



# OpenSSF

OPEN SOURCE SECURITY FOUNDATION

## Building Strong Foundations for a More Secure Future:

Addressing The Systemic Issues in the Software Supply Chain that Led to Log4Shell

**Brian Behlendorf, OpenSSF GM**

**February 2023**

# Open source is critical to the software supply chain

**97%**

percent of audited commercial codebases contain OSS

**78%**

percent of code in codebases is OSS

**85%**

percent of codebases contain open source that is more than four years out of date

There has been an astonishing

**742%**

average annual increase in Software Supply Chain attacks over the past 3 years.

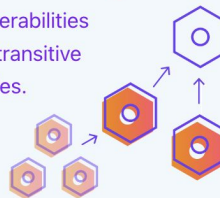


*Key Finding*

About

**6** out of every **7**

project vulnerabilities come from transitive dependencies.



*Key Finding*

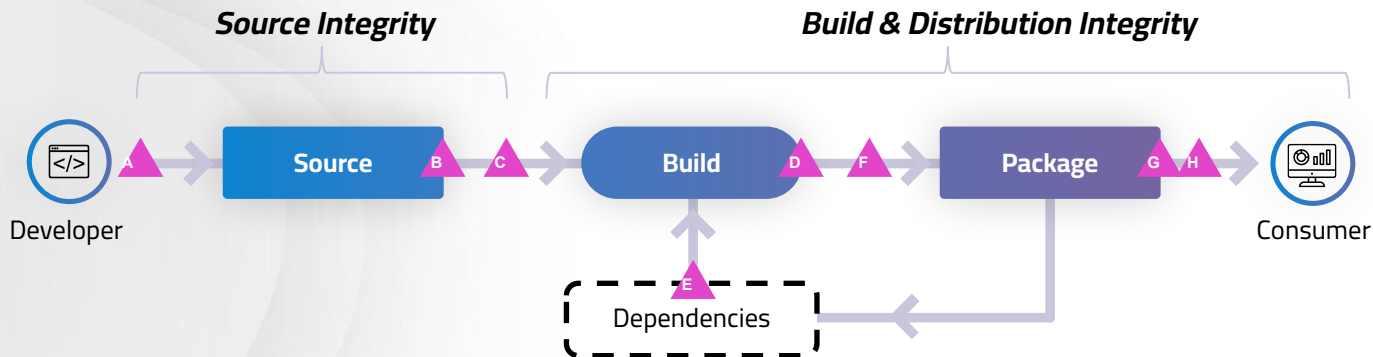
Source:

[Synopsys2022] "2022 Open Source Security and Risk Analysis Report" by Synopsys  
<https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/rep-ossra-2022.pdf>

[Sonatype2022] "2022 State of the Software Supply Chain" by Sonatype  
<https://www.sonatype.com/state-of-the-software-supply-chain/introduction>

[Sonatype2022]

# Securing Software: Make it secure AND secure its supply chain



**A** Bypassed code review

**B** Compromised source control system

**C** Modified code after source control

**D** Compromised build platform

**E** Using a bad dependency

**F** Bypassed CI/CD

**G** Compromised package repo

**H** Using a bad package



**OpenSSF**  
OPEN SOURCE SECURITY FOUNDATION

# Log4Shell Timeline (aka Everyone's 2021 Winter Holiday)

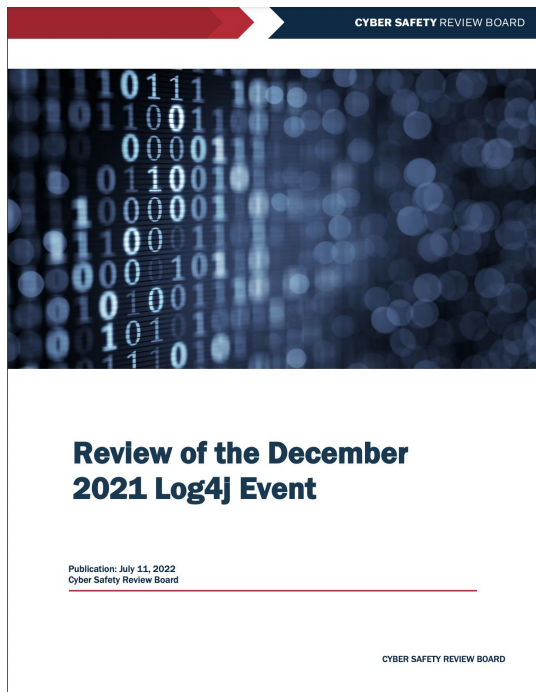
- Nov 24, 2021: Security researcher Chen Zhaojun discovers the now infamous **CVE-2021-44228**, or “Log4Shell,” vulnerability that allows unauthenticated attackers to execute remote code on vulnerable systems, scoring a CVSS of 10 out of a possible 10.
- Dec 6: Apache Log4j releases version 2.15.0 to remediate the vulnerability. Shortly after, **CVE-2021-45046** was discovered (a flaw that eventually netted a CVSS of 9.0/10) after further research led to the discovery that this vulnerability allowed for remote code execution by an attacker.
- Dec 10: UK NCSC issues Log4j warning to UK organizations
- Dec 11: CISA director comments on “**urgent challenge to network defenders**”
- Dec 13: Version 2.16.0 of Apache Log4j is released to remediate. Yet another vulnerability is discovered **CVE-2021-45105**, a CVSS 5.9/10 denial of service vulnerability due to infinite recursion in lookup evaluation.
- Dec 19: The Log4j team releases version 2.17.0 to fix the denial of service vulnerability
- Dec 20: Log4j exploited to install Dridex and Meterpreter
- Dec 22: Data shows 10% of all assets vulnerable to Log4Shell
- Dec 28: Yet another patch is released, version 2.17.1, this time to remediate **CVE-2021-44832**, a CVSS 6.6/10 that allows code execution by attackers with permissions to modify the logging configuration file.
- Jan 4, 2022: FTC tells companies to patch Log4j vulnerability, threatens legal action
- Jan 10: Microsoft warns of China-based ransomware operator exploiting Log4Shell

# Log4Shell Raises Some Serious Questions

- Is open source software's generally good reputation... actually well deserved?
- Does this demonstrate deep and pervasive technical issues with how we all consume and develop open source code?
- Do these issues extend to the sustainability model for open source code? Can we really depend upon so many "volunteers"?

Now it's not just devs, CTOs, and CISOs asking these questions — it's compliance and risk officers, it's the cybersecurity insurance industry, it's the European Union, the White House's National Security Council, the FTC, UK's NCSC and many other governments and agencies.

# Cyber Safety Review Board Report on Log4J Event



- [Final Report of the Cyber Safety Review Board \(CSRB\) - July 11 2022](#)
- Report concluded that the Log4j vulnerability could have been prevented if Log4j developers had access to:
  - Training in secure coding practices consistent with established secure development lifecycle tools and techniques
  - Security-oriented design reviews
  - Threat models
  - Security audits

# Lessons Learned

- CSRB Report concludes a focused review could have identified the unintended functionality, but such security resources for a review were not available at the time of adding the JNDI support in 2013
- The Log4j vulnerability could have been prevented if Log4j developers had access to:
  - Training in secure coding practices consistent with established secure development lifecycle tools and techniques
  - Security-oriented design reviews
  - Threat models
  - Security audits

# Reducing Risk to the Ecosystem

- CSRB also found that the only way to reduce the likelihood of risk to the ecosystem caused by vulnerabilities in Log4j, and other widely used open source software is to ensure that code is developed pursuant to industry-recognized secure coding practices and an attendant audit by security experts
- The volunteer-based open source community would need sustained financial support and expertise to make this possible

# Log4Shell, one year later (Brian's Take)

- Log4Shell is not exclusively a supply chain security story or a vulnerability disclosure story, but rather a hybrid of both. This reflects the ways different kinds of weaknesses combine to create deep threats.
- But it became a supply chain story because of how difficult it was to detect and report on the presence of unpatched Log4j in the wild.
- The vulnerability disclosure issue — ensuring that when vulnerabilities are found, the appropriate patches are distributed in a timely manner to minimize exploitation — is also critical to address in order to prevent Log4Shell-like events in the future.

# Log4Shell, one year later (Brian's Take)

- Log4Shell also demonstrates security imbalances in open source by a gap in the mapping of motivations: developers are often incidentally working on open source software such as Log4j in the pursuit of fixing bugs or adding features, which rarely leads them to invest extra time on functions that would reduce the risk of security issues in their code.
- Just like closed source developers, OSS developers often need to fight with their managers for the right to work on security improvements, from adding extra tests and adopting other security best practices, to simply paying down technical debt and removing underused features. Hard to measure the “ROI” of security work, so it gets de-prioritized.

