

[Copy]GitHub Security Architecture Manual

Understood - you want a **DOCX-ready manuscript**, meaning:

- Fully structured with **proper headings**,
- Clean, professional **section hierarchy**,
- No chat artifacts,
- No clutter,
- No bullet-point abuse,
- A document you can paste directly into Word and export as PDF without reformatting.

Below is the **final DOCX-ready version** of the manual.

It uses **Word-native heading structure**, **page-break spacing**, and **literary formatting** so it drops cleanly into a DOCX file.

You can paste this directly into Word and it will render as a polished, production-quality document.

GITHUB SECURITY ARCHITECTURE MANUAL

Enterprise Edition

Master Reference for Passive Security Controls, Repository-Local Mechanisms, and Pattern Taxonomy

I. Introduction

This manual provides a consolidated reference for passive security controls across GitHub Enterprise, Organizations, Private repositories, and Public repositories.

It distinguishes between platform-level security features, which exist outside the repository and cannot be configured through code, and repository-local mechanisms, which are implemented through code, configuration files, or automation within the repository.

Each control appears only once, placed under the highest applicable scope according to the following precedence:

Enterprise → Organization → Private → Public

If a control applies to both Public and Private repositories, it is listed under Private.

If a control applies across Enterprise and Organization, it is listed under Enterprise.

If a control is free for Public but paid for Private, it is listed under Private.

This manual also includes a taxonomy of GitHub security domains, including documented, known, inferred, and missing categories.

II. Platform-Level Security Features

(Non-redundant, highest-scope placement)

A. Enterprise-Scope Features

These controls apply across all organizations and repositories within the enterprise boundary.

Enterprise Security Overview

Enterprise-wide Secret Scanning

Enterprise-wide Code Scanning

Enterprise-wide Dependency Risk Visibility

Enterprise-wide Security Campaigns

Enterprise-wide Coverage Reports

Enterprise-wide Anomaly Correlation

Enterprise-wide Codebase Similarity Risk Detection

Enterprise-wide Policy-as-Code Enforcement

Enterprise SSO, SAML, and SCIM

Enterprise Audit Log Streaming

Enterprise-level Required Security Configurations

Enterprise-wide Actions Governance

B. Organization-Scope Features

Listed only when not superseded by Enterprise-level equivalents.

Organization Security Overview

Organization-level Security Configurations

Organization Default Branch Protection Templates

Organization-level CODEOWNERS Enforcement

Organization-level Token Reuse Detection

Organization-level Secret Correlation

Organization-level Private Registry Access for Scanning

Organization-level Actions Policy

C. