Collaborative Security: Securing Open Source Software

Dr. Nicko van Someren
Chief Technology Officer
The Linux Foundation
Open Source Software has had it’s fair share of major security issues
Security Is Hard For Open or Closed Source - These Are Complex Systems
FOSS Security Is Different Though

FOSS is not more or less secure, but it *is* different

- Typically there are many more people contributing
- Sometimes (often?) there is a culture of “code is more important than specification”
- Processes are often more ad hoc
- There may be less market pressure to put security first
Linus’s Law: “Given enough eyeballs, all bugs are shallow.”
But what if you don’t have enough eyeballs?
The CII aims to substantially improve security outcomes in the FOSS projects that underpin the Internet.

The CII funds work in security engineering, security architecture, tooling, testing and training on key FOSS projects, as well as supporting general development on security-specific projects (such as crypto libraries).

The CII is a project run by the Linux Foundation, a 501(c)6 non-profit corporation.
The Linux Foundation created the Core Infrastructure Initiative with support from 19 Industry Giants.
What can we do to improve the security of Open Source Software?

We can do all the same things as we do when building commercial software.

The big difference is that we have to do it collaboratively, without having a top-down mandate demanding it.
Security is a process, not a product

- Think about security early. Think about security often.
- This requires buy-in from the whole project community

**Fostering a culture of security within your open source project is the single most important thing that you can do to improve your security outcomes**

- Security needs to be given equal weight with scalability, performance, usability and all the other design factors that matter to your users
Applying “Best Practice” to FOSS

- There are a great many widely known and widely used techniques that have been shown to improve security outcomes
  - Use them!

**FOSS Tool:** [https://bestpractices.coreinfrastructure.org](https://bestpractices.coreinfrastructure.org)

An easy-to-use check-list tool for assessing your project’s security posture
Security design

▪ Build a threat model and keep it up to date
  • Threat modeling doesn’t need to be hard or complex
  • **Tool**: Elevation of Privilege Threat Modeling Card Game
▪ Don’t use weak crypto
  • And definitely don’t try to design your own crypto!
▪ Know your dependencies
  • Fix known broken things
Change control

- Tracking who proposed changes, who reviewed those changes and who released them is critical to security.
  - This is often more complex in collaborative OS projects
  - As it happens, failures here are how Heartbleed made it into OpenSSL
- As soon as your project has two more people coding you need a policy for how code will get reviewed
Change control

▪ Use a version-controlled source repository
  FOSS Tool: git, Mercurial, bazaar
▪ Make code publicly visible between major releases
  • Public code review before final release is valuable
▪ Change logs are a must
  • If other people rely on your code, you can break their security by changing things in your code
Quality testing

Not all bugs represent vulnerabilities

... but all vulnerabilities are bugs, and...

It’s often very hard to tell the difference (at least until someone publishes an exploit!)
Quality testing

- Writing comprehensive tests is far less fun than writing new code to solve new and interesting problems
- But it’s a hell of a lot more fun than dealing with bugs after they get released
- Measure your test coverage and require collaborators to write tests for all contributed code

FOSS **Tools**: gcov (C/C++), CodeCover (Java), CodeCoverage (Python), and many more...
Security analysis tools

▪ Fancy commercial static analysis tools are expensive
  ... Switching all of your compiler warnings on is not!
▪ Use linters, code complexity checkers, fuzzers and other analysis tools where you can; they all can help
▪ The earlier in the project you start using these the less you will have to deal with “low signal to noise ratio”

FOSS Tools: SonarCube, FramaC, AFL & many more
Bug reporting: Closing the SDL loop

- Bugs happen; you need a process for dealing with them
- Users need a way to report security vulnerabilities that doesn’t broadcast them to the whole world!
- Take reports of security vulnerabilities seriously
  - Just because you can’t work out how to exploit a bug doesn’t mean that it can’t be exploited

**Tools:** Bugzilla, Trac, GitHub Issues
What can users of Open Source Software do to make themselves safer?

▪ There are many things that consumers of Open Source Software can do to make their deployments safer.
▪ In general these come down to two key areas:
  • Know your dependencies
  • Don’t freeload
Know your dependencies

- There are many ways that you might be running FOSS:
  - You bought a supported distribution
  - You downloaded a FOSS application
  - You bought a commercial application which uses a FOSS component internally
  - One of your developers pulled a library from GitHub and embedded it in your infrastructure
  - ... the list goes on.
Know your dependencies

- In order to ensure that the FOSS that you use is safe you need to know:
  - What open source components you are using?
  - What versions you are currently running, and where?
  - How these components can be updated, where do you get the update, what do you need to do to install them?
- Only when you have all this can you consider monitoring the vulnerability announcements
Know your dependencies

- Supported distributions make their business by solving these problems, but only for the bits that they supply.
- For commercial software, demand that your vendor disclose the FOSS components in their applications.
- For internal apps, there are commercial scanning tools.
  - In many cases proactive scanning with `fgrep –i copyright` can be a useful first step.
- Remember, version numbers are important.
Don’t Freeload!
Don’t freeload!

“Maybe part of the reason that the Heartbleed bug had such a big impact is that more than $1 trillion per year of business was being run on a project that had received only $3,000 in support in the preceding year”
Don’t freeload!

- Open Source projects rely on more than cash donations.
  - They rely on a vibrant community forming around them.
- You can support projects by providing:
  - Detailed feedback and analysis of bugs you find
  - Patches, both to fix bugs and add capabilities
  - When you build a business around a project, engineers
  - Engagement with the community to crowd-source support and help share knowledge
Projects must think about security early & often and they must be willing to prioritise it as highly as other features.

Most of the ways that we can make open source software more secure are common industry “best practices”. It is simply a matter of choosing to adopt them.

For users, the most important factor is to track what FOSS projects you use in your infrastructure.

Contribute! Your feedback, time, engineers, and money!
Thank you.

https://www.coreinfrastructure.org