# Log4Shell, one year later (Brian's Take)

- \$50k-\$100k third-party audit likely would have found all four CVEs. Add \$50k-\$100k for the fixes and coordinated disclosure process.
- \$200k: more than most groups of developers can spend on their own project; but far, far less than the impact on society from a single Log4Shell.
- Can we find the next likely critical weakness and spend \$200k to prevent the next one? No.
- Can we find, say, 200 projects each with a ~1% chance of being the next one? **YES.**
- Is it worth \$40M to scan and remediate 200 OSS projects per year? **YES.**



# OSS Software Security: A Theory of Change

- How do you bring about culture change in security practices?
  - Answer: **Carrots, then Defaults, then Sticks**
- For example, encrypting the web with SSL/TLS:
  - Carrots: the green key in browser location bar. Early adopters like banking websites, e-commerce requirements.
  - Defaults: Let's Encrypt
  - Sticks: non-TLS websites generated warnings, then blocks
- OSS Do-ocracy: everything is done by those who show up. Mandates, even by Foundations themselves, can often backfire.
- The EU Cyber Resilience Act is the backlash to "Move Fast and Break Things"
  - But OSS developers are not Mark Zuckerberg!
  - We need Carrots and Defaults, long before we need the Sticks

# *The OpenSSF Has Entered The Chat*

Established by the Linux Foundation in 2020, the OpenSSF is a **global initiative** securing investment, resources, and expertise to measure and improve the security of open source software (OSS) and the software supply chain.

It brings together cybersecurity and open source software leaders building an array of different technology initiatives:

- **Open Source Security Software**
- **Open Specifications**
- **Open Education Resources**

...and other products and activities that **build cybersecurity capacity** and **reduce global cybersecurity risk**.



**OpenSSF**  
OPEN SOURCE SECURITY FOUNDATION

# OpenSSF Working Groups & Projects

## Best Practices

*Identification, awareness, and education of security best practices*

- [OpenSSF Best Practices badge](#)
- [Scorecards](#)
- [Great MFA distribution SIG](#)
- [Common Requirements Enumeration \(CRE\)\\*](#)
- [Secure Software Development Fundamentals courses SIG](#)
- [Security Knowledge Framework \(SKF\)\\*](#)

## Vulnerability Disclosures

*Efficient vulnerability reporting and remediation*

- [Guide to coordinated vulnerability disclosure for OSS projects](#)
- [Vulnerability Disclosures Whitepaper](#)
- [osv-schema](#)

## End Users WG

*Voice of public & private sector orgs that primarily consume open source*

## Identifying Security Threats

*Security metrics/reviews for open source projects*

- [security-reviews](#),
- [Project-Security-Metrics \(dashboard\)](#)
- [SECURITY-IMPACTS.yml spec](#)

## Security Tooling

*State of the art, globally accessible security tools*

- [ossf-cve-benchmark](#)
- [Web Application Definition spec](#)
- [fuzz-introspector](#)

## Securing Software Repositories

*collaboration of repositories & tools to improve security*

- Coming soon!

## Supply Chain Integrity

*Ensuring the provenance of open source code*

- [Supply-chain Levels for Software Artifacts \(SLSA\) \[repo\]](#)

## Securing Critical Projects

*Identification of critical open source projects*

- [criticality score](#)
- [Harvard research](#)
- [package-feeds](#) / [package-analysis](#)
- [allstar](#)

## Associated Projects

- [Project Alpha-Omega](#)
- [Project Sigstore](#)
- GNU Toolchain Infrastructure (GTI) support

# Are You an OSS Maintainer? Use These!

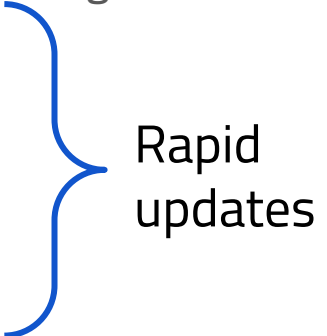
# Concise Guide for Evaluating Open Source Software

1. Can you avoid adding it?
2. Are you evaluating the intended version?
3. Is it maintained?
4. Is there evidence that its developers work to make it secure? ["Developing"]
5. Is it easy to use securely?
6. Are there instructions on how to report vulnerabilities?
7. Does it have significant use?
8. What is the software's license?
9. What is your evaluation of its code?

