ETSI/IQC Quantum Safe Cryptography Conference 2026

Best Open Source Practices in (Post-Quantum) Cryptography: *Lessons Learned from the Linux Foundation, and More*

Presented by:

Hart Montgomery

Talk Outline

Open Source Background

- Open source background
- The Linux Foundation, or why you should care about my opinion

PQ Crypto @ Linux Foundation

- Our projects
- OQS, PQCP, CBOMkit

Basic Open Source Best Practices

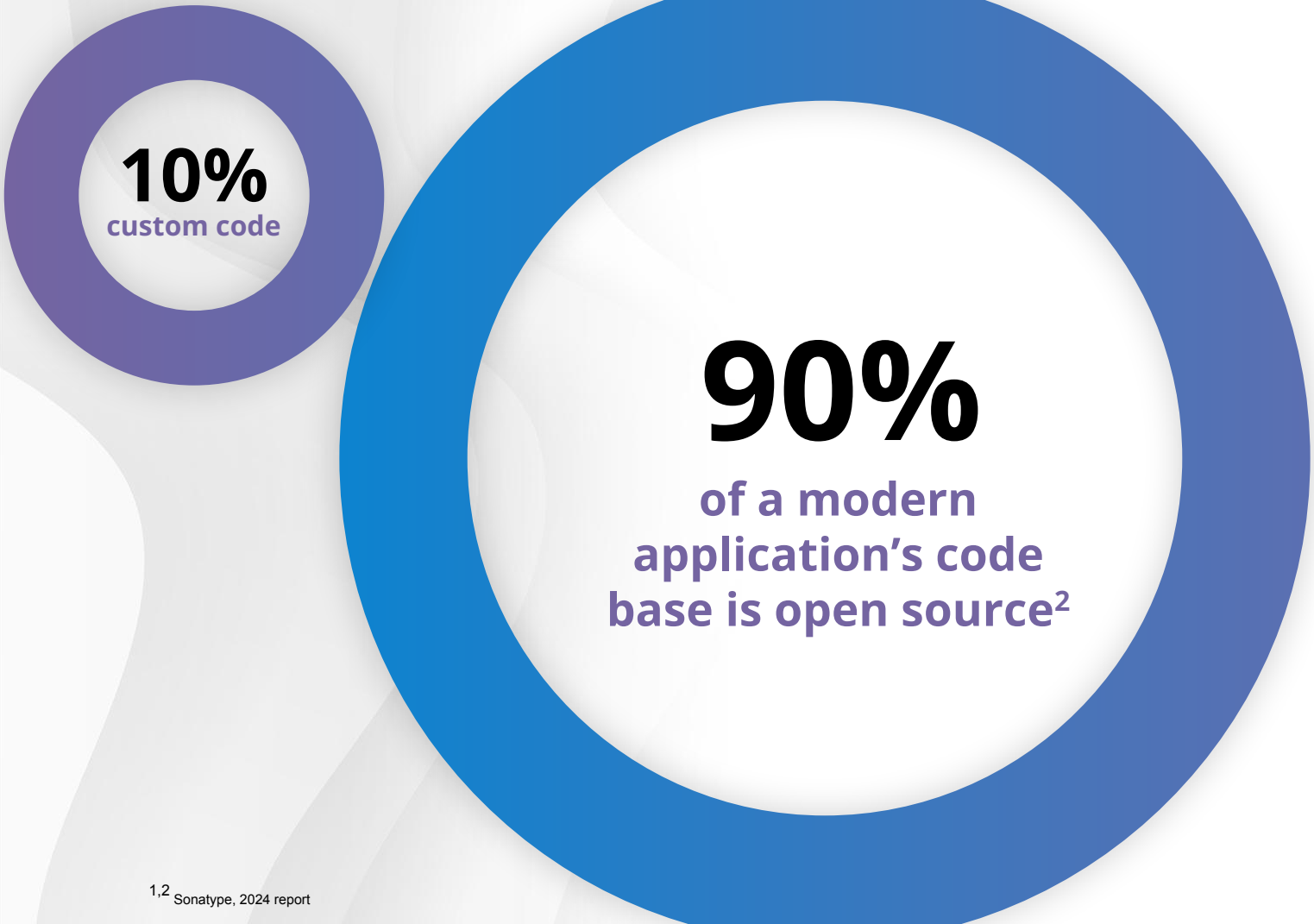
- Open source software licensing
- IP protections for software
- Basic best security practices

Usual Disclaimer: I'm not a lawyer, this is not legal advice, and opinions are my own.

Perception: Open Source is Cypherpunk

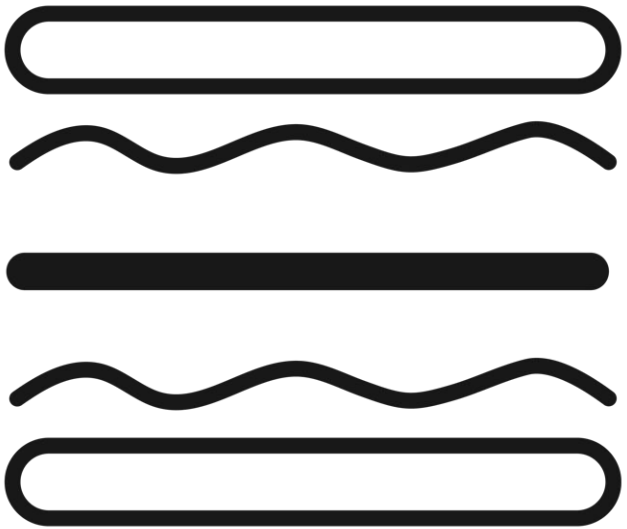
“Cypherpunks write code. We know that someone has to write software to defend privacy, and since we can't get privacy unless we all do, we're going to write it. We publish our code so that our fellow Cypherpunks may practice and play with it. Our code is free for all to use, worldwide. We don't much care if you don't approve of the software we write. We know that software can't be destroyed and that a widely dispersed system can't be shut down.”

–Eric Hughes, “A Cypherpunk’s Manifesto”



[1,2] Sonatype, 2024 report

Building a Modern Platform or Application

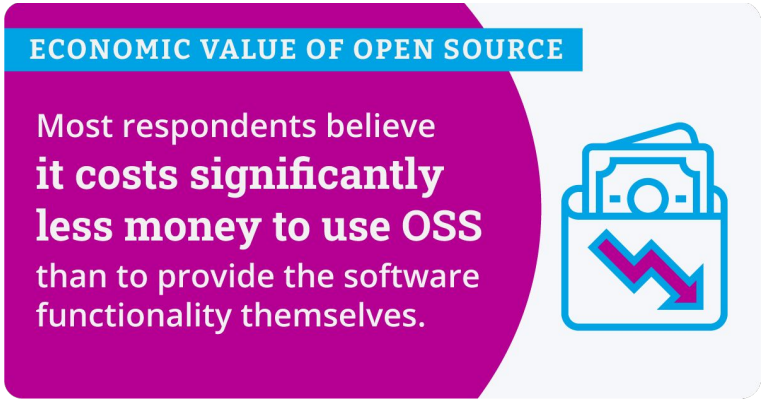


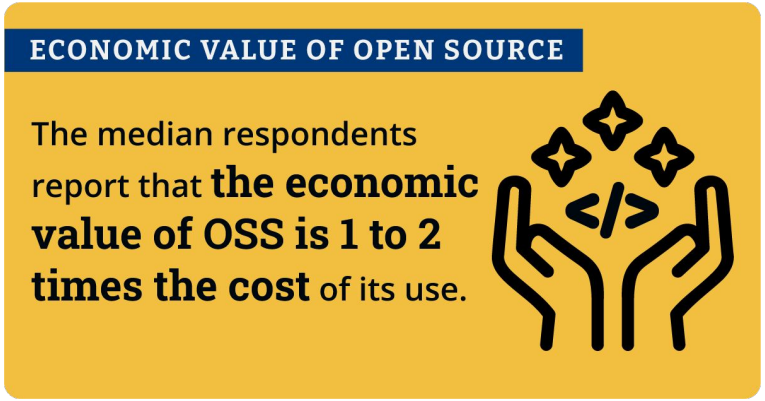
Use Open Source
Libraries to Solve Problems
Open Source Code (~70%)

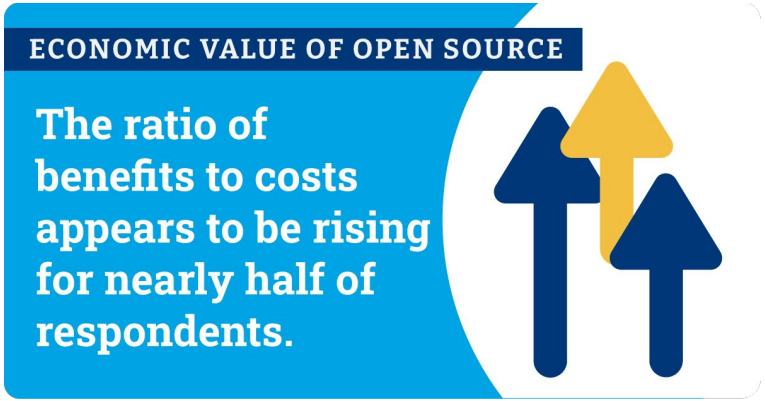
Write Custom Code
Custom Code (~10%)

Choose a Framework
Open Source Code (~20%)

