Mojaloop
Platform Quality & Security

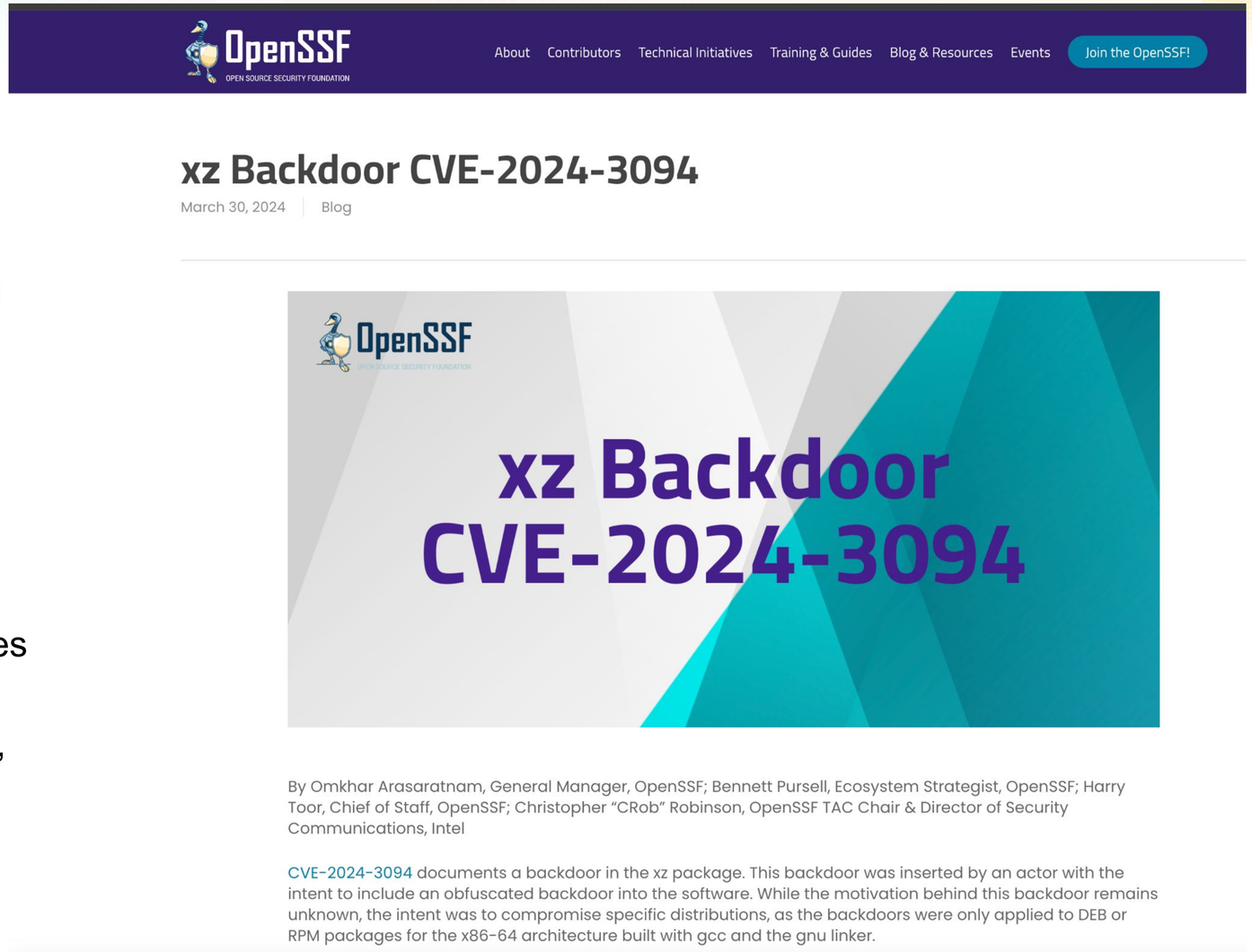
Mojaloop PI-25 Online community event, Sam Kummary, MLF

Mojaloop PQS: Context, recent issue

Recent vulnerability in tax xz

Mitigation process at Mojaloop:

1. Reviewed guidance, recommendations and vulnerability details
2. Internal assessment of issue and impact
3. Detailed assessment of Mojaloop components and tooling
4. Addressing issue - fixes where possible, mitigation measures where needed
5. Guidance offered to community and releases made with fixes
6. Internal review of mojaloop platform assets, procedures and new measures introduced



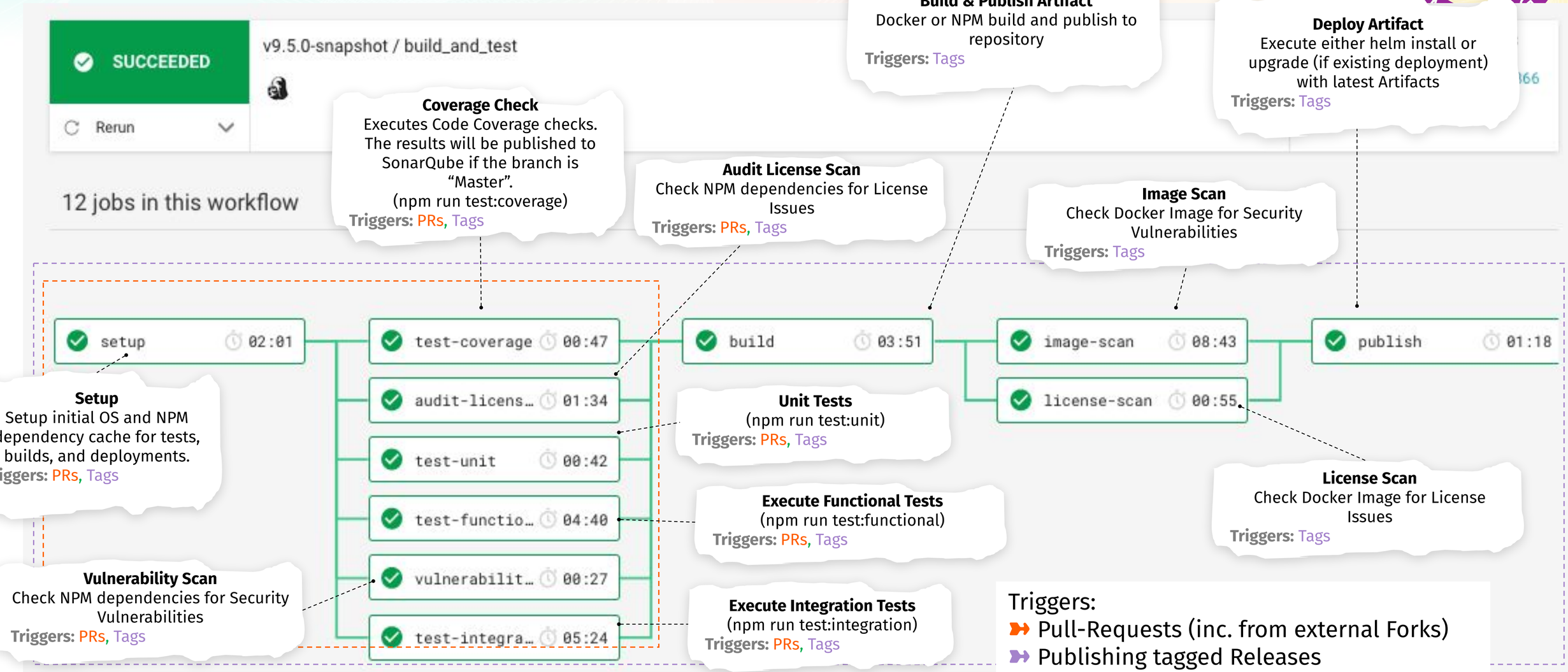
OpenSSF blog post on tar xz vulnerability:
https://openssf.org/blog/2024/03/30/xz-backdoor-cve-2024-3094

Mojaloop PQS: Measures for OSS Security

1. Image scanning
2. License scanning
3. Dependant alerts
4. Codeql
5. MFA where user access is involved
6. GitHub's secret scanner
7. Protected main branches on Git
8. Codeowners and PR reviews (requirng approvals) for critical repos
9. Periodic security testing for vulnerabilities, etc.
10. Learnings from pen-testing done by specific implementations and addressing vulnerabilities

