Code Quality & Security Workstream Update

PI 20 Feedback Session – DevSecOps Maintenance and Enhancements

Presenter

Godfrey Kutumela

PI Contributors – Miguel & Aime(Helm Signing), Omaru (Code security standards with API guideliness) and Pedro (Code Security Standard with design principles)

PI Objectives

The focus is to enhance DevSecOps practices to ensure secure code delivery to our adopters.

1. Perform a quarterly baseline open-source software (OSS) scan to detect and mitigate software supply chain risks.

2. Develop and implement a code security standard – Including the OSS policy.

   - ❖ Enforce the code security standard using CodeQL SAST *(Blocked due unavailability of Github Enterprise License)*

3. Plan and implement a code signing mechanism.

Mojaloop DevSecOps Journey

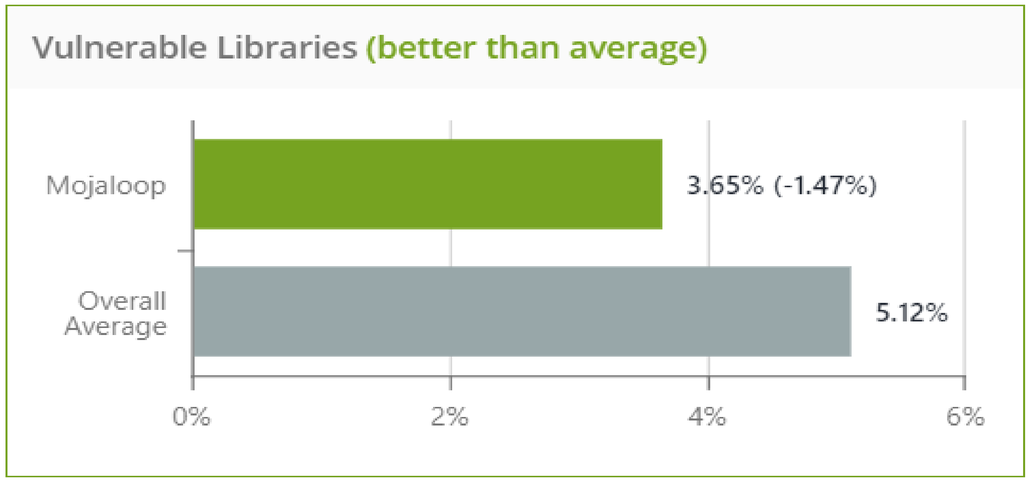
Requirements	Design	Development	Deployment	Run
Product Security Functional Requirements *(Used DDD on vNext)* Compliance Requirements Threat Modeling	Enforce secure design principles *vNext and Actio are reference implementations*	Open-source management Alignment with secure software standards (OWASP Top 10 and CWE/SANS Top 25 PE)	Ensure secure deployment through: Helm Charts Mini-loop IAC AWS/Azure	Vulnerability detection and patching management Maintain secure runtime configuration Security monitoring & incident response
Mojaloop			**Mojaloop and Implementers**	

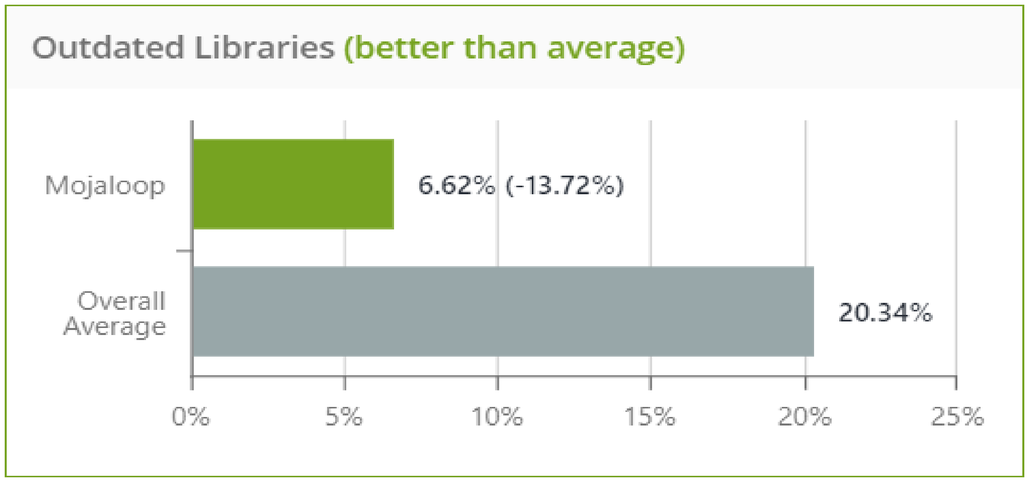
Quarterly OSS Audit Results - 2022

All Mojaloop repos (excluding vNext and Actio) were included in the scan:

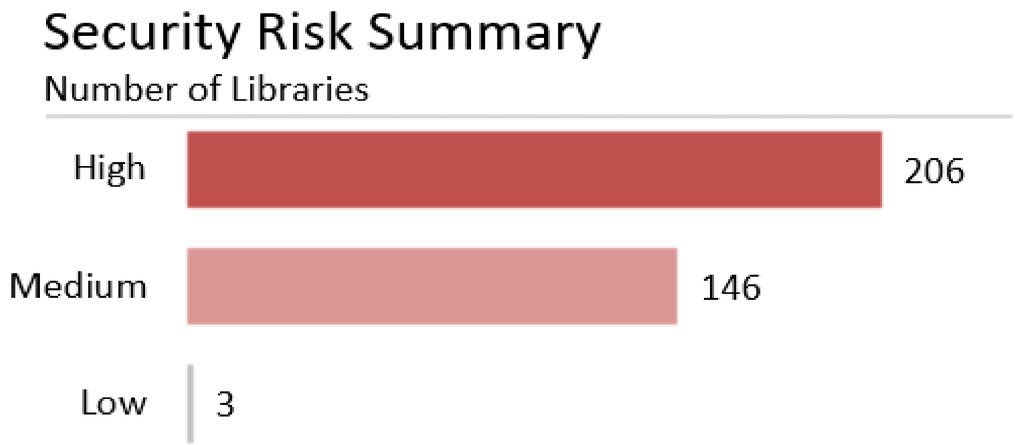
18 January	18 April	26 July	27 October
104 Codebases, 7055 libraries • **6625** libraries are up to date • **430** libraries are outdated • **130** libraries with multiple versions	**111 Codebases, 7729 libraries** • **7221** libraries are up to date • **489** libraries are outdated • **165** libraries with multiple versions	**119 Codebases, 8461 libraries** • **7881** libraries are up to date • **561** libraries are outdated • **167** libraries with multiple versions	**121 Codebases, 8499 libraries** • **7918** libraries are up to date • **563** libraries are outdated • **169** libraries with multiple versions

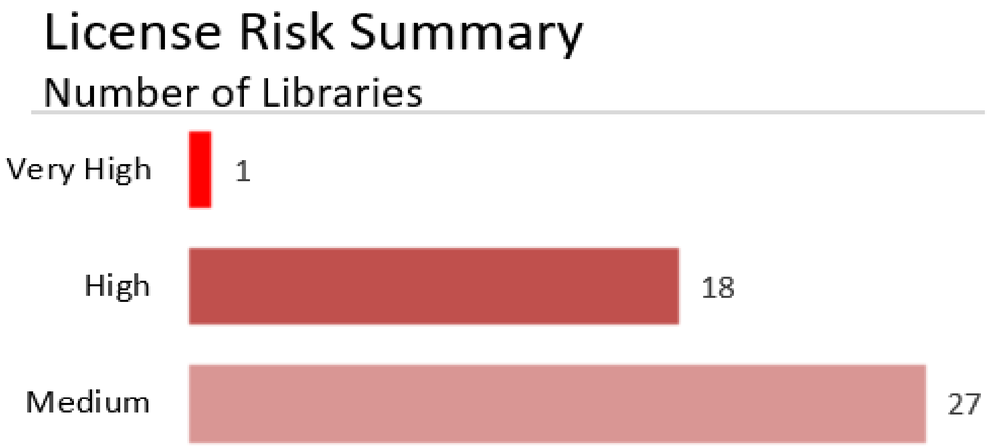
According to WhiteSource Benchmark database as of 24 Oct 22, Mojaloop is better than average in both vulnerable and outdated libraries.