Economic Value of Open Source



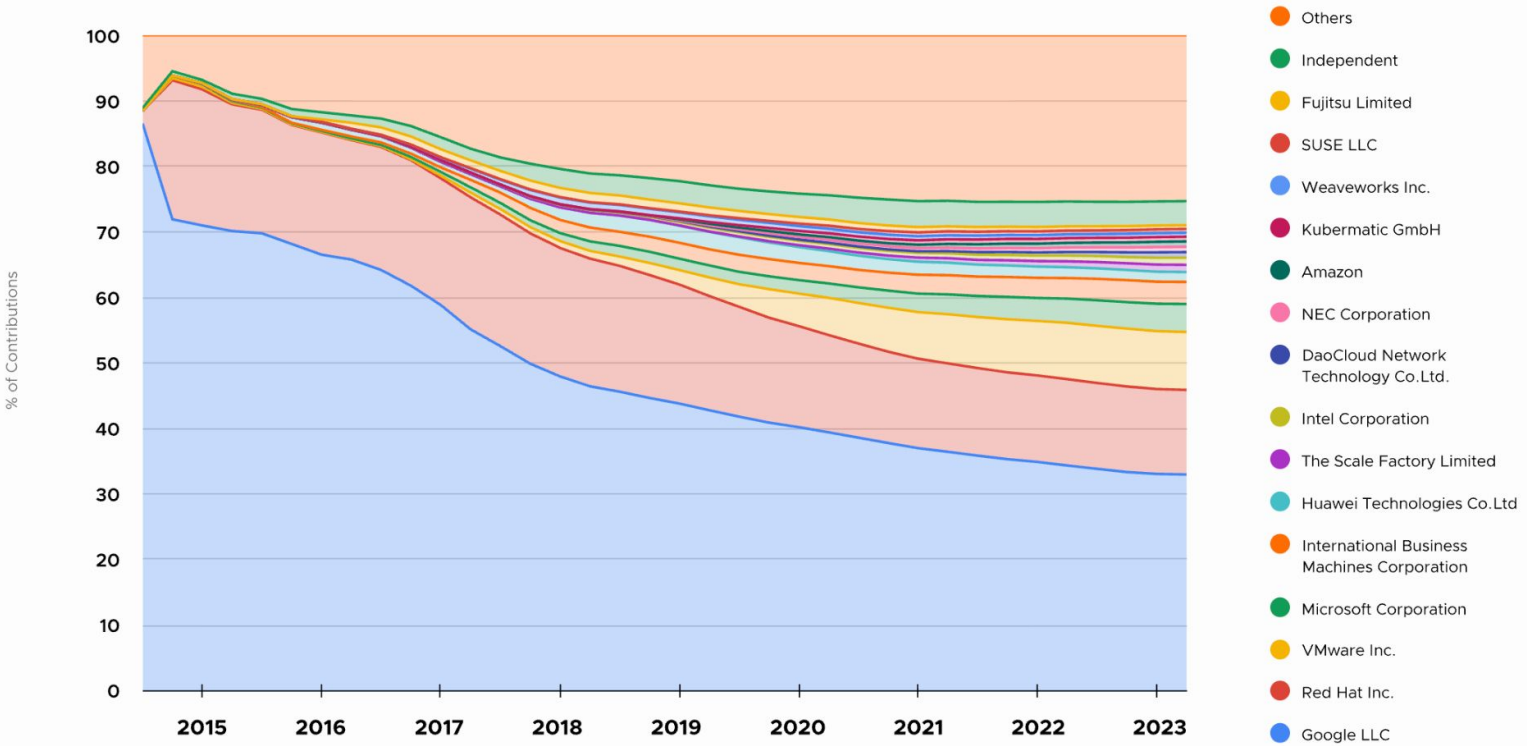




Source: [Economic Value of Open Source]

The Power of Communities

Percentage Breakdown of Kubernetes Contributions by company Q2 2014 – Q2 2023



The Linux Foundation solves **collaboration** for open source code.

When multiple companies, entities, or individuals want to collaborate on open source software but don't trust one single party to own the code, they turn to the Linux Foundation.

THE LINUX FOUNDATION

The Linux Fo l development.

When multiple com want to collaborate on open source softwa party to own the code, th tion.

THE LINUX F

Across our Foundations:

110M
Lines of Code Added Weekly

14.4M
Lines of Code Removed Weekly

722,330
Technical Contributors

118,556
Ecosystem Contributors

12,925
Contributing Companies

12,899
Repositories

12.2M
Commits

1.4M
Pull Requests

2.1M
Builds Monitored

933,150
Logged Issues

11.2B
Container Downloads

236,499
Vulnerabilities Detected

4.4M
Email Messages

5.1M
Chat Messages

32,517
CLA Contributors

82,939
Event Attendees (12 months)

1100+
Project Domains

28,028
Community Meetings

Some of Our Projects

Some of Our Projects

Post-Quantum Cryptography Alliance

Mission: to advance the adoption of post-quantum cryptography, by producing high-assurance software implementations of standardized algorithms, and supporting the continued development and standardization of new post-quantum algorithms with software for evaluation and prototyping.

