mojaloop

Mojaloop Quality, Security & Compliance Improvement Workstream

PI-10 Meeting – 23 April 2020

Core Team – Godfrey, Kim, Lewis & Pedro – Crosslake
Miguel, Sam & Victor – Modusbox

Support Team – Adrian, Aime, Azeem, Creg, Donovan, Max, Miller, Philip, Renjith & Simeon

mojaloop

Overall Journey – PI to PI View

PI 1 – P 7 (Ensure platform security from design & build phase of the project)

1. Coding Standard
2. Signatures for Non-Repudiation and Integrity
3 Encryption & PKI standard to preserve confidentiality
4. Open Source license policy
5. Static Code Analysis - code quality and security
6. API Security for API based threat

PI 8 (Tackle Container security, identify and close critical gaps)

1. Introduce Container Security Policy Gates into our CI\CD release pipeline
2. Perform Docker Image Vulnerability Scanning
3. Baselining of docker container configuration (Dockerfile Security) against CIS cloud Security best practice
4. Identify critical security gaps and benchmark open source\commercial toolset

PI 9 (Engage, Standardize security architecture and tackle compliance)

1. Improve community engagement – Issued a vulnerability reporting procedure & a code security summary
2. Started tackling compliance requirements – Data Privacy (GDPR) and Payment Security Standards – PCI DSS and HSM Deployment.
3. Performed Mojaloop overall security architecture review.

Key Objective

To continuous improve the overall **Trust** of the Mojaloop Platform inline with best practice standards.

Trust entails Reliability, Privacy, Transparency, Security, and Compliance.

PI 9 Objectives

1) Provide a procedure for the public to report vulnerabilities.
2) Provides a summary of all code level security measures built into the Mojaloop Platform.
3) Define GDPR scope focusing on what is addressable at Mojaloop level.
4) Container security enhancements and policy customization.
5) Benchmark adopted open source tools against commercial tools.
6) Perform a platform security architecture review and standardize (Top down approach).
7) Plan HSM deployment - explore best practice design options and identify use cases.
8) Solicit Input from TIPS and Mowali – Pain points and requests.

Code Quality/Security Epic - https://github.com/mojaloop/project/issues/1213

Source Code Security Summary

Below are source code level security measures that has been applied to the latest code base through our CI\CD release pipeline. The key **objective** of these measures is to ensure Mojaloop OSS code is of ***high quality, secure and compliant*** to our DevSecOps policies and coding standards:

- **Open Source Vulnerability Management** – To enforce consistent use of non-vulnerable and non-obsolete open source components and dependencies on the Mojaloop code base.

- **Open Source License Compliance** – To ensure compliance with Mojaloop open source license policy

- **Static Code Analysis** – To ensure that all custom code is free of well known low coding quality issues and vulnerabilities before any code merging, pull request or release as per our CI\CD pipeline.

- **Container and Docker Security** – To ensure compliance with our adopted secure image configuration standards and find and fix medium\high container based vulnerabilities in all repos of our code base.

We encourage Implementation teams to adopt our open source toolset or any other toolset (open source or commercial) to address the above objectives.

Link - Project#1222 Application security summary

Vulnerability Reporting Procedure

The Objective of the procedure is to guide community members and the public on **how security vulnerabilities should be reported safely and responsibly** to the dedicated security team within Mojaloop.

We strongly encourage everyone to alert us of the potential security vulnerabilities privately first, before disclosing them in a public forum – a right everyone is entitled to without any permission whatever from us as the maintainer of the platform.

Link - Project#1224 Define a vulnerability reporting procedure

Define OSS GDPR Scope

Goal :

As a "Hub Operator or Implementer", I want to "understand how the OSS community can help me fulfil my GDPR requirements", so I can "plan accordingly in my organization“

Tasks : Investigate how we can handle GDPR Requirements as a part of Mojaloop OSS Initiative.

- Produce a list GDPR controls for Mojaloop – Completed in PI 9
- Perform a gap assessment and architecture review – deferred to PI 10
- Solution design and an implementation plan - deferred to PI 10.

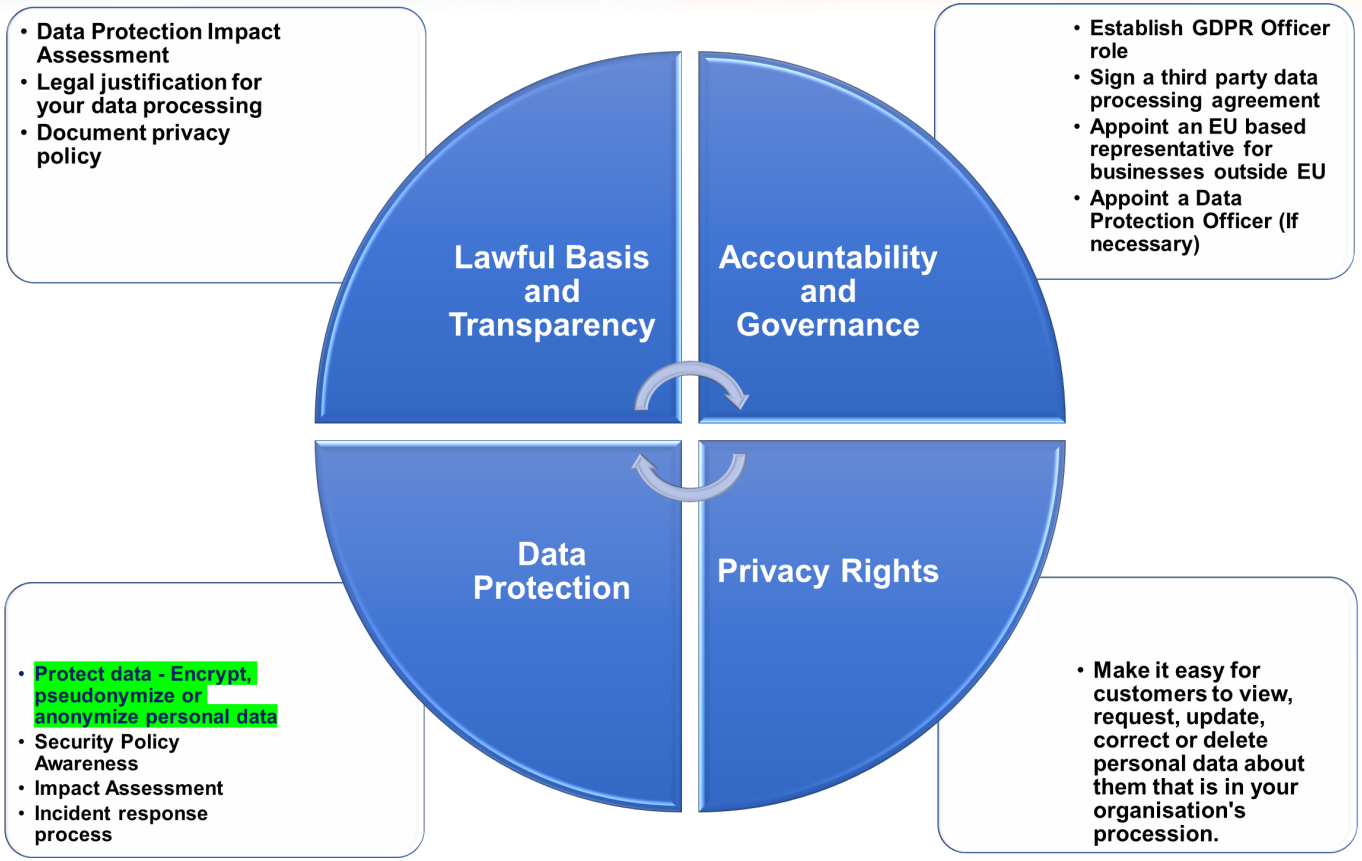
Output:

