Open Source, Long Term: What Makes It Safer, and What It Still Demands of You

Open source software is now part of the foundation beneath modern business.

It powers websites, cloud platforms, databases, security tools, development environments, artificial intelligence systems, and business applications. Many organizations depend on open source every day, whether they realize it or not.

That reliance can create a long-term advantage. Open source software is often more transparent, more adaptable, and less dependent on a single vendor than closed proprietary software. It can remain viable long after a commercial product changes direction, raises prices, or reaches end of life.

But open source is not automatically secure. It is not automatically maintained. It is not automatically compliant.

Open source provides visibility and flexibility. Your organization still provides governance.

Open Source Can Reduce Long-Term Software Risk

In our previous article, Why Outdated Proprietary Software Is a Risk You Can't Afford to Keep Running, we examined the dangers of continuing to rely on closed software that no longer receives meaningful support.

Open source can reduce several of those risks because the code, licensing, and development process are more visible. If one commercial provider abandons an open source project,

another company, community, or internal team can continue maintaining it. Organizations can also migrate, fork, or contract with a different support provider without being completely trapped by one vendor’s roadmap.

That flexibility matters.

A long-term software strategy should not depend entirely on whether one company continues selling, supporting, or pricing a product in a way that works for your business.

Why Open Source Is Often Safer

Open source does not eliminate security risk. It creates conditions that can make risk easier to identify, evaluate, and manage.

1. Independent Review Creates Visibility

With proprietary software, customers generally cannot inspect the underlying source code. They depend on the vendor to identify vulnerabilities, explain the security design, and provide patches.

Open source allows qualified developers, researchers, security teams, and organizations to examine the code independently. That transparency supports more thorough security reviews and makes it easier to understand how a system handles authentication, logging, data protection, and access controls.

More visibility does not guarantee more security. “Many eyes” only helps when people are actually reviewing the project. Still, the ability to inspect and test the code is a meaningful advantage.

It also supports compliance and data privacy reviews. Your team can investigate how software processes sensitive information instead of relying entirely on a vendor’s assurances.

2. Multiple Maintainers Can Reduce Vendor Dependence

A proprietary product typically has one commercial owner. If that owner discontinues the product, changes its licensing model, or stops supporting your version, your options may be limited.

Open source projects can have multiple maintainers, contributors, vendors, and user communities. If one organization steps away, others can continue development or create a maintained fork.

That does not make every project sustainable. It does mean your organization has more paths forward.

You can migrate to another supported distribution, hire a qualified provider, contribute fixes internally, or move to a commercial service built around the same technology. This reduces long-term lock-in and makes it easier to hand off the system when your IT needs change.

3. Open Standards Improve Integration

Open source tools often work well with modern security platforms and operational systems. They may support established logging formats, application programming interfaces, identity providers, multifactor authentication, monitoring platforms, and vulnerability scanners.

That flexibility helps your team build a connected security program instead of maintaining isolated tools.

For example, a business may integrate an open source application with centralized authentication, security information and event management, endpoint monitoring, automated backups, and patch management. Better integration improves visibility and helps your team identify unusual activity sooner.

Open source can also support a broader AI Implementation strategy by giving your organization greater control over deployment environments, data flows, integrations, and model-supporting infrastructure.

4. Transparent Licensing Can Support Better Planning

Open source is not the same as “free software with no obligations.” Every project has a license, and that license determines how the software can be used, modified, and distributed.

When properly documented, open source licensing gives your organization more predictable options than a proprietary product that can change pricing or usage terms at the vendor’s discretion.

The Open Source Initiative’s FAQ explains the difference between permissive licenses and copyleft licenses. Some

licenses allow broad commercial use with limited obligations. Others may require modified versions or distributed applications to remain available under the same license.

Understanding those terms before deployment helps you avoid legal, compliance, and operational surprises.

Open Source Still Has Serious Weaknesses

A balanced security strategy must account for the risks as well as the advantages.

