1. Introduction

Go to Crucible

Crucible provides a controlled and reproducible approach for constructing, hardening, assessing, transferring, deploying, and maintaining software and Infrastructure Environments. It combines Declarative Descriptions, composable Baselines, Machine Images, Infrastructure as Code (IaC), Dependency Capture, deployment automation, and compliance automation into a coordinated operational lifecycle.

Constructing, hardening, assessing, transferring, deploying, and maintaining an operational system requires substantial engineering effort. These activities are necessary to deliver a secure, compliant, reproducible, and supportable system, but they usually do not implement the mission-specific Functional Requirements that justify the project.

A project that develops these capabilities independently must dedicate skilled personnel, time, and funding to infrastructure automation, image construction, dependency capture, compliance assessment, evidence generation, deployment, disconnected operation, and lifecycle maintenance. Implementing these capabilities correctly can consume many person-years. The work also creates an ongoing obligation because the project must maintain the resulting tools, configurations, integrations, security controls, and evidence throughout the Operational Lifecycle of the system.

These expenditures compete directly with the resources available to develop, test, and improve the project’s mission capabilities. Although each project could create its own solution, repeated project-specific implementations duplicate effort, increase inconsistency, and create separate maintenance burdens.

Organizations that support multiple products often compound this problem. Individual product teams frequently create and maintain separate approaches to image construction, infrastructure automation, compliance assessment, deployment, dependency management, and lifecycle support. These approaches can differ in tools, conventions, interfaces, evidence formats, and operating procedures, even when the products must satisfy similar technical and governance obligations.

This inconsistency limits reuse across the organization. Software, Baselines, automation, evidence, and operational practices developed for one product can be difficult to apply to another. Personnel must learn different toolchains and procedures as they move between projects, reducing personnel mobility and making specialized skills harder to reuse. Organizations must therefore maintain additional teams, duplicate expertise, and spend more time integrating or reconciling product-specific solutions.

A centrally maintained Crucible capability provides a common operational foundation across products. Shared tooling, controlled terminology, reusable Baselines, consistent interfaces, and common lifecycle practices allow organizations to reuse both software and expertise. Product teams can move personnel between efforts more readily, reduce duplicated maintenance, and apply improvements across multiple products rather than implementing the same change independently for each system.

Crucible provides these shared capabilities as a controlled foundation. Projects can use Crucible rather than independently solving the same infrastructure, deployment, compliance, transfer, and lifecycle problems. This allows project teams to concentrate a greater share of their resources on the functionality that distinguishes the system and delivers its mission value.

Crucible supports operations in Connected Environments, Disconnected Environments, and Air-Gapped Environments. It preserves Traceability, Provenance, and Evidence across lifecycle stages and transfer boundaries.

Crucible supports Accredited Software Factory objectives, but Crucible does not grant Accreditation, Operational Approval, or an Authority to Operate (ATO). The responsible governance, assessment, and authorizing authorities make those determinations using the governing criteria, assessment results, and supporting Evidence.

Contents