Below are the key GDPR controls to be addressed as part of OSS level :

- Take data protection into account from product development to data processing.
- Encrypt, pseudonymize or anonymize personal data *whenever possible*

For more detail on the OSS GDPR scope - https://github.com/mojaloop/project/issues/1225

Define OSS GDPR Scope



For more detail on the scope - https://github.com/mojaloop/project/issues/1225

mojaloop

1. Docker Security Enhancements.
2. Benchmarking of our adopted open source container vulnerability tools against commercial tools.

By Lewis and Victor

mojaloop

Container Security Enhancements

1) Added custom policy and tooling for evaluating Mojaloop container scans:
   a) Policy evaluation based on Docker CIS Benchmark best practices
   b) "Vulnerability Diff" A tool for comparing found vulnerabilities between Base Docker Images (**`node:12.16.1-alpine`**) and Derived Docker Images (**`Mojaloop/central-ledger:v9.5.0`**)
      See https://github.com/mojaloop/ci-config#container-scanning for more information

2) Started implementing Dockerfile best practices as guided by CIS Benchmark
   a) Focus on runtime pod configurations
   b) Key findings:
      i. Enable auditing on docker daemon and critical docker files. Do not mount sensitive system dir's are not mounted in containers
      ii. Configure AppArmor profile
      iii. Restrict traffic between containers on default bridge
      iv. Ensure live restore is enabled. Add healthcheck instructions to containers
      v. Restrict containers from acquiring new privileges. Do not run containers with root privilege.
      vi. Run containers with specific users other than root. Restrict ownership of specific docker files to root.
      See https://github.com/mojaloop/project/issues/1082 for more detail on the Dockerfile security recommendations

3) Focus on tightening existing security tooling, without slowing down developer workflow
   a) Better Slack build notifications
   b) Shared CI config that can be updated across all repos

Benchmark adopted Open Source Container Vulnerability Managements Tool Against Commercial versions

Goal: To establish if there are any significant gaps between the open source tools we have adopt and the commercials versions specific to secure container configuration and vulnerability management

- Tests done with out of the box settings.
- Both commercial tool and community tool give vulnerabilities identified output
- Community
  - On average tool listed more vulnerabilities listed
  - Commercial tool offer neater reports with easy to prioritise options
  - Limited ability to connect to proprietary repositories.
  - Manual reporting i.e. extract and compile by yourself
- Commercial tool
  - Threat tracking as well as dashboards to track trends as well as reporting
  - Integration to various repositories (docker, quay, gcr)
  - Auto reporting including scheduled alerts
  - Advanced features available

With workarounds we feel community tools can comfortably support an open source CI/CD process.

Comparison Summary

AREA	OPEN	COMMERCIAL
Vulnerability Database & Updates	Open source vulnerability database e.g. https://docs.anchore.com/current/docs/overview/feeds/	Proprietary feeds
Out of the box Features	Basic features. 1. Manual Scanning 2. CI/CD integration 3. On prem and Cloud support 4. Additional features available on Enterprise tool (commercial)	Basic features. 1. Manual Scanning 2. CI/CD integration 3. On prem and Cloud support 4. Enterprise features such as workflows, automations, exploit testing, trend analysis, linkage to enterprise vulnerability tracking
Vulnerability Findings effectiveness	Vulnerability details with CVE	1. Vulnerability details with CVE 2. Exploitability and details of available exploit (some tools)
Liability	Limited liability due to open source	Enterprise support cover
Hidden Costs	Many open source projects provide active forums, up-to-date documentation and detailed tutorials, but there is a trend towards charging for dedicated support, something to watch out for if your budget is tight.	Cost as quoted at time of purchase. Usually license + professional services + AMC and support (other costs may include hardware etc.). Additional features may be licensed separately.