*... includes how to get  
information to estimate  
the answers*

<https://github.com/ossf/wg-best-practices-os-developers/blob/main/docs/Concise-Guide-for-Evaluating-Open-Source-Software.md#readme>

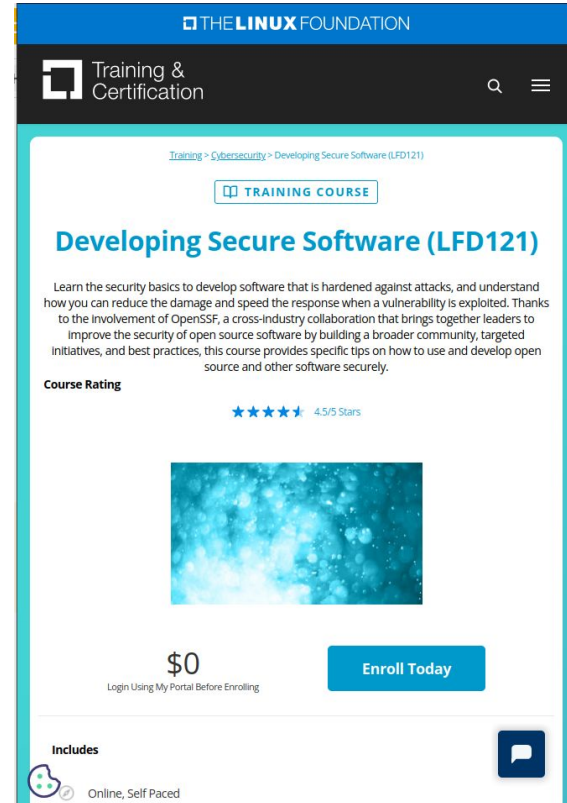
# Concise Guide for Developing More Secure Software

1. Ensure all privileged developers use multi-factor authentication (MFA) tokens.
  2. Learn about secure software development.
  3. Use a combination of tools in your CI pipeline to detect vulnerabilities.
  4. Evaluate software before selecting it as a direct dependency. ["Evaluating"]
  5. Use package managers.
  6. Implement automated tests [high coverage, negative testing].
  7. Monitor known vulnerabilities in your software's direct & indirect dependencies.
  8. Keep dependencies reasonably up-to-date.
  9. ... (more, e.g., OpenSSF Best Practices Badge & OpenSSF Scorecards)
- 
- Rapid updates

<https://github.com/ossf/wg-best-practices-os-developers/blob/main/docs/Concise-Guide-for-Developing-More-Secure-Software.md#readme>

# Course: Secure Software Development Fundamentals

- **Free** course, 14-18 hours, with 3 parts:
  - Requirements, Design, and Reuse
  - Implementation
  - Verification and More Specialized Topics
- Courses teach fundamentals of developing secure software (OSS or not)
- Free certificate via LF Training (evidence you learned the material)
- <https://openssf.org/training/courses/>



The screenshot shows the course page for "Developing Secure Software (LFD121)" on The Linux Foundation Training & Certification portal. The page has a blue header with the Linux Foundation logo and navigation links. The course title is prominently displayed in blue, followed by a description of the course's focus on security basics and open source software. A "Course Rating" section shows a 4.5/5 star rating. Below this is a placeholder image for the course. At the bottom, the price is listed as "\$0" with a note to login before enrolling, and an "Enroll Today" button is visible. The footer indicates the course is "Online, Self Paced" and includes a chat icon.

THE LINUX FOUNDATION

Training & Certification

Training > Cybersecurity > Developing Secure Software (LFD121)

TRAINING COURSE

## Developing Secure Software (LFD121)

Learn the security basics to develop software that is hardened against attacks, and understand how you can reduce the damage and speed the response when a vulnerability is exploited. Thanks to the involvement of OpenSSF, a cross-industry collaboration that brings together leaders to improve the security of open source software by building a broader community, targeted initiatives, and best practices, this course provides specific tips on how to use and develop open source and other software securely.