PQS: CI/CD Pipeline

Ref https://app.circleci.com/projects/project-dashboard/github/mojaloop



Mojaloop PQS: Mojaloop release v16.1.0 RC

Review core and related repositories released with Mojaloop for

1. Snyk alerts for Central-ledger repository are addressed
2. Dependabot alerts for Central-ledger repository are addressed (moderate, high and critical)
3. Ensure main branch is protected and collaborator list is up-to-date
4. Ensure open PRs are addressed, closing stale PRs
5. Update audit exceptions json file to remove exceptions added that are not necessary anymore
6. Close issues on the repository that are fixed / out-of-date
7. Ensure codeowners file is current

Reviewing updates and vulnerabilities (GitHub, NVD, other related media) constantly

1. Take necessary action for Mojaloop services as required
2. Provide guidance and mitigate exposure to Mojaloop

Mojaloop PQS: Design & Implementation guidance

1. Mojaloop Invariants: https://docs.mojaloop.io/community/standards/invariants.html
2. Mojaloop CVD policy:

   https://docs.mojaloop.io/community/contributing/cvd.html#disclosing-and-receiving-information-regarding-security-vulnerabilities
3. Mojaloop cyber-security architecture: https://docs.mojaloop.io/community/tools/cybersecurity.html#mojaloop-cybersecurity-architecture
4. Guidance around moving from alpha / beta to release quality:

   https://community.mojaloop.io/mojaloop_foundation/pi-22-core-releases-work-stream-reviews-4ami
5. Other Mojaloop standards: https://docs.mojaloop.io/community/standards/guide.html#standards

Mojaloop PQS: Objectives

1. Hardening of Mojaloop releases
2. Security testing of Mojaloop
   - Assess feedback from testing done by security experts, internally ✅
   - Prioritize issues based on criticality and severity ✅
   - Update backlog with stories to address issues
3. Review core and related repositories released with Mojaloop for
   - Ensure dependabot alerts are addressed ✅
   - Ensure Snyk issues are addressed
4. Review any tasks “to-do” in the codebase and related quality issues
5. Review PRs in critical repositories and address (close, approve / reject)
6. Update nodejs version for any services not yet upgraded ✅
7. Update maintenance tooling (license-scanner and such) ✅
8. GitHub maintenance of Mojaloop repositories (used in the platform) (continuation)
9. Hardening Mojaloop core and addressing any known and reported quality, security issues
10. SBOM of dependencies in Mojaloop to better manage situations

Mojaloop PQS: SBOM for Mojaloop

SBOM of all services included in Mojaloop Platform's core (getting started)

1. Exploring tooling - CycloneDX and SPDX
2. Individual SBOMs for core repositories
3. Centralized list for all core / critical repositories
4. Automate generation during release time
5. Use circleci/cron jobs for maintenance

Mojaloop PQS: SBOM for Mojaloop & Metrics

Goals

1. Generate and document SBOMs for each repository/service.
1. Automate monthly publication of metrics and SBOMs.
2. Flag anomalies and discrepancies in SBOMs.
3. Maintain a list of open security alerts for core platform repositories.

What is an SBOM?

A Software Bill of Materials(SBOM) is a machine and human-readable list of a project's entire software inventory.

Why do organizations need an SBOM?

1. Transparency
2. Security
3. Compliance
4. Maintenance

Formats - generated/converted

xml - json - csv - html

What all does SBOM contain?

1. Open source components
2. Third-party components
3. Licenses
4. Versions of components
5. Patch status of components
6. Open source vulnerabilities
7. Package names
8. Package versions

CycloneDX Generator

1. It is the official OWASP SBOM tool
2. Supports many languages-C, C++, JavaScript, Java, Python
3. Comes with a CLI that can scan locally or as part of a CI/CD pipeline and an API server
4. Output format is CycloneDX.

Links

1. https://github.com/mojaloop/community-tools/issues/9
2. https://www.npmjs.com/package/%40cyclonedx/cyclonedx-npm
3. https://github.com/mojaloop/ml-api-adapter/pull/530

Mojaloop PQS: PI-25 Roadmap

1. Stress / Load testing of Mojaloop
2. SBOM for component and Mojaloop platform
3. Hardening of Mojaloop releases - continuation
4. SonarCloud - address issues in detail
5. Improve tooling used in CI/CD - ci tools, license checkers
6. OpenSSF checklist and compliance
7. Blog posts and guidance to implementers and improve documentation

PQS Workstream Mojaloop slack channel: https://mojaloop.slack.com/archives/C06UW0E2KBN