In Conclusion, there are no notable gaps between our adopted open source toolset against leading commercial tools that can leaves exposed. We will continue using the current tools and updating them as the threat landscape changes.

mojaloop

1. Security Architecture Review\Update.
2. HSM Deployment Recommendations.

By Pedro & Godfrey

mojaloop

Security Architecture Review & Standardization (Top Down Approach)

1. **Switch Security**
    a) Intra switch security
        i. How services authenticate and authorise when talking to each other
    b) Inter switch security (multiple switches scenario)
        i. How multiple switches authenticate and authorise when talking to each other
    c) Operator authorisation and authentication (identity management)
2. **Switch Infrastructure Security**
    a) Message topics and database authorisation
    b) Platform secrets (cluster) & Gateway secrets
    c) Data Protection - Kalka and Database Security
3. **Switch <-> FSP Security**
    a) How to secure FSPs authenticate and authorise when talking to each other
    b) Callbacks from the switch (Check for pain points from existing implementations)
4. **Secure Mechanisms for key/secret generation, storage, verification and invalidation**
    a) Provide a standardized way with HSM been the underlying support technology implementation.

Our approach is to provide OSS based solutions as pluggables with standard provided implementations so that implementers do not have to touch the OSS code in provisioning all these services.

Our HSM Deployment Approach

1) **Adhere to best practice HSM design, deployment & operational practices:**

   a. Key Management System (Master Keys, Session Keys, BDK's, CA, Certificates, etc..), Secure management processes (SOD's) & systems authentication
   a. FIPS 140-2 Level 3 Compliance, PCI DSS, CC, other regional standards
   b. Secure Crypto Processing (Payment vs General Purpose Functions),
   c. Infrastructure & Operational - Performance, High Availability, Scalability, etc..

2) **Support multiple open cryptographic standards:**

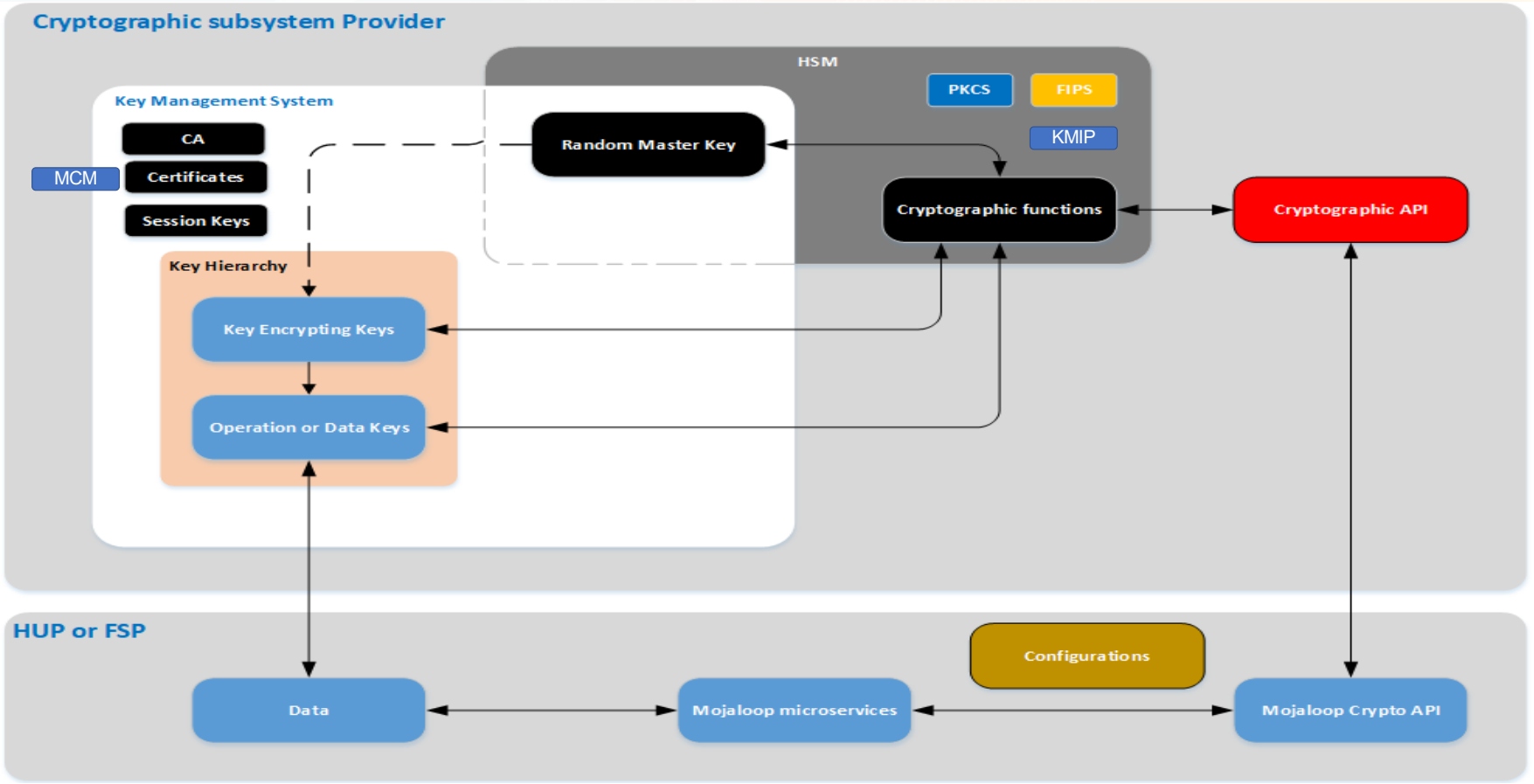
   a. Support both KMIP and PKCS#11 as the standardized interface to the KMS, HSM and CryptoAPI resources (frontend, backend and OpenAPI layer, respectively). HSM Vendor command independent and decoupling our deployment from vendor-provided SDK's is a must.

