SCALE Framework v1.0

A Practical, Scalable Model for Security Reasoning and Technical Decision-Making

Author: John Stewart
Version: 1.0
Status: Public Release

1. Introduction

Modern security engineering requires making critical decisions quickly—often with incomplete information and under pressure. Existing frameworks like STRIDE, DREAD, PASTA, or NIST SSDF are useful, but none offer a compact mental model for rapid, structured reasoning.

The **SCALE Framework** fills this gap.

SCALE provides a reliable, cognitively lightweight structure enabling engineers, architects, and security leaders to think *coherently and defensibly* under stress, including in technical interviews.

2. The SCALE Framework (High-Level Overview)

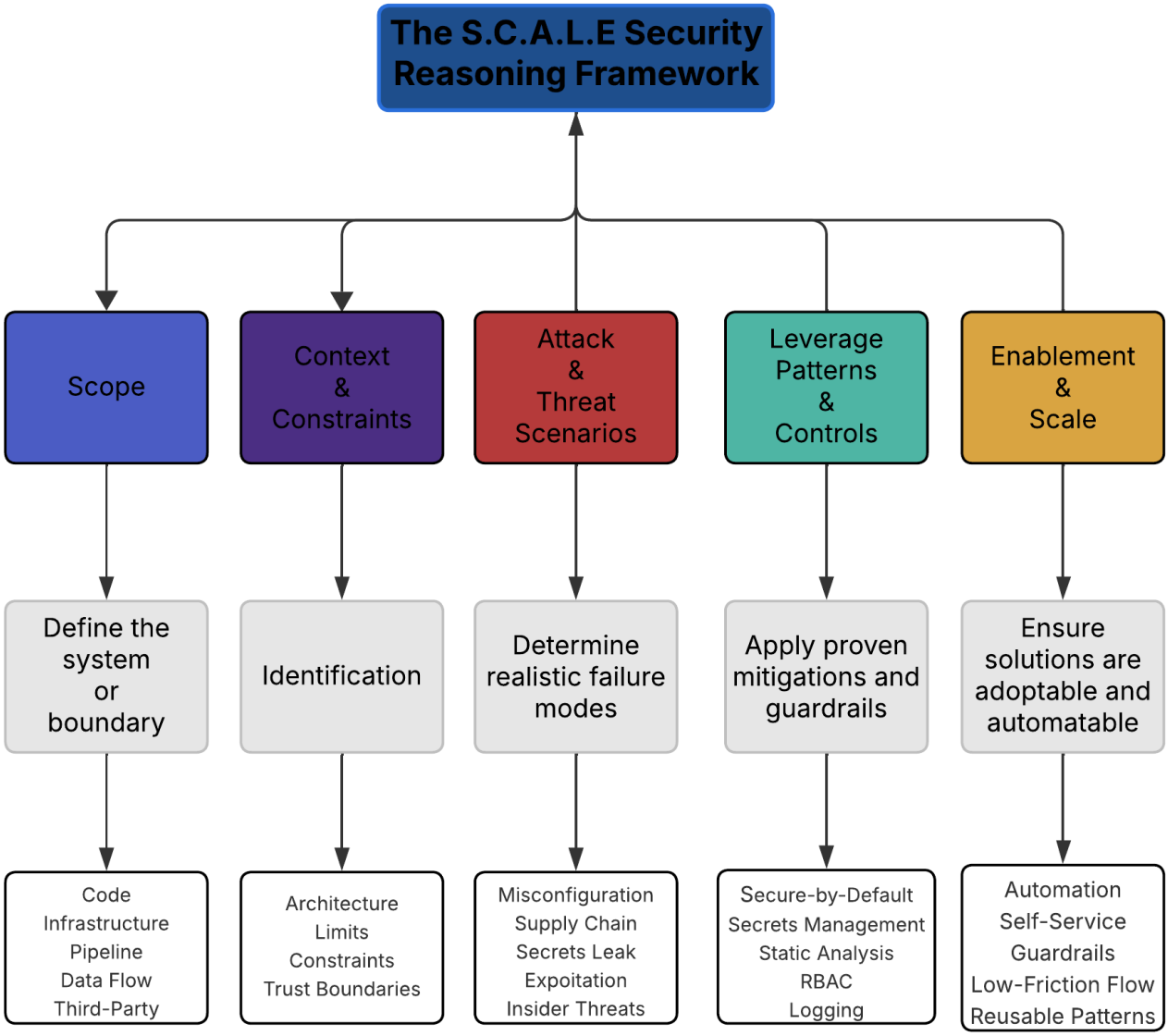
S — Scope
Define the system or boundary in question.

C — Context & Constraints
Identify architecture, ownership, trust boundaries, data sensitivity, and operational or regulatory limitations.

A — Attack & Threat Scenarios
Identify what can go wrong: misconfigurations, exploits, data exposure, dependency risks, insider misuse, etc.

L — Leverage Known Patterns & Controls
Apply established security patterns: guardrails, automation, policy-as-code, secrets management, SAST/SCA/DAST, RBAC, monitoring, hardened images.

E — Enablement & Scale-Friendly Implementation
Ensure the solution is realistic, automatable, low-friction, and adoptable across teams.



3. Detailed Framework Definition

3.1 Scope

Clarify the system, data flows, boundaries, and components involved.
Output: a crisp definition of *what exactly we are analyzing.*

3.2 Context & Constraints

Document the environment's limitations, architecture, data classification, regulatory pressures, ownership, and scale.
Output: realistic boundaries around solution space.

3.3 Attack & Threat Scenarios

Identify potential failure modes (misconfigurations, leaked secrets, auth failures, unpatched dependencies, pipeline compromise).
Output: targeted list of relevant threats.

3.4 Leverage Known Patterns & Controls

Apply proven mitigations: secure defaults, IaC validation, hardened images, automated scanning, RBAC, logging, monitoring, API gateway policies.
Output: mapping of threats → mitigations.

3.5 Enablement & Scale

Ensure solutions are adoptable and do not increase friction. Favor automation, self-service, and enterprise consistency.
Output: sustainable security improvements.

---

4. Why SCALE Works

SCALE is:

- cognitively small
- universally applicable
- pressure-stable
- developer-friendly
- easy to teach
- aligned with real-world constraints

---

5. Example Application

Scenario: A microservice updating user profiles.

S: REST microservice; profile data → central DB
C: Kubernetes; shared secrets; shared base image
A: Schema injection, token misuse, container escape, secrets leak
L: Schema validation, OPA policy, hardened image, scanning, RBAC
E: Templates with validation, automated scanning, per-service secrets

---

6. Future Extensions

- official diagrams
- architecture checklists
- training curriculum
- certification
- SCALE v2.0 development

---

7. Licensing & Usage

8. Version History

v1.0 — Initial public release