OSS Scan Findings Summary





Security Risk Analysis

- **Most of the high security findings affects only transitive dependencies** and requires a refinement of the dependency strategy

- **10 libraries** (convict-6.0.0.tgz, ejs-2.7.4.tgz, handlebars-4.7.6.tgz, json-pointer-0.6.1.tgz, url-parse-1.5.3.tgz, minimist-1.2.5.tgz, uglify-js, lodash, y18n-3.2.1.tgz & mongoose-6.0.11.tgz) have many vulnerabilities classified as critical according to CVSS3 scoring (>9)

License Risk Analysis

- **10 libraries** have alternative permissive licenses (Apache, BSD, MIT), reducing the risk to low

- **8 libraries (**fuzzball-1.3.0.tgz, schema-utils-3.0.0.tgz, cbor-4.3.0.tgz, docker-hub-api-0.8.0.tgz, jxon-2.0.0-beta.5.tgz, librejs-librejs-7.20.2*, types-5.0.0.tgz & utils-0.2.3.tgz have GPL license only which makes them non-compliant with the Mojaloop Apache 2.0 license policy

Mojaloop Code Security Standard

The objective is to establish rules and public guidelines to prevent software design flaws and security vulnerabilities within the Mojaloop codebase. It covers 4 types application security flaws, exposures, and how Mojaloop has planned to prevent and mitigate against those.

Secure Design Principles Leveraging OWASP Top 10(Web, Mobile & API) and SANS CWE Top 25 + Additional Guidelines *Mojaloop Reference Architecture Security Model*	**Open-source software (OSS) policy implementation** *(License permissibility, support and version management)* *Enforced through CI/CD rules using NPM audit, Dependabot and Mend*	**Secure Coding Practices** to enforce OWASP and SANS CWE Top 25 Programming Errors *Selected CodeQL as the SAST tool to enforce coding rules and guidelines*	Use of Threat Modeling to detect and avoid application logic flaws (Business, Data Leakage and Insider Actors) *Not mandatory and open for contributions from the community*
Started and Ongoing Improvement		PI 21 Backlog	Ongoing development

The code security standard will be enforced with every code change or significant addition to the Mojaloop code base. Link to Github documentation - Mojaloop Code Security Standard - 13 October 2022.docx - Google Docs

Code Signing using Helm Provenance & Integrity

The objective of implementing code signing is to assure origin authentication to the implementors of the Mojaloop software.

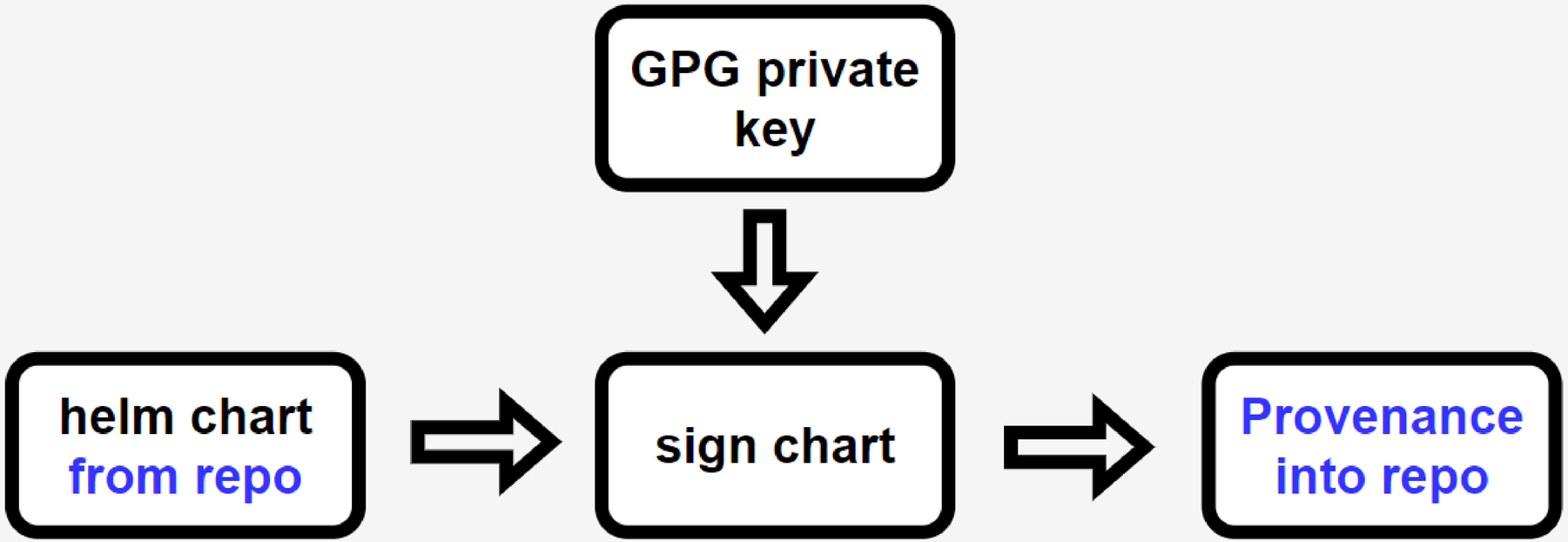
1. Helm has Provenance Tools which helps chart users verify the integrity and origin of Packages.