Open Quantum Safe (OQS)

Goal: Provide an open-source software for evaluating and using post-quantum cryptography.

Main Components:

- **liboqs:** A cryptographic library in C providing implementations of standards-track post-quantum signature schemes and public key encryption / KEM schemes, plus support for testing and evaluating new experimental PQ algorithms.
- **OQS Provider:** An OpenSSL 3 provider adding post-quantum algorithms from liboqs into OpenSSL 3-based applications, providing support for post-quantum and hybrid TLS, X.509, and S/MIME/CMS.
- **OQS Demos:** Provide prototype integrations of post-quantum algorithms into protocols and applications, to support experiments and interoperability by early adopters and assist in protocol-level standardization.

Github: https://github.com/open-quantum-safe

Website: https://openquantumsafe.org/

TSC Page: https://github.com/open-quantum-safe/tsc/blob/main/README.md#members

Post-Quantum Code Package (PQCP)

Goal: Develop and maintain high-assurance, production implementations of NIST standards for a variety of target architectures (ARMv7, ARMv8, x86_64, ...) and languages (C, Rust, Go, ...), distributed primarily as source code. Aim for implementations to be audited and/or **formally verified**.

Intended Audience: Implementers of cryptographic libraries and tools that need to add NIST standard primitives to their software as source code. The code packages should be organized in a way that allows for easy tracking of changes and integration into software development lifecycles.

Deployments: PQCP is being actively deployed by many large companies and institutions, including in aws-lc (the main cryptography library of AWS)

Website: https://docs.pqcodepackage.org/main/

TSC Page: https://github.com/pq-code-package/tsc/blob/main/README.md

CBOMkit

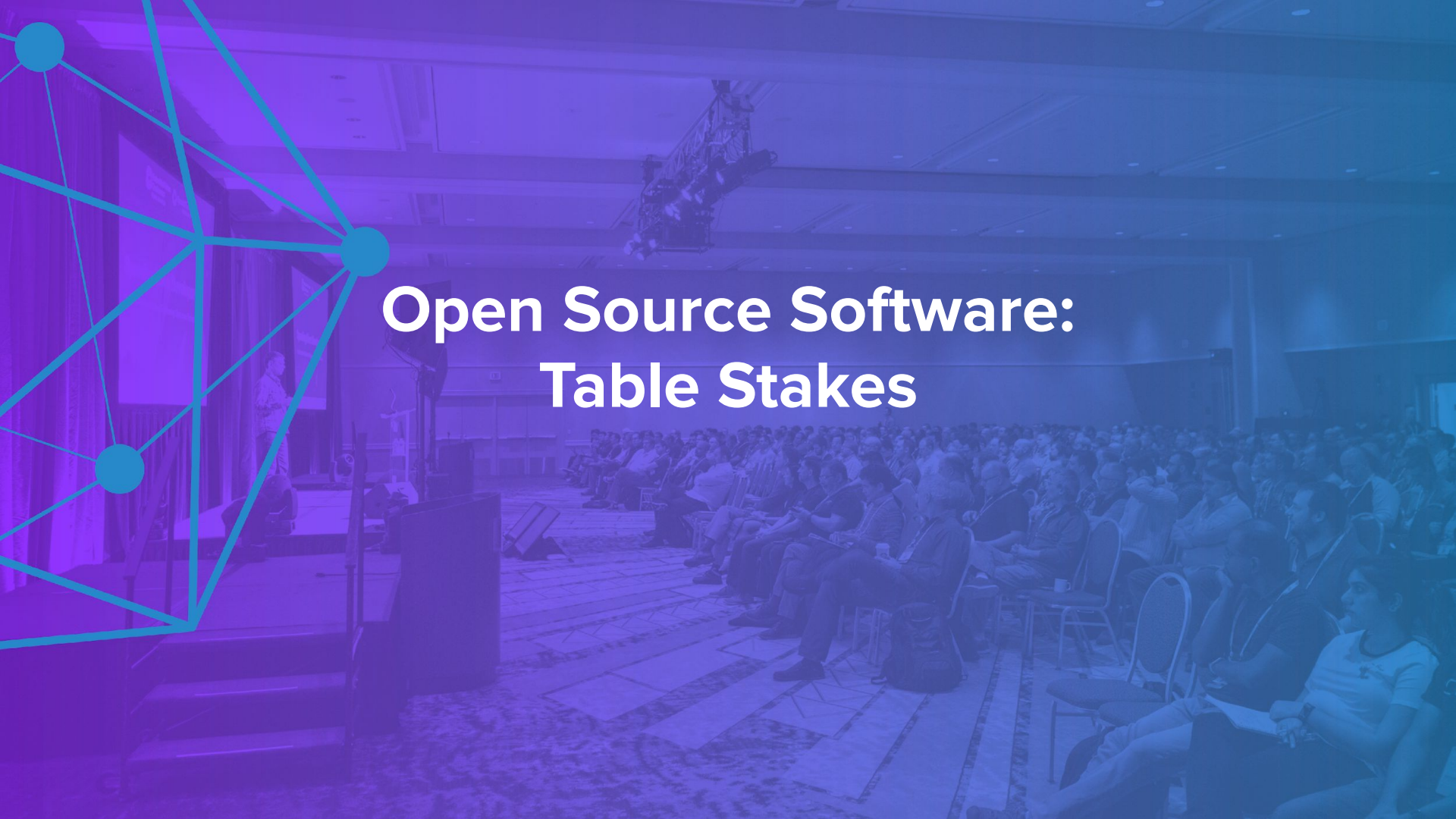
CBOMkit is a **toolset for dealing with Cryptography Bill of Materials (CBOM)**.

