analysis (a few months)
- **Questions?** Any more questions about data handling or recording?
- "The recording is now on"
- **Restate** consent question.

❒ ● S1 Project & Demographics

[Check project(s) beforehand]

Notes:____

____

____

____

- ☐ S1Q1 **Project:** Can you tell us a bit about the [project(s)] you are involved in?
  Follow-Ups:
  - ☐ S1Q1.1 **About:** What is the project about? What is its purpose?
  - ☐ S1Q1.2 **Age:** When was the project created?
  - ☐ S1Q1.3 **Contributors:** How many regular contributors does the project have?
  - ☐ S1Q1.4 **Connection:** How do contributors know each other? (Virtually, Personally)
  - ☐ S1Q1.5 **Distribution:** How are the contributors distributed geographically?
- ☐ S1Q2 **Project Relation:** How are you related to [project]?
  Follow-Ups:
  - ☐ S1Q2.1 **Join:** When did you join the project?
  - ☐ S1Q2.2 **Role:** What is your role in the project?

❒ ● S2 Incidents

[Mention “hypocrite commits“ incident: If participant is aware continue, else introduce (see Appendix)]

- ☐ S2Q1 **Opinion:** What do you think about this incident?
- ☐ S2Q2 **Challenges:** Can you remember any security challenges that your project faced in the past?
  Nudges:
  - malicious committers/commits,
  - security issues with your repository (software/provider)
  - tool chain, etc.?

❒ ● S3 Guidance

- ▢ S3Q1 **Guidance:** Are there guides/best practices/hints available for contributors, maintainers, etc.?
- [IF NOT Guidance]:
  - What are your thoughts about including guides/best practices/hints for contributors, maintainers, etc.?

Follow-Ups:

- ▢ S3Q1.1 **Infrastructure:** Does your project have security guidelines for configuring/running infrastructure? (cloud, VCS, etc.)
- ▢ S3Q1.2 **Languages:** Does your project use security or style guidelines for the utilized programming languages? What do they cover?
- ▢ S3Q1.3 **Cryptography**: If you're using cryptography in your code: Do you have a guide on how to use cryptography? (forbidden functions etc.)

❒ ● S4 Security Policies

- ▢ S4Q1 **Security Policies**: What do your security policies contain?
- [IF NOT Policies]:
  - What would you like your security policies to contain?

Follow-Ups:

- ▢ S4Q1.1 **Content**: What specific parts do they cover?
- ▢ S4Q1.2 **Applicability**: Do these have to be read and acknowledged by committers/contributors?

- ▢ S4Q2 **Disclosure Policies:** What is the (coordinated) disclosure policy of the project?
- ▢ S4Q3 **Security Incidents:** How are security incidents/issues handled?

Follow-Ups:

- ▢ S4Q3.1 **Policy:** By what policy?
- ▢ S4Q3.2 **Who:** By whom? (specific security team?)
- ▢ S4Q3.3 **Access:** Private/Public?
- ▢ S4Q3.4 **Process:** Are there "Playbooks" for Incident Response and Vulnerability Management? (What do these specify?)
- ▢ S4Q3.5 **History:** How was this process developed?

- ▢ S4Q4 **Security Testing / Reviews**: What measures do you have to test security?

Follow-Ups:

- ▢ S4Q4.1 **Aspects**: What aspects of security are checked and how?
- ▢ S4Q4.2 **Tools**: Is the project using specific (software) tools? (SAST, DAST, Manual, Pentests)
- ▢ S4Q4.3 **Project Stages:** At what stages of the project? (Only initially, on changes, etc.)
- ▢ S4Q4.4 **Frequency**: How often are manual security reviews done? Pentesting etc.? Who carries out these reviews? (skills, external/internal person)
- ▢ S4Q4.5 **Threat Modeling**: Is some form of Threat Modelling used?

❒ 🔵 S5 Project Structure

- ▢ S5Q1 **Repository**: What does the general repository structure look like? (file system, stages, C, etc.)
  Follow-Ups:
  - ▢ S5Q1.1 **Stages**: What stages do exist? (Code → Commit → PRs → Review → CI for tests and Build → Deployment, …)
  - ▢ S5Q1.2 **Control**: Who controls which stage?
  - ▢ S5Q1.3 **Main**: How are branches setup? Is it possible to directly push to the main branch?
  - ▢ S5Q1.4 **Systems**: How are Build and Deployment systems secured? Who has access/control?
  - ▢ S5Q1.5 **PRs**: How are incoming pull requests handled?
  - ▢ S5Q1.6 **Signed Commits**: Are commits signed? (Requires PGP key, how are those trusted?)
  - ▢ S5Q1.7 **Secret Management**: Does the project use some form of secret management system?
  - ▢ S5Q1.8 **Access**: Who has access to those systems?
- ▢ S5Q2 **Supply Chain**: What does the setup for the Supply Chain (e.g., libraries and other dependencies) look like?
  Follow-Ups:
  - ▢ S5Q2.1 **Criteria**: What criteria are considered when deciding on external libraries and dependencies?
  - ▢ S5Q2.2 **Checks**: How are the processes to check if those are secure and trusted?
  - ▢ S5Q2.3 **Vulnerabilities**: How is checked whether a dependency has security vulnerabilities? (SAST, code reviews, checking open source projects itself and its contributors, etc.)
  - ▢ S5Q2.4 **Signed**: Do libraries have to be cryptographically signed?
  - ▢ S5Q2.5 **Private Packages**: Does the project use a private package repository with vetted and secure dependencies?
- ▢ S5Q3 **Infrastructure**: Does the project have additional infrastructure such as a project website or chat tools?
  Follow-Ups:
  - ▢ S5Q3.1 **Access**: Who controls the additional infrastructure? Same set of maintainers for all infrastructure?

