OpenSSF

OPEN SOURCE SECURITY FOUNDATION

Securing Upstream: Addressing the Systemic Issues in the Software Supply Chain That Led to Log4Shell

Brian Behlendorf, OpenSSF GM

October 2022

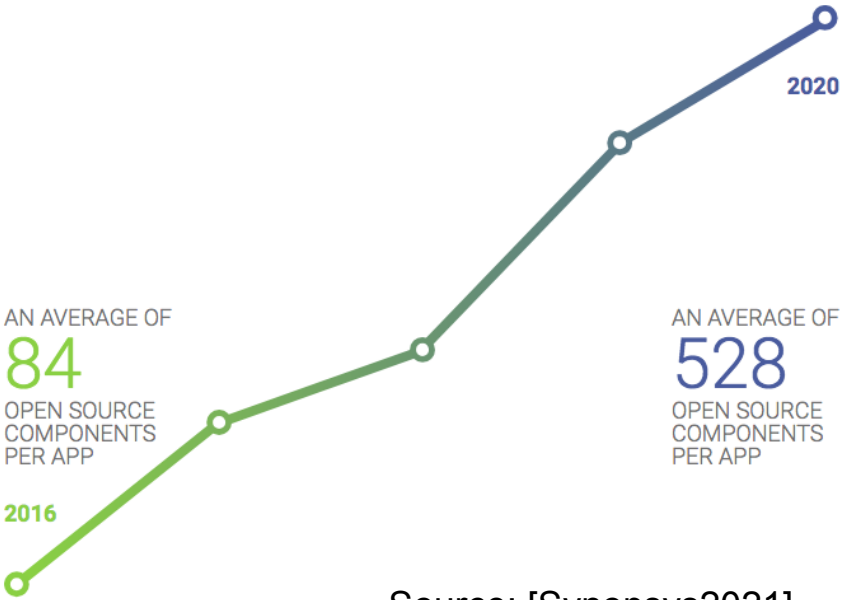
Open Source is critical part of the software supply chain

98%

Percent of codebases and Android apps that contain OSS [Synopsys2021]

70-90%

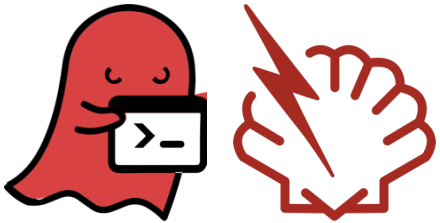
Percent of codebase that was OSS on average [Synopsys2020] [Sonatype2020]



Source: [Synopsys2021]

Source:
[Synopsys2021] "2021 Open Source Security and Risk Analysis Report" by Synopsys https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html
[Synopsys2020] "2020 Open Source Security and Risk Analysis Report" by Synopsys https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2020-ossra-report.pdf
[Sonatype2020] "2020 Stateof the Software Supply Chain"by Sonatype https://www.sonatype.com/resources/white-paper-state-of-the-software-supply-chain-2020

All software under attack, via vulnerabilities & supply chain



GitHub hacked, millions of projects at risk of being modified or deleted

By Sebastian Anthony on March 5, 2012 at 7:22 am | 21 Comments

Log4Shell

POLICY

Cleaning up SolarWinds hack may cost as much as $100 billion

Government agencies, private corporations will spend months and billions of dollars to root out the Russian malicious code

Source: https://www.rollcall.com/2021/01/11/cleaning-up-solarwinds-hack-may-cost-as-much-as-100-billion/

SolarWinds' Orion attacked via a *subverted build environment*

Computer Science > Cryptography and Security

[Submitted on 19 May 2020]

Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks

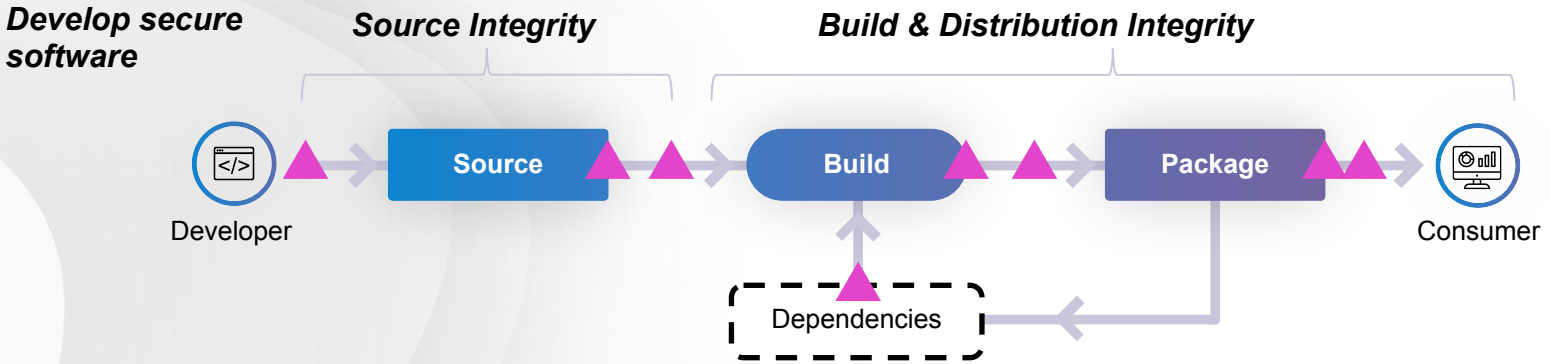
Marc Ohm, Henrik Plate, Arnold Sykosch, Michael Meier