3) **Adopt open architecture integration methods:**

   a. Implement a bridge\adapter architecture for internal services consume security functionality from an internal service.
   b. Provide an open source solution, which in turn can implement several adapters for each external interface (for example talking to a specific HSM implementation or standard).

From core architecture perspective, HSM deployment should be flexible and open enough to support multiple use cases with minimal or no changes to the OSS code base

CryptoAPI Design Model



mojaloop

Thank you
Questions and Comments

Old Indian Proverb – " Work is not done by a magnificent plan or strategy; work is done when is done, and done by people"

mojaloop

Proposed PI 10 Objectives

1) HSM Implementation - Design, POC, Crypto API\Adapter & Use Case Implementation (ATM\POS OTM and Signatures).
1) Plan Implementation of the GDPR scope – Perform an assessment and Investigate options for addressing data protection requirements.
1) Baseline Mojaloop Security against PCI DSS Standard - Gap Analysis and Implementation Plan
2) Threat Modelling – Focusing on Parties, Quote & Transfer processes.
3) Develop new security standards for:
    a. Intra Switch Communication - How services authenticate and authorize when talking to each other
    b. Data Protection - Kalka and Database Security
    c. Key/secret generation, storage, verification and validation

NB. No input received so far from implementations projects however the door is always open.

mojaloop

Backup Slides

mojaloop

Mojaloop Application Security Summary

Core Quality\Security Core Team – Kim, Pedro, Lewis, Godfrey, Miguel, Sam & Victor

Community Contributors – Karim, Azeem & Aime

Application Security Initiative Overview

In order to address application security issues Mojaloop, we have a dedicated security stream consisting of the members the core team and community contributors with the key focus of building security into the development cycle of the Mojaloop platform as much as possible following the below principles.