Maintainer Burnout and Abandonment

Some widely used projects depend on one or two volunteers. Those maintainers may be responsible for critical code while receiving little or no financial support.

If they become unavailable, the project may stop receiving patches. A project can remain popular and functional while quietly becoming a security liability.

Do not evaluate open source software only by its current popularity. Review its release cadence, number of active maintainers, security policy, issue response, documentation, and recent vulnerability history.

Supply Chain Attacks

Open source dependencies can introduce supply chain risk through malicious packages, typosquatting, dependency confusion, compromised maintainer accounts, and tampered build processes.

A trusted package today can become a serious problem after a maintainer account is breached or a malicious update is published.

Your organization should obtain software from trusted repositories, verify packages where possible, restrict build permissions, use multifactor authentication, and monitor dependencies continuously. Automated scanning and controlled internal repositories can reduce exposure.

Patch Lag and Downstream Versions

A vulnerability may be fixed in an upstream project while your operating system distribution, application vendor, or internal deployment continues using an older version.

This creates patch lag. Your team must know which version is installed, where it came from, who is responsible for updating it, and whether the update has been tested.

A Software Bill of Materials helps create that visibility. CISA describes an SBOM as an inventory of the components within a software product. It can help your team quickly determine whether a system is affected by a newly discovered vulnerability.

No Automatic Vendor SLA

Many open source projects do not provide a guaranteed response time, dedicated support desk, or contractual service-level agreement.

If your business depends on the software, the responsibility for patching, upgrading, troubleshooting, and Incident Response may fall on your team unless you purchase commercial support.

That support should be part of your planning. The purchase price may be low, but the operational responsibility is not always low.

Open Source Is Not Free to Run With No Investment

Open source developers need an incentive to continue maintaining the code that businesses rely on.

Companies often build revenue-producing products and services on top of open source software while contributing nothing back. This creates a free-rider problem. The business benefits from the code, but the developer or project community absorbs the cost of maintenance, infrastructure, security fixes, documentation, and user support.

That model is not sustainable indefinitely.

If open source software is important to your operations, support the people and organizations maintaining it. Practical options include:

1. **Provide financial support.** Donate or sponsor maintainers through platforms such as GitHub Sponsors, Open Collective, or Patreon. Support relevant foundations and projects directly.

2. **Purchase commercial support.** Pay for an enterprise edition, managed service, support contract, or consulting relationship when one is available. This gives your organization accountability while helping fund continued development.