Main Components:

- CBOM Generation (CBOMkit-hyperion, CBOMkit-theia): Generate CBOMs from source code by scanning private and public git repositories to find the used cryptography.
- CBOM Viewer (CBOMkit-coeus): Visualize a generated or uploaded CBOM and access comprehensive statistics.
- CBOM Compliance Check: Evaluate CBOMs created or uploaded against specified compliance policies and receive detailed compliance status reports.
- CBOM Database: Collect and store CBOMs into the database and expose this data through a RESTful API.

Github: https://github.com/cbomkit

TSC Page: https://github.com/cbomkit/tsc


Open Source Software:
Table Stakes

What Is Open Source Software?

Open source software (OSS) is a type of computer software in which source code is released under a **license** in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose. Open source software may be developed in a collaborative public manner.

License is a legal document that records and formally expresses a set of legally enforceable acts, processes, or contractual duties, obligations, or rights related to the software and/or source code.

Business Source License (BSL)

Technically a source-available license, **not an open source license**. The source code is **public but use is limited to certain users**.

After a certain amount of time the **code converts to a standard open source license**.

Viewed as a **compromise between proprietary software and open source**. Treat software under this license as **proprietary**, but with the extra added benefit that you (and others) can review the source code.

Notable examples: BUSL
Notable Projects: MariaDB MaxScale, Hashicorp Terraform, Arbitrum Nitro

Business Source License (BSL)

Technically a source-available license, **not an open source li**

code is **public b**
certain users.

After a certain a
converts to a st
license.

Viewed as a **compromise between and open source**.

s license as
e extra added
ners) can review

SL
DB MaxScale,
bitrum Nitro

Use a BSL license when:

- **You want to be able to control who uses your code (like it were closed source)**
- **...but you want people to be able to independently verify and study your code (like academic work).**
- **NOTE: The LF DOES NOT accept BSL-licensed code, and pro-OSS organizations don't like it either.**

Copyleft License

Open source licenses that **require users to make available all derivative works**. In other words, if you modify the code and use it in something, you typically have to make the source of that "something" available for free.

Very hard to use commercially–often you must release to the public code that you wish to keep private!

Notable Examples: GPL, MPL, LGPL
Notable Projects: Linux Kernel, Mozilla Firefox, Geth

"GNU is not in the public domain. Everyone will be permitted to modify and redistribute GNU, but no distributor will be allowed to restrict its further redistribution. That is to say, proprietary modifications will not be allowed. I want to make sure that all versions of GNU remain free."

–Richard Stallman, GNU Manifesto

Copyleft License

Open sou…ly–often
to make …ode that
In other w…
use it in s…
make the… LGPL
available … Mozilla

Copyleft goal: increase contributions back to the codebase from users by legally requiring them.

Unfortunate reality: people just won't use your code if there are any viable alternatives due to the potentially cumbersome legal requirements.

It is VERY hard to relicense to a permissive license.

"GNU is not in the public domain. Everyone will be permitted to modify and redistribute GNU, but no distributor will be allowed to restrict its further redistribution. That is to say, proprietary modifications will not be allowed. I want to make sure that all versions of GNU remain free."

–Richard Stallman, GNU Manifesto

Permissive License

An open source license that lets you **modify and use the code freely**. Essentially **“do whatever you want”** with few extra restrictions.

Most commercially used open source code and most Linux Foundation codebases are licensed with a permissive license. It’s **by far the easiest kind of code to use**.

We recommend using code with a permissive license **if you want others to use your code** because it is by far the easiest to use without any legal repercussions.

Notable examples: Apache 2.0, MIT License
Notable projects: Chromium, Kubernetes, most LF projects, OpenSSL, Besu, **All PQC projects at the LF**

Permissive License

An open so… …ith a **modify a… …others to** Essentiall… … far the few extra…

Most com… … MIT code and… codebases are licensed with a permissive license. It's **by far the easiest kind of code to use**.