Underpinning Principles:

- Deep integration into DevOps and CI\CD processes (All security must be automated gates to keep the DevOps workflow from slowing down)
- Developer centric and inclusive (Empower and delegate with trust security integration activities to the devops teams)
- Open source first approach to tooling (Commercial tools to be considered only if open source tools does not fulfil the requirements)
- Adapt Application Security to Cloud Native Technologies (Containers and Microservices)

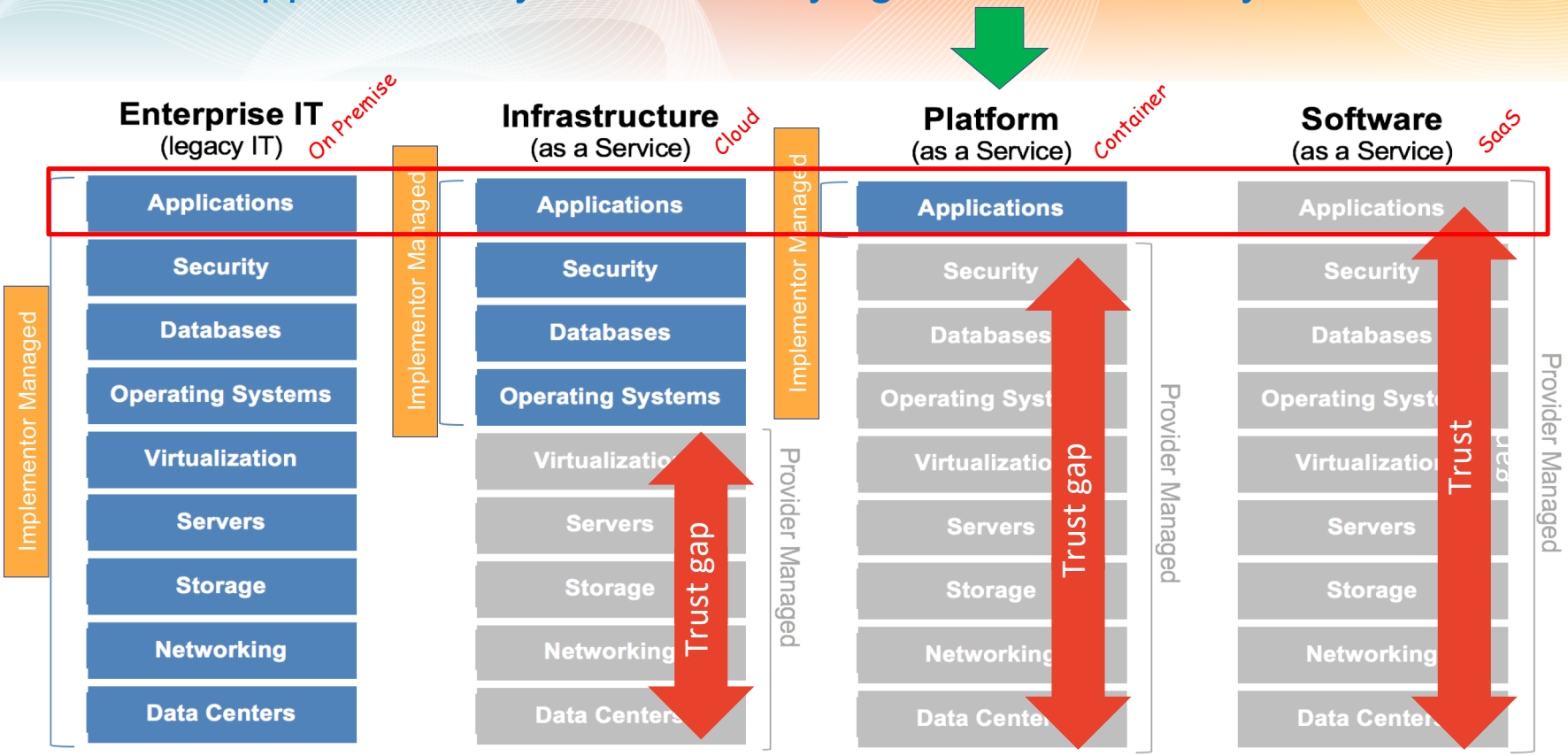
Our Prescriptive Security Approach

Prescriptive in the sense that we require the use of certain security practices and techniques (delivered as guidelines).