**Course Rating**

★★★★★ 4.5/5 Stars

**\$0**  
Login Using My Portal Before Enrolling

**Enroll Today**

**Includes**

Online, Self Paced

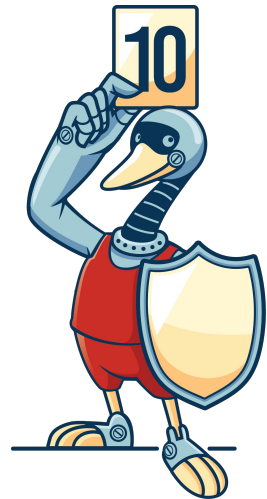
# OpenSSF Best Practices Badge



- Identifies best practices for OSS projects
  - Goal: Increase likelihood of better quality & security. E.g.:
    - "The project sites... MUST support HTTPS using TLS."
    - "The project MUST use at least one automated test suite..."
    - "At least one static code analysis tool MUST be applied..."
    - "The project MUST publish the process for reporting vulnerabilities on the project site."
  - Based on practices of well-run OSS projects
- If OSS project meets best practice criteria, it earns a badge
  - Enables projects & potential users know current status & where it can improve
  - Combination of self-certification, automated checks, spot checks, public accountability
  - Three badge levels: passing, silver, gold
- Participation widespread & continuing to grow
  - >5,000 participating projects, >850 passing+ projects in 2022-10
  - Current statistics: [https://bestpractices.coreinfrastructure.org/en/project\\_stats](https://bestpractices.coreinfrastructure.org/en/project_stats)
- A project within the OpenSSF Best Practices Working Group (WG)
- For more, see: <https://bestpractices.coreinfrastructure.org>

# OpenSSF Scorecards

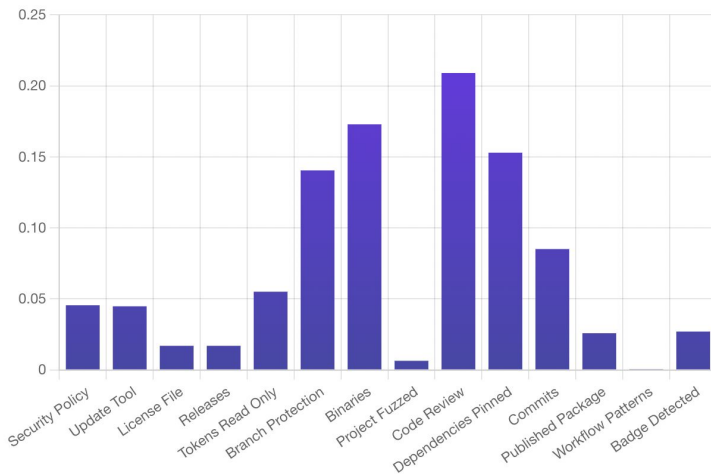
- *Automatically* scores OSS projects on heuristics ("checks")
  - Each related to security, scored 0-10, weighted average computed
  - Can use to evaluate your own or others' projects (they don't need to cooperate)
  - Currently only works on projects hosted on GitHub (not fundamental)
- Sample checks:
  - Binary-Artifacts - Is the project free of checked-in binaries?
  - Branch-Protection - Does it use Branch Protection ?\*
  - CI-Tests - Does it run tests in CI, e.g. GitHub Actions, Prow?
  - CII-Best-Practices - Does it have an OpenSSF (formerly CII) Best Practices Badge?
  - Code-Review - Does it require code review before code is merged?
  - Contributors - Does it have contributors from at least two different organizations?
- <https://github.com/ossf/scorecard>



# OpenSSF Scorecard Checks Actually Work.

## Project Quality Metrics

FIGURE 2.2. ELEMENTS MOST USEFUL FOR IDENTIFYING VULNERABLE PROJECTS

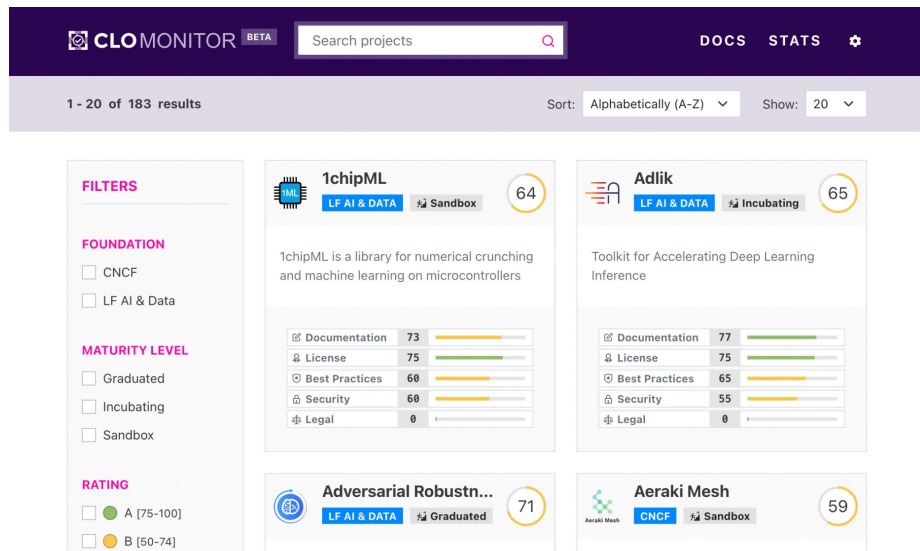


Source: Sonatype

<https://www.sonatype.com/state-of-the-software-supply-chain/project-quality-metrics>