A software supply chain attack is characterized by the injection of malicious code into a software package in order to compromise dependent systems further down the chain. Recent years saw a number of supply chain attacks that leverage the increasing use of open source during software development, which is facilitated by dependency managers that automatically resolve, download and install hundreds of open source packages throughout the software life cycle. This paper presents a dataset of 174 malicious software packages that were used in real-world attacks on open source software supply chains, and which were distributed via the popular package repositories npm, PyPI, and RubyGems. Those packages, dating from November 2015 to November 2019, were manually collected and analyzed. The paper also presents two general attack trees to provide a structured overview about techniques to inject malicious code into the dependency tree of downstream users, and to execute such code at different times and under different conditions. This work is meant to facilitate the future development of preventive and detective safeguards by open source and research communities.

Source: https://arxiv.org/abs/2005.09535

Reviewed 174 OSS SC attacks 2015-2019,
61% malicious packages use typosquatting

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

Log4Shell Breaks the Internet

- December 2021
- Industry-wide four-alarm-fire to address a serious vulnerability in the Apache Log4j package
- Humbling reminder of how far we still have to go to security the open source software supply chain

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Log4Shell Timeline (aka Everyone's Ruined Holiday)

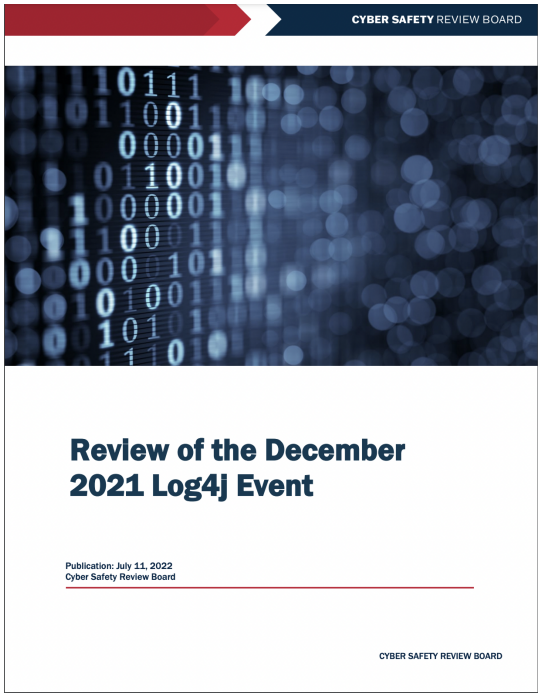
- Nov 24, 2021: Security researcher Chen Zhaojun discovers the now infamous **CVE-2021-44228**, or "Log4Shell," vulnerability that allows unauthenticated attackers to execute remote code on vulnerable systems, scoring a CVSS of 10 out of a possible 10.
- Dec 6: Apache Log4j releases version 2.15.0 to 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 government entities like 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
- Log4j has become an “endemic vulnerability” that will be exploited for years to come
- Download at:
  - https://www.cisa.gov/sites/default/files/publications/CSRB-Report-on-Log4-July-11-2022_508.pdf

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Contributing Factors

- Log4j is routinely embedded in other software components, often unknown at later levels of integration or system operation, making it a difficult vulnerability to identify using common scanning approaches
- Log4j framework is incorporated into thousands of software components globally, and many of the nation's critical infrastructure and government systems rely on it
- Consistent with industry convention, the Log4j and ASF teams do not manage or know who uses the software they produce, or how it is eventually used
- Log4j vulnerability is easy for attackers to exploit and affords a high degree of system control to an attacker once exploited

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

A Perfect Storm (Brian's Added Take)

- The Log4J developers weren't amateurs, or untrained in security work, or careless.
- They accepted a feature submission for a major subsystem that was not well maintained after the initial contribution.
- They were not in general over-worked as a team before the exploit, but during the incident they were extremely stressed and short of useful assistance.
- Users of Log4j "took it for granted" - or more benignly put, were not well informed about its risk factors, or consequently the need to invest in its security.