[We are now about halfway done]

❐ 🟠 S6 Release and Updates

- ▢ S6Q1 **Releases**: How are releases and updates published?
  Follow-Ups:
  - ▢ S6Q1.1 **Decision**: How is decided if/when a security update is released?
  - ▢ S6Q1.2 **Secured**: What security concepts are considered for releases? Are releases “secured” in any way?
  - ▢ S6Q1.3 **Update System**: Are security updates made automatically? How does the update system work?
  - ▢ S6Q1.4 **Deprecation**: Do you publish information about deprecated / insecure versions?

❐ 🟡 S7 Roles and Responsibilities

- ▢ S7Q1 **Contributors**: Can you tell us a bit about the maintainer / contributor hierarchy of the project?
  Follow-Ups:
  - ▢ S7Q1.1 **Security Roles**: Are there security specific roles in your projects?
  - ▢ S7Q1.2 **Roles known**: Are these groups/roles common knowledge?

❐ 🟢 S8 Trusting Contributors

- ▢ S8Q1 **Trust**: Can you tell us a bit about the trust model of the project?
  Follow-Ups:
  - ▢ S8Q1.1 **Establish**: How do you establish trust for new committers?
  - ▢ S8Q1.2 **Identity**: Do you have some form of identity check? (e.g., by being coworkers, at conferences, etc.)
  - ▢ S8Q1.3 **Authentication**: Do you authenticate committers? (How?)
  - ▢ S8Q1.4 **Access Control**: Are you using some form of access control in your project?
  - ▢ S8Q1.5 **Trusted**: How could a new contributor become trusted members of the project / team?
- ▢ S8Q2 **License** Is there a Contributor License Agreement?
  - ▢ **Yes**: What does it look like?
  - ▢ **No**: Do you know why the project does not have one?
- ▢ S8Q3 **Public:** Does the project maintain a public list of contributors and their contributions?

❐ ● S9 Untrustworthy Contributors

- ▢ S9Q1 **Trust Incidents**: Are you aware of any contributors that turned out to be not trustworthy?
- [IF NO]:
    - Assuming there is an untrustworthy contributor ...

  Follow-Ups:
    - ▢ S9Q1.1 **Approach**: How did the project deal with such a situation?
    - ▢ S9Q1.2 **Excluding**: If applicable, can you explain the process for excluding contributors from the project?
    - ▢ S9Q1.3 **Identifying**: How does the process for identifying untrustworthy contributors look like?
- ▢ S9Q2 **Trust Strategy**: What is the project's strategy to deal with contributors who become untrustworthy?

  Follow-Ups:
    - ▢ S9Q2.1 **Who**: Who makes the decision? (BDFL, committee, maintainers)
    - ▢ S9Q2.2 **Playbook**: Is there a playbook/defined process?
    - ▢ S9Q2.3 **Circumstances**: Under which circumstances are untrustworthy committers excluded from future contributions?
    - ▢ S9Q2.4 **Process**: What does the exclusion process look like?
    - ▢ S9Q2.5 **Investigation**: Does the project have a defined process to investigate potential vulnerable contributions?
- ▢ S9Q3 **Removal**: Did the project decide to remove their contributions?

  Follow-Ups:
    - ▢ S9Q3.1 **Decision**: How was this decision made?
    - ▢ S9Q3.2 **Process**: What did the removal process look like?

[We are now at the last question block]

❐ ● S10 Problems and Improvements

- ▢ S10Q1 **Reputation**: What is, in your personal opinion, the reputation of the project in terms of security and trust?

  Follow-Ups:
    - ▢ S10Q1.1 **Internal**: Internal reputation.
    - ▢ S10Q1.2 **External:** External reputation.
- ▢ S10Q2 **Improvements Project**: Assuming no limitations whatsoever (e.g., monetary or in terms of people-power), how would you personally like to improve security and trust in the project?

  Follow-Ups:
    - ▢ S10Q2.1 **Problems**: Where do you see problems in the current system?
    - ▢ S10Q2.2 **Why Exist:** What do you think are the reasons for the current (trust) system to be the way it is?
    - ▢ S10Q2.3 **Improvements:** What would you like to improve?

Outro

– “The recording is now off”
– Thank the participant again for their valuable time
– We will be in contact for a preprint

Debrief

– Is there something that we did not cover during the interview but you would like to talk about?
– Do you know of any other unique projects or persons we could invite for an interview?

Appendix

Incident: Hypocrite Commits

In April 2021, after receiving “poor quality patches” Linux kernel developer Greg Kroah-Hartman reverted 68 patches submitted by University of Minnesota email addresses and announced that all further patches coming from University of Minnesota addresses should be summarily rejected by default.

This incident actually started in August 2020, when a number of “bad faith” patches were sent to Linux kernel developers by University of Minnesota researchers under false identities. These patches were part of an ongoing work studying the feasibility of introducing vulnerabilities into open source software projects through minor patches (“hypocrite commits”).