2. Uses PKI, GnuPG, and package managers.

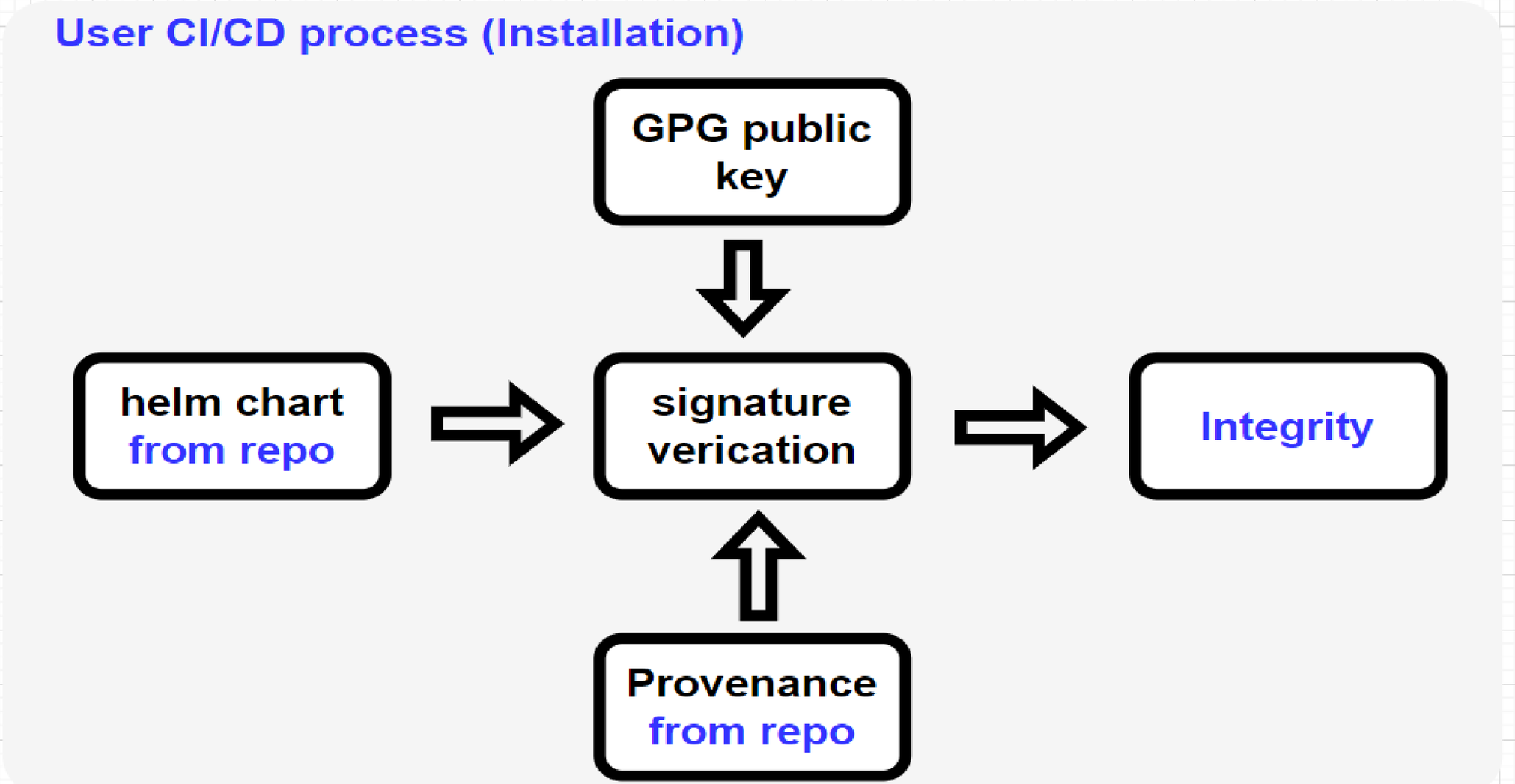
3. Helm can generate (Provenance) and verify signature files (Integrity).

Helm Provenance & Integrity – CI/CD

Mojaloop CI/CD process (code packaging)



Code Signing using Helm Provenance & Integrity



muchas gracias

Thank you

Questions and comments