For some areas we will have reference technology implementations in place, like npm dependency, docker image vulnerability and license scanning, for other areas we require certain policies or standards to be adhered to and verifiable.

Mojaloop DevSecOps Approach

Focus is on the application layer not underlying infrastructure layer



Output:

- ***Provision of a security guidelines based on the Platform as a Service Model*** – Implementers to adapt to their own environment and policies
- Alignment to CIS benchmarks and CSA cloud security essentials for IaaS and PaaS

Application Security Domains

Our application security programme covers the four domains briefly descripted below.

1. Open Source Vulnerability Management – To enforce consistent use of non-vulnerable and non-obsolete open source components and dependencies.

2. Open Source License Compliance – To ensure compliance with Mojaloop open source license policy

1. Static Code Analysis – To ensure custom code is free of well known high and medium vulnerabilities before any code merging.

2. Container Security – To ensure secure image configuration and also find\fix in container based vulnerabilities and kerbunates applications

There are more solutions under investigation in this PI and will be added to these or new domains once they fully implemented, tested and signed.

Open Source Vulnerability Management

Objectives	Perform vulnerability assessment on open source components (composition analysis) before every commit, pull request and release as per our CI\CD pipeline workflow and security gates
Tools	1. NPM - https://docs.npmjs.com/cli/audit 2. Archery (Under Evaluation) 3. Snyk (Under Evaluation)
Policies and Standards	1. Default tool policies has been applied and busy investing best practices polices as a basis to adopt a Mojaloop specific policy 2. No standard has been adopted yet for this domain – Still under investigation
Recommendation to Implementors	1. Custom-tuned policies to your use case using the outcome of your threat and risk assessment.

Static Code Analysis

Objectives	Perform code level vulnerability assessment and code quality analysis with every commit, pull request and release as per our CI\CD pipeline workflow and security gates.
Tools	1. NodeJsScan - https://github.com/ajinabraham/NodeJsScan 2. SonarQube - https://github.com/SonarSource/sonarqube 3. Owasp Dependency Check - https://owasp.org/www-project-dependency-check/
Policies and Standards	1. Default tool policies has been applied and busy investing best practices polices as a basis to adopt a Mojaloop specific policy 2. No standard has been adopted yet for this domain – Still under investigation
Recommendation to Implementors	1. We advise implementors to custom-tuned their policies to your use case using the outcome of your threat and risk assessment.