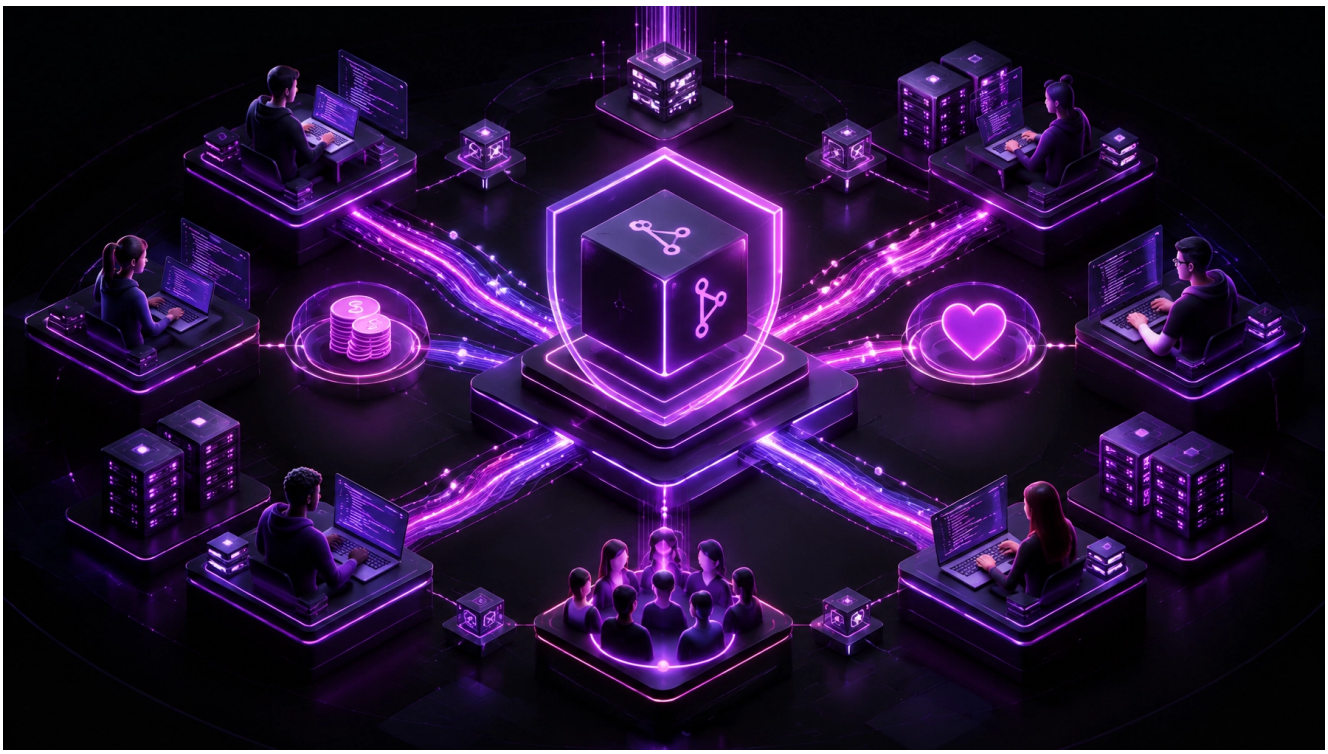
3. **Contribute engineering resources.** Submit code, documentation, testing, translations, bug reports, or security improvements. Contributions reduce the burden on maintainers and improve the project for everyone.

4. **Fund security work.** Sponsor independent audits, penetration testing, secure build systems, code signing, and vulnerability remediation for software your

organization depends on.

5. **Hire or contract maintainers.** If a project is critical to your business, employing the people who understand it best can protect both the software and your operational continuity.
6. **Join a foundation or consortium.** Shared stewardship gives businesses a structured way to help guide, fund, and secure important projects.

Free to use does not mean free to sustain. Funding maintenance is often far less expensive than recovering from an incident caused by an abandoned or compromised dependency.



Build an Open Source Policy Before You Build Dependency Risk

Your organization should treat open source as part of its IT management and cybersecurity program.

Start with five practical actions:

1. **Inventory your software.** Track direct and indirect dependencies, versions, sources, licenses, and business owners.
2. **Generate and maintain an SBOM.** Keep the inventory current as software is patched, upgraded, or deployed in new environments.
3. **Assess project health.** Review maintainer activity, release cadence, security reporting, documentation, governance, and available support.
4. **Automate monitoring and patching.** Use dependency scanning, vulnerability alerts, update testing, and defined remediation timelines.
5. **Budget for stewardship.** Include commercial support, sponsorships, security audits, and internal maintenance in the total cost of ownership.

This approach allows you to use open source strategically instead of treating it as an unmanaged collection of free downloads.

Choose Transparency, Then Manage It Decisively

Open source is often a strong long-term alternative to outdated proprietary software. It can improve transparency, reduce lock-in, support integration, and provide more options when a commercial vendor changes direction.

But open source is not a security shortcut.

You must evaluate the project, track its dependencies, comply with its license, secure your software supply chain, and support the developers who keep it viable. When your organization depends on open source, contributing back is not

just a goodwill gesture. It is a practical investment in continuity and risk reduction.

DarkBox Security Systems helps organizations assess software portfolios, strengthen IT management, improve supply chain visibility, and align technology decisions with cybersecurity and data privacy requirements. We provide proactive monitoring, secure system design, compliance-driven consulting, and practical recommendations built around your environment.

Review your open source dependencies before they become an operational blind spot. Contact DarkBox Security Systems to assess your software portfolio and build a stronger long-term security strategy.