# Scorecards in Usage

- CNCF's CLO Monitor uses the Scorecard API for its measurements
  - <https://clomonitor.io/>
- CNCF SecuritySlam
  - CNCF project maintainers worked over a month to raise their projects' security score (measured by OpenSSF Security Scorecard) ahead of KubeCon + CloudNativeCon NA to increase security awareness, posture & compliance
  - <https://community.cncf.io/cloud-native-security-slam/>



# Sigstore: Software Signing Service



- Tools currently exist to cryptographically sign OSS packages
  - No widely-practical mechanism to determine if public keys used are correct
  - No easy way to detect malicious signing
  - Key revocation typically impractical in practice
- Sigstore is a free-to-use non-profit software signing service
  - Users generate ephemeral short-lived key pairs using the sigstore client tooling
  - sigstore PKI service provides a signing certificate generated upon a successful OpenID connect grant
  - All certificates are recorded in certificate transparency log
  - Software signing materials are sent to a signature transparency log
  - Guarantees that claimed user controlled their identity service providers' account at time of signing
  - Once the signing operation is complete, the keys can be discarded, removing any need for further key management or need to revoke or rotate.
- Using OpenID connect identities enables use of existing security controls such as 2FA, OTP and hardware token generators
- Transparency logs are public and open; anyone can monitor transparency logs for issues

## How sigstore works

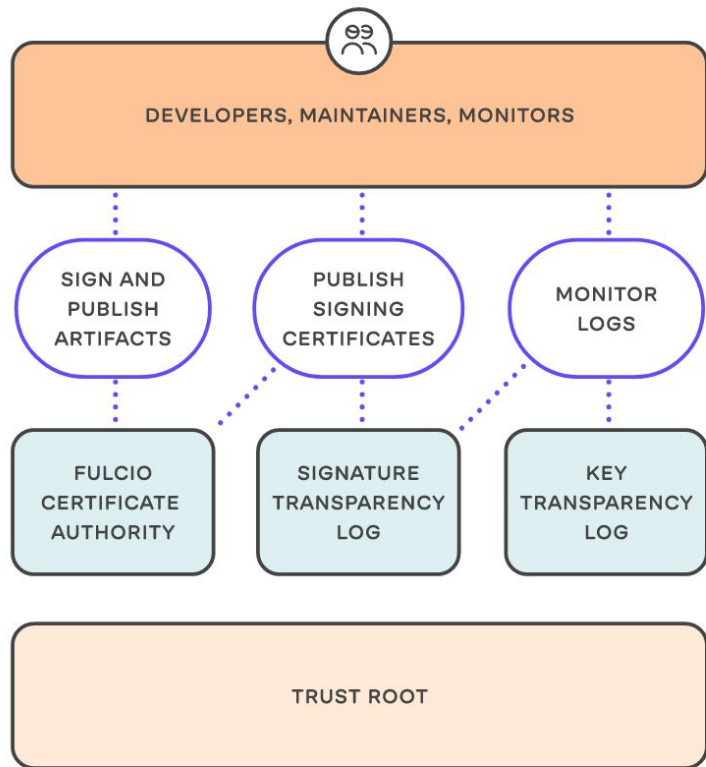
sigstore is a set of tools developers, software maintainers, package managers and security experts can benefit from. Bringing together free-to-use open source technologies like Fulcio, Cosign and Rekor, it handles digital signing, verification and checks for provenance needed to make it safer to distribute and use open source software.

### A standardized approach

This means that open source software uploaded for distribution has a stricter, more standardized way of checking who's been involved, that it hasn't been tampered with. There's no risk of key compromise, so third parties can't hijack a release and slip in something malicious.

### Building for future integrations

With the help of a working partnership that includes Google, the Linux Foundation, Red Hat and Purdue University, we're in constant collaboration to find new ways to improve the sigstore technology, to make it easy to adopt, integrate and become a long-lasting standard.



# Other OpenSSF Initiatives



Supply chain Levels for  
Software Artifacts (SLSA) -  
<https://slsa.dev/>



Guide to coordinated  
vulnerability disclosure for  
open source software projects  
[https://github.com/ossf/oss-  
vulnerability-guide#readme](https://github.com/ossf/oss-vulnerability-guide#readme)

# Measuring Risk: OSS Dashboard

- Provide metrics to help make decisions about adding/using some OSS
- Build on existing work
  - E.g., OpenSSF Scorecards, OpenSSF Best Practices Badge, LFX, CHAOSS, etc.
- Maps interestingly to recommendations from the CSRB report as well as from the Securing Open Source Software Act released in September 2022.
- Work in progress as part of Identifying Security Threats Working Group
- Eventually, this should be how companies measure their own risk in use of OSS
- Eventually, this is how developers should figure out which dependencies to use or eliminate, and what steps to take to improve their own score.

# Addressing OSS Security At Scale: The Alpha-Omega Project

## Alpha:

- Systematically build the maturity and “capacity” inside major open source software projects and foundations to prepare for and respond to security issues.
- Yearly ~\$500k grants with broad objectives:
  - Python Software Foundation
  - Rust Foundation
  - Eclipse Foundation
  - JQuery
  - NodeJS

## Omega:

- Systematically scan the top 10K OSS projects for new vulnerabilities, and then work with maintainers to get them addressed.
- 11 Vulnerabilities identified in 2022, setting up for hundreds to thousands in 2023.



# Trellix Advanced Research Center Patches 61,000 Vulnerable Open-Source Projects

By [Douglas McKee](#) · January 23, 2023

Late last year, the Trellix Advanced Research Center team uncovered a [vulnerability in Python's tarfile module](#). As we dug in, we realized this was CVE-2007-4559 – a 15-year-old path traversal vulnerability with potential to allow an attacker to overwrite arbitrary files. CVE-2007-4559 was reported to the Python project on 2007, and left unchecked, had been unintentionally added to an estimated 350,000 open-source projects and prevalent in closed-source projects.

Today, we're excited to share an update on this work. Through GitHub, our vulnerability research team has patched **61,895** open-source projects previously susceptible to the vulnerability. This work was led by [Kasimir Schulz](#) and [Charles McFarland](#), and concluded earlier this month.

## Phased approach to patching at scale

Open-source developer tools, like Python, are necessary to advance computing and innovation, and protection from known vulnerabilities requires industry collaboration, especially since many open-source projects lack dedicated staff and resources. To effectively minimize the vulnerability surface area, Trellix Advanced Research Center executed a months-long automated effort to patch open-source projects known to use the vulnerable code.

Through GitHub, developers and community members are able to push code to projects or repositories on the platform via a process called [pull request](#). Once a request is opened, the project maintainers review the suggested code, request collaboration or clarification if needed, and accept the new code. In our case, the code pushed via pull request delivered unique patches to each of the vulnerable GitHub projects.

### RECENT NEWS

- Jan 17, 2023  
[Trellix Endpoint Scores 100% Detection with Zero False Positives in Latest SE Labs Endpoint Security Test](#)
- Dec 7, 2022  
[Trellix Predicts Heightened Hacktivism and Geopolitical Cyberattacks in 2023](#)
- Nov 30, 2022  
[Trellix Expedites Delivery of XDR with AWS](#)
- Nov 16, 2022  
[Ransomware Activity Doubles in Transportation and Shipping Industry](#)
- Sep 28, 2022  
[Trellix Expands XDR Platform to Transform Security Operations](#)

### RECENT STORIES

- Jan 30, 2023  
[Trellix's Differentiated Position in the XDR Market - Highlighted in](#)

# Investing Back Into OSS Security

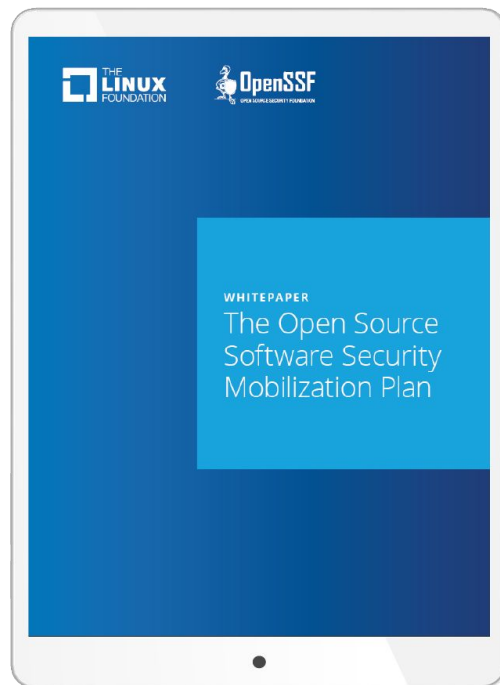
# In Response to Log4Shell, the OpenSSF Developed a Plan

The OpenSSF community developed a series of 3 to 5 page proposals answering the following questions:

- What are the major problems to address that would lead to better open source software supply chain security?
- What pre-existing efforts, whether inside OpenSSF today or not, are already starting to address those problems?
- Building on those pre-existing efforts, what financial and other resources would it take to fully or mostly tackle each problem?
- What are some pragmatic but ambitious targets we can set for solutions to each problem, with measurable results within the first two years?

# The Result: The Open Source Software Security Mobilization Plan

- First-of-its-kind plan to broadly address open source and software supply chain security
- Developed collaboratively by the OpenSSF Governing Board and OpenSSF's expert community
- Details \$150M of funding over two years to rapidly advance well-vetted solutions to ten major problems the plan identifies



**\$150M**  
**may sound like a lot of money**

~~\$150M~~

**\$700M**

is the fine the FTC levied on Equifax  
for the 2017 data breach caused in part by  
unpatched OSS (Apache Struts)

## Goals Identified:

**Secure OSS  
Production**

**Improve  
Vulnerability  
Discovery &  
Remediation**

**Shorten  
Ecosystem  
Patching  
Response Time**

# Multi-year investment into key OSS initiatives will modernize the security and integrity of the software supply chain.

STREAM	FIRST YEAR	SECOND YEAR
1. Baseline Secure Software Development Education	\$4.5M	\$3.5M
2. Risk Assessment Dashboard for OSS	\$3.5M	\$3.9M
3. Digital Signatures to Deliver Enhanced Trust	\$13M	\$4M
4. Replacement of Non-Memory-Safe Languages	\$5.5M	\$2M
5. Open Source Security Incident Response Team	\$2.75M	\$3M
6. Accelerate Discover and Remediation of New Vulns	\$15M	\$11M
7. Third Party Audits/Code Reviews and Remediation	\$11M	\$42M
8. Data Sharing to Determine Critical Projects	\$1.85M	\$2.05M
9. SBOMs Everywhere: Security Use Cases, Tooling	\$3.2M	TBD
10: Build Systems, Package Managers, and Distribution Systems	\$8.1M	\$8.1M
<b>Total</b>	<b>\$68.4M</b>	<b>\$79.5M</b>

# Launched at the Open Source Software Security Summit II

- Washington, DC on May 12-13, 2022
- The Linux Foundation and OpenSSF gathered a cross-section of open source developer & commercial ecosystem representatives along with leaders & experts from key U.S. federal agencies
- We reviewed the plan together, both at a high level and into specifics, to ensure they were the right targets, and that they built on the work the US Government had already begun.
- **Through the event we received \$30M in pledges from OpenSSF members towards the plan.**



# Open Source Software Security Summit Japan

- Tokyo, Japan on August 3, 2022
- The Linux Foundation and OpenSSF gathered a cross-section of senior cybersecurity representatives from leading Japanese firms, OpenSSF members, and representatives from the Japanese government.
- We convened to discuss open source software (OSS) security challenges, modern challenges to the global software supply chain, and how to accelerate improvements. We discussed how each stream of the Mobilization Plan could align with national policies and priorities for Japan, and how Japanese industry could participate in the further definition and implementation of the plan.



# Future: where do we go from here?

- We plan to continue to work towards funding and establishing the workstreams in the Mobilization Plan
- The Plan is necessary but not sufficient; other actors have roles to play beyond the OpenSSF, and open source needs sustainable long-term funding where the users of OSS give back to the software repositories that they use.
- Other actors have roles to play, including government; for example, we saw the introduction of the [Securing Open Source Software Act](#) in the United States Congress in September 2022
  - We plan to continue to engage and work together with policymaker efforts to do what is most beneficial for the open source community.

# In Summary:

- Attacks on the integrity of open source software — and as a result, on the full software supply chain — are increasingly disruptive and require coordinated industry efforts to address. We have learned more about the nature of these problems since Log4Shell.
- Key steps in improving security of the OSS ecosystem:
  - a. **Education** (Concise Guides, Courses)
  - b. **Measurement** (Scorecards, Best Practices Badge, Criticality Scores)
  - c. **Tooling** (Sigstore, SLSA) and **Defaults** (working with IDEs and distributors)
  - d. **Investment** (Where are the weakest links? How can we address upstream?)
- The OpenSSF is here to help.

# Thank you!