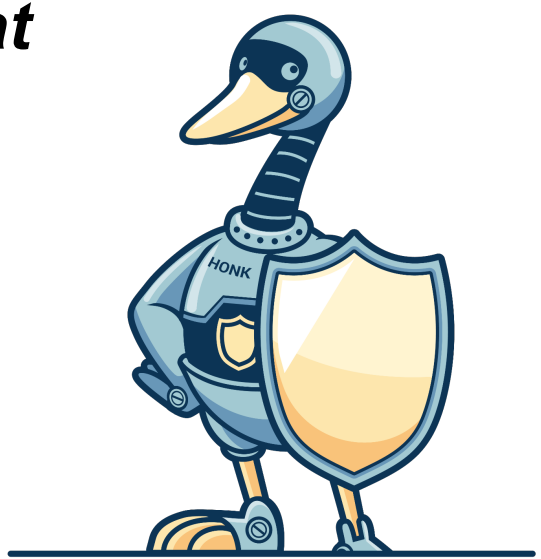
* The OpenSSF Has Entered The Chat

Established by the Linux Foundation in 2002, the OpenSSF is a **global initiative** securing investment, resources, and expertise to 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

Working Groups (and their projects & Associated 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

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

OpenSSF products your organizations can consume (and help us develop!) to improve your security posture

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)

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

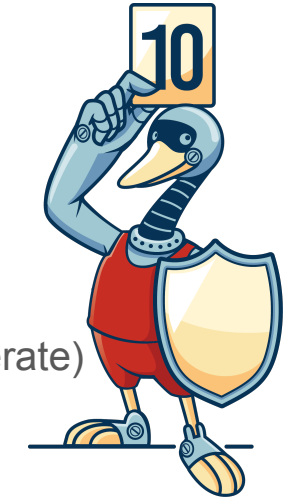
Course: Secure Software Development Fundamentals

- *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)
- See: https://openssf.org/training/courses/

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
- A project within the OpenSSF Best Practices Working Group (WG)
- For more, see: https://bestpractices.coreinfrastructure.org