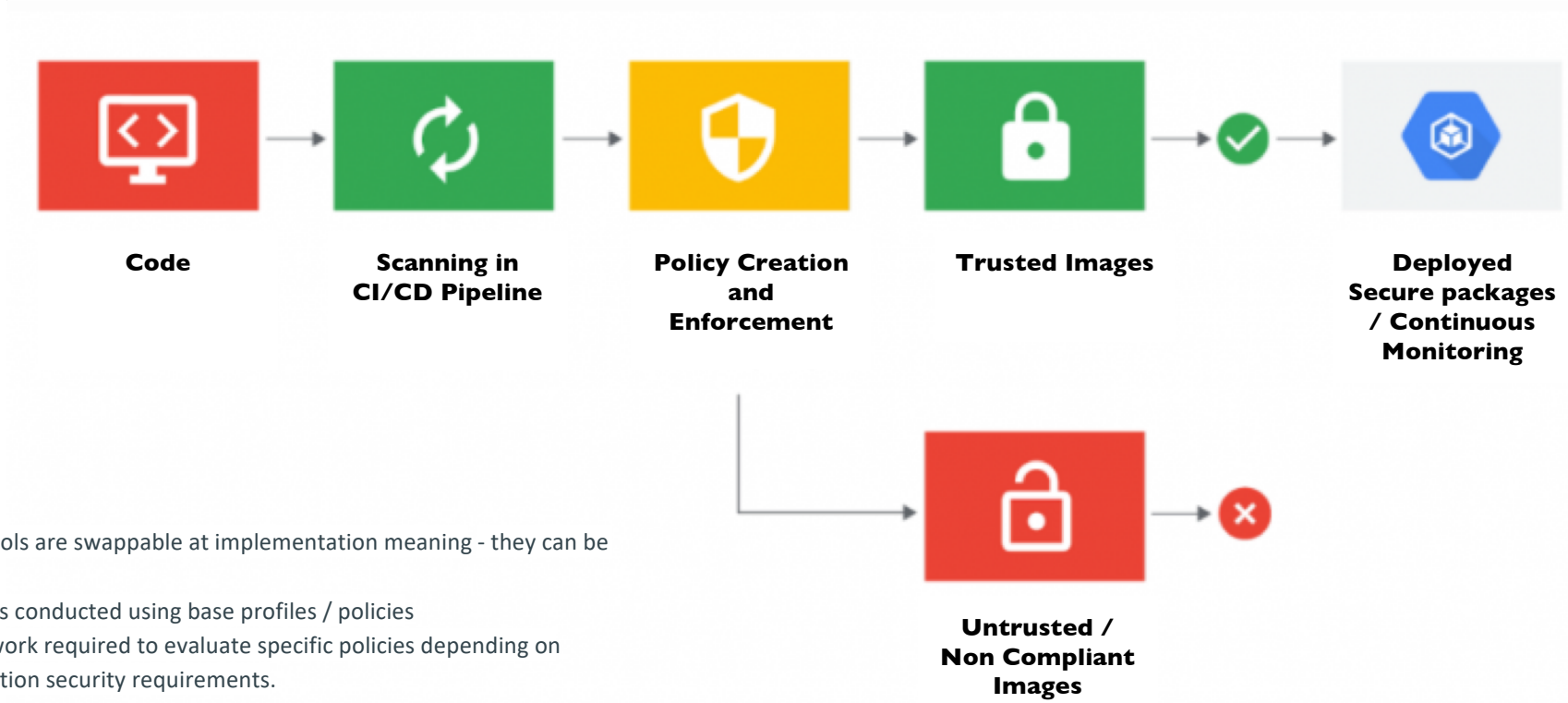
Container Security

Objectives	Ensure only secure container configuration is used in Mojaloop and find and fix container based vulnerabilities before any release.
Tools	1. Dockerfile Security 2. Anchore 3. AppArmor
Policies and Standards	1. Default tool policies has been applied and busy investing best practices polices as a basis to adopt a Mojaloop specific policy 2. CIS Cloud Security Benchmark (141 Checks)
Recommendation to Implementors	1. We advise implementors to custom-tuned their policies to your use case using the outcome of your threat and risk assessment.

Open Source License Compliance

Objectives	Ensure compliance with the Mojaloop open security license policy before every commit, pull request and release as per our CI\CD pipeline workflow and security gates
Tools	1. NPM License check 2. Snyk (under investigation)
Policies and Standards	1. Apache 2.0 and the ones compatible to it are allowed / conforms to the Apache 2.0, whereas GPL / GNU aren't?
Recommendation to Implementors	1. Custom-tuned policies to your use case using the outcome of your threat and risk assessment.

DevSecOps - How it all fits



- Proposed tools are swappable at implementation meaning - they can be replaced
- Current tests conducted using base profiles / policies
- Additional work required to evaluate specific policies depending on implementation security requirements.

mojaloop

More coffee stains

Thank You

For Queries feel free to contact

sam@modusbox.com
lewisd@crosslaketech.com
victor.akidiva@modusbox.com
godfreyk@crosslaketech.com

Oh no! A spill!