Notable projects: Chromium, Kubernetes, most LF projects, OpenSSL, Besu

Recommendation: default to permissive licenses for research code, or code you don't have plans to monetize.

Permissive licensing typically maximizes the number of people who can use your code.

Pay close attention to the licenses and licensing requirements of the open source projects that you use and to which you potentially contribute!

Recommendation: **use Apache 2.0**, which is a permissive license and has an explicit patent grants, for open source cryptography projects, particularly if you aren't trying to monetize your code.

Other Legal/IP and Copyright Protections

Problem: IP-Protected Code?

Contributors to open source software may unintentionally or maliciously **add code that has some form of IP protection** (like patents) to an open source project. Institutions are **very worried** about this kind of thing.

SCO–Linux disputes

From Wikipedia, the free encyclopedia

In a series of legal disputes between SCO Group and Linux vendors and users, SCO alleged that its license agreements with IBM meant that source code IBM wrote and donated to be incorporated into Linux was added in violation of SCO's contractual rights. Members of the Linux community disagreed with SCO's claims; IBM, Novell, and Red Hat filed claims against SCO.

On August 10, 2007, a federal district court judge in *SCO v. Novell* ruled on summary judgment that Novell, not the SCO Group, was the rightful owner of the copyrights covering the Unix operating system. The court also ruled that "SCO is obligated to recognize Novell's waiver of SCO's claims against IBM and Sequent". After the ruling, Novell announced they had no interest in suing people over Unix and stated "We don't believe there is Unix in Linux".[1][2][3][4] The final district court ruling, on November 20, 2008, affirmed the summary judgment, and added interest payments and a constructive trust.[5]

SCO–Linux disputes

Overview

Timeline · SCO–SGI code dispute of 2003 · SCOsource

Litigation

SCO v. IBM · *SCO v. AutoZone* · *SCO v. DaimlerChrysler* · *SCO v. Novell* · *Red Hat v. SCO*

Companies involved

SCO Group · IBM · Novell

Individuals involved

Ralph Yarro III · Pamela Jones · Darl McBride

Other

Project Monterey · United Linux · *USL v. BSDi* · Groklaw · Xinuos

V · T · E

On August 24, 2009, the U.S. Court of Appeals for the Tenth Circuit partially reversed the district court judgment. The appeals court remanded back to trial on the issues of copyright ownership and Novell's contractual waiver rights. The court upheld the $2,547,817 award granted to Novell for the 2003 Sun agreement.[6]

On March 30, 2010, following a jury trial, Novell, and not The SCO Group, was unanimously found to be the owner of the UNIX and UnixWare copyrights.[7] The SCO Group, through bankruptcy trustee Edward Cahn, decided to continue the lawsuit against IBM for causing a decline in SCO revenues.[8]

On March 1, 2016, SCO's lawsuit against IBM was dismissed with prejudice; SCO filed an appeal later that month.[9]

Protect Yourself from IP Trolls!

We recommend that your project either use the **Developer Certificate of Origin (DCO)** or a **Contributor License Agreement (CLA)**.

It is important to have appropriate legal frameworks around your codebase so that users of the code cannot get frivolously sued! **This is especially important if you allow contributions from people outside of a single company**.

Developer Certificate of Origin
Version 1.1

1 Letterman Drive
Suite D4700
San Francisco, CA, 94129

Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or

(b) The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

(c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

(d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

Developer Certificate of Origin
Version 1.1

1 Letterman Drive
Suite D4700
San Francisco, CA, 94129

Developer's

By making a

(a) The cont en source license indic

(b) The cont n appropriate cations, whether cre to submit under a differ

(c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

(d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

That's it!

DCO is very easy to set up on Github using commits signed with the "-s" flag.

Be sure to have legal protections for your code if you have any ambitions for it beyond research!
If you don't, **people will (justifiably) be less likely to use your code!**

We require this for all LF projects, and should should too for your OSS!

We like to use the **DCO at the Linux Foundation** because we find it has less contributor friction than a CLA, but a CLA (e.g. the Apache CLA) is a perfectly adequate solution too!


Open Source Security

Security Can Be Different in OSS!

Placing a priority on security is essential for any open source project–ESPECIALLY CRYPTOGRAPHY!

But you already knew that.

Users need to be able to verify that best practices are being followed.

Some things to make sure are visible and open:

- **Vulnerability disclosures and a security reporting pipeline** are very important and different from closed-source software.
- A **software bill of materials (SBOM)** is essential for making sure your software is secure when building with and in open source. **CBOMs** can be used for cryptography!
- **Authenticating your software and artifacts** is important for users.

Open Source Security Foundation

The Open Source Security Foundation (OpenSSF) is a community of software developers, security engineers, and more under the Linux Foundation who are working together to secure open source software for the greater public good.

The OpenSSF has answers and guidelines for all of the things you need to be doing for basic open source software security. Check it out for more details!

Security Vulnerability Disclosures/Pipelines

An advantage of open source software is that community members can find and report bugs. **You want to make this as easy as possible for them**. Contributor friction for bug reporting may cause you to miss a big bug!

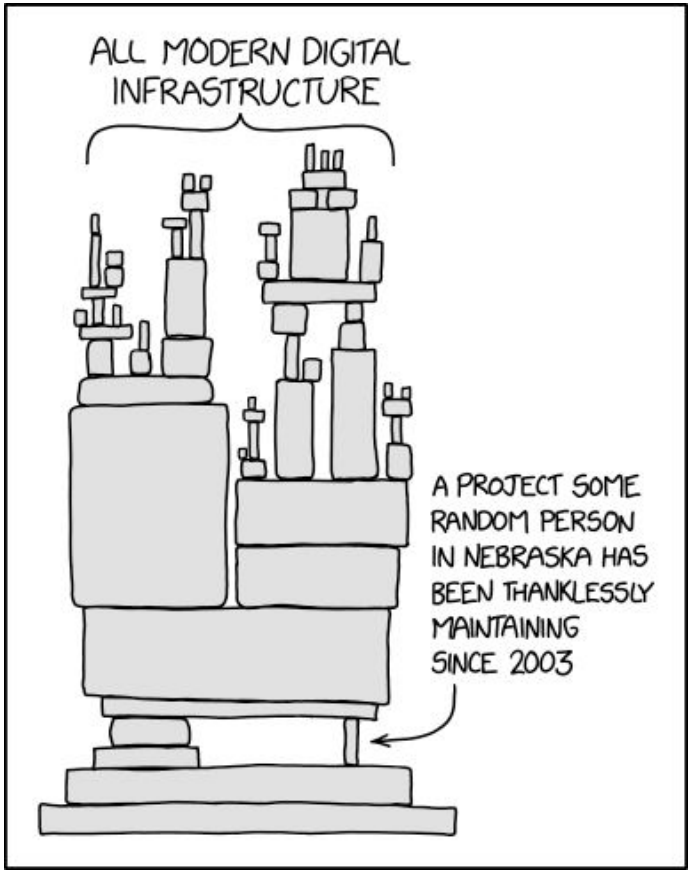
Form a security response team that can quickly respond to and handle bugs.

It doesn't matter too much the exact method you prefer for reporting, as long as it is **easy to find and follow**.

Recently in the Linux Foundation, many of our projects have been using the **Github tooling for security bugs**.

On the other hand, Besu has 7 reporting channels!

Software Supply Chain Attacks



https://xkcd.com/2347

Software Supply Chain Attacks

XZ Utils

Github Actions Attack
(No Logo!)

Software Supply Chain

≃ 1,000 codebases across 16 industries

86% of commercial codebases contain open source vulnerabilities

81% contain high or critical-risk vulnerabilities

90% of codebases contain OSS components more than four years out of date

Software Bill of Materials (SBOM)

Keep track of all of the code pulled in to an OSS project. If there are bugs you need to fix them and update quickly!

Carefully examine the code you do use to make sure it's **well-maintained**.

Well-run open source projects are **careful with the dependencies they use**!

There are a lot of good tools to help with this!

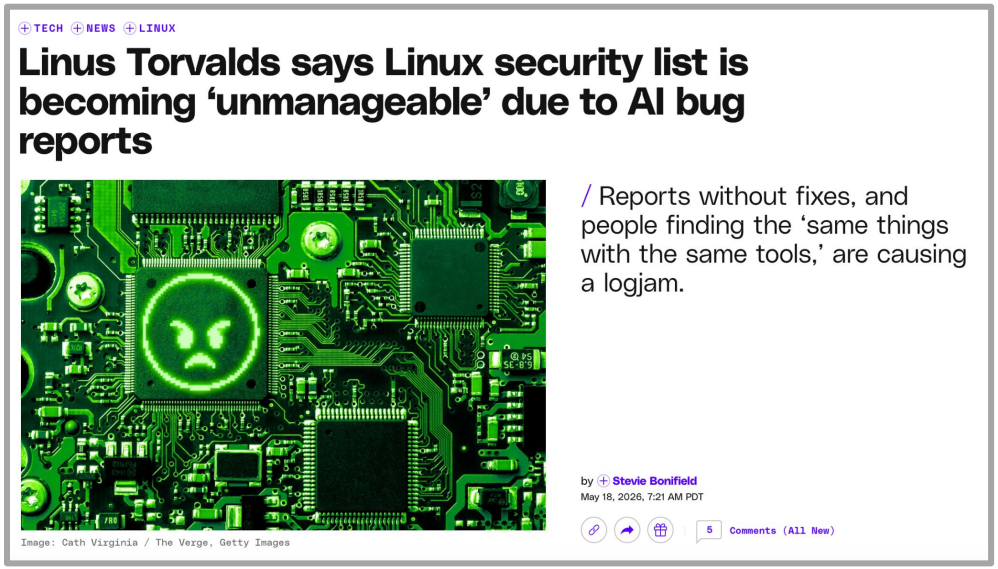
Software and Artifact Authentication

If you build a popular open source project, **people will attempt to impersonate you**!

This **constantly happens for Linux Foundation projects** (e.g. people attempt to create fake npm packages that look like those of official LF projects).

There are **plenty of tools** that you can use to sign and authenticate code; **use them!**

AI: Significantly Changing Things!



TECH NEWS LINUX

Linus Torvalds says Linux security list is becoming ‘unmanageable’ due to AI bug reports

/ Reports without fixes, and people finding the ‘same things with the same tools,’ are causing a logjam.

by Stevie Bonifield
May 18, 2026, 7:21 AM PDT

5 Comments (All New)

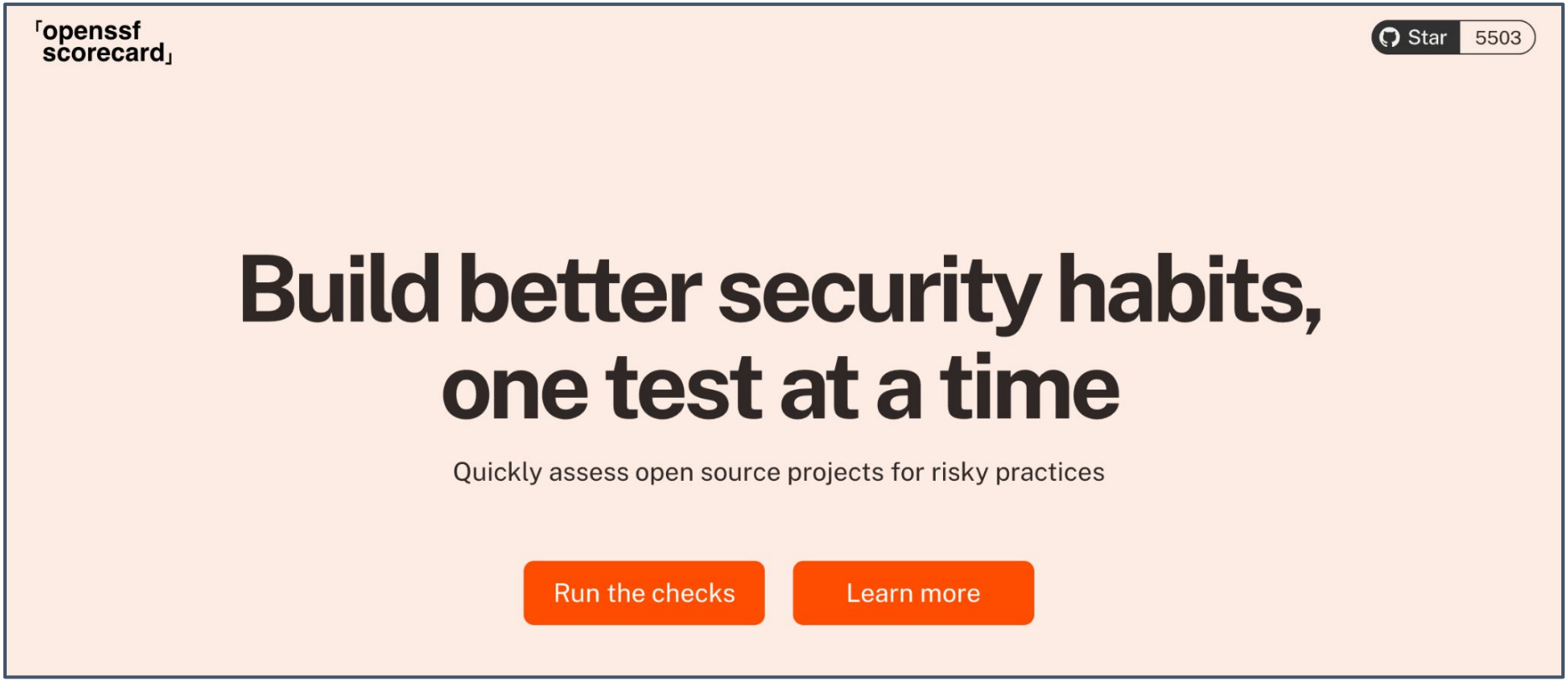
Image: Cath Virginia / The Verge, Getty Images

We recommend an AI contributor policy.

See the Linux kernel example for inspiration.

Consider making AI-reported security bugs public by default. This can help make managing them easier and isn’t likely to result in new attacks (since enough people will have access to finding them anyways).

Verifiability: ScoreCard



Any Questions?

hmontgomery@linuxfoundation.org