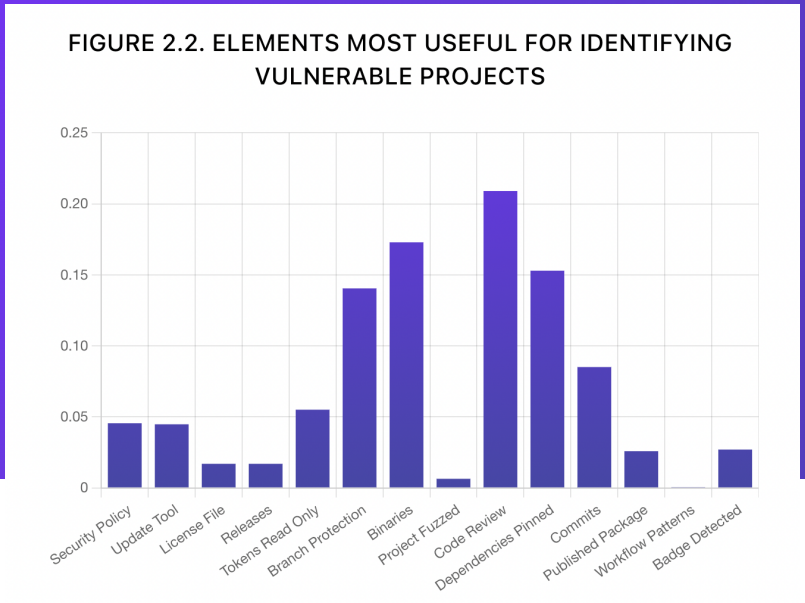
OpenSSF Scorecards

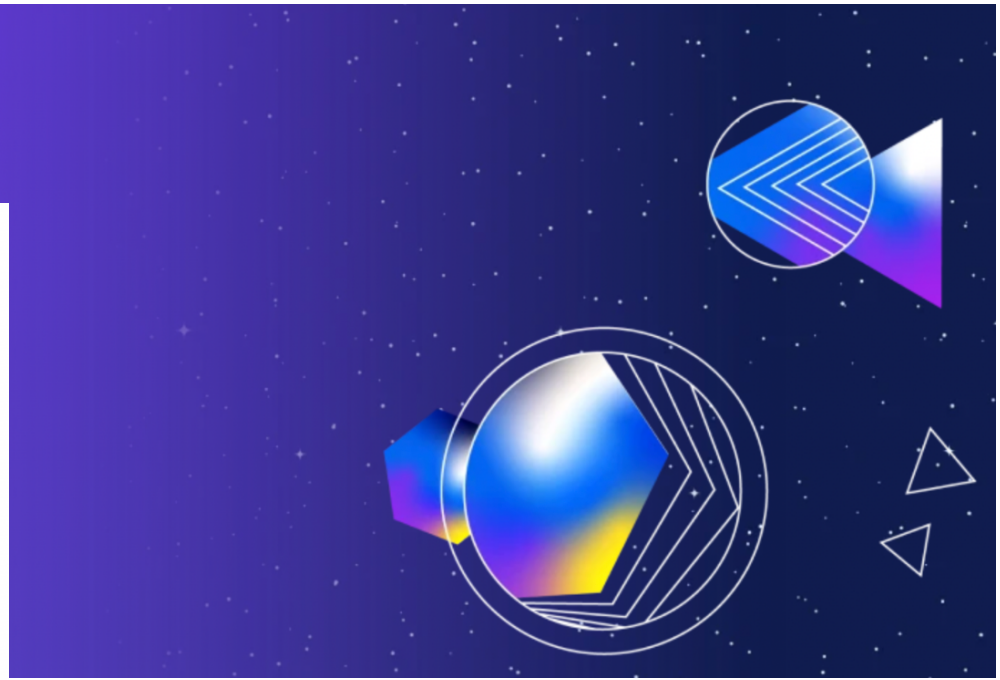


- *Automatically* scores OSS projects on heuristics ("checks")
  - Each related to security, scored 0-10, weighted average computed
  - Can use to evaluate your own or others' projects (they don't need to cooperate)
  - Currently only works on projects hosted on GitHub (not fundamental)
- Sample checks:
  - 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 found to be highly effective measures to assess project security

Project Quality Metrics



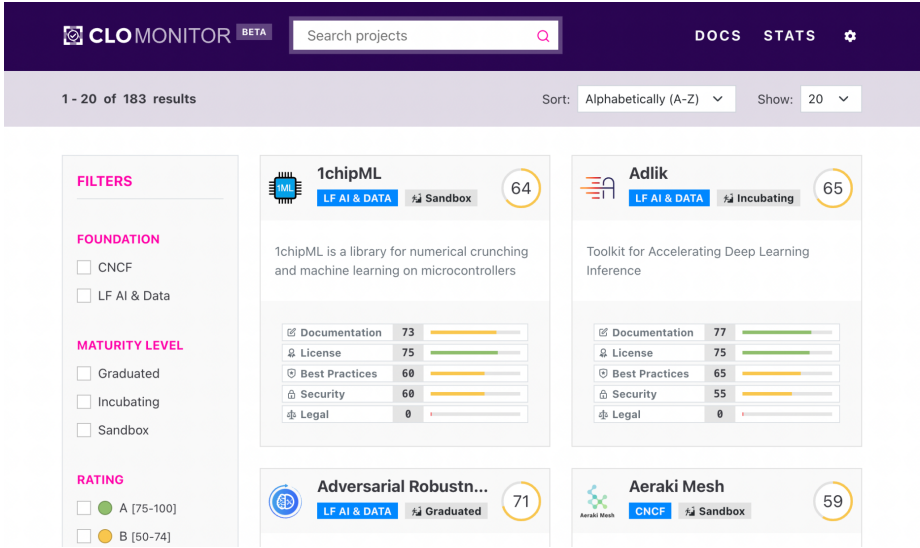


Source: Sonatype https://www.sonatype.com/state-of-the-software-supply-chain/project-quality-metrics

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Scorecards in Usage

- CNCF's CLO Monitor uses the Scorecard API for its measurements
  - https://clomonitor.io/
- SecuritySlam in progress now
  - CNCF project maintainers aim 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/



OpenSSF
OPEN SOURCE SECURITY FOUNDATION

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 (OpenSSF project) in development to be 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

Source: https://sigstore.dev/what_is_sigstore/

Some Other OpenSSF Materials

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

In Progress: 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
  - Please join us!

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Investing Back Into OSS Security

In Response to Log4j, 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

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
Total	**$68.4M**	**$79.5M**

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

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.**

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

$150M

may sound like a lot of money

~~$150M~~
$700M

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

$700M

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

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.
- There are many things enterprises can - **must** - do to increase their degree of scrutiny of the open source components they use.
- Education (concise guides, courseware), measurement (Scorecards, Best Practices Badge, criticality scores), and then investment (where are the weakest links? How can we address upstream?) are the key steps.
- The OpenSSF is here to help, but also to drive systemic change across the OSS landscape and beyond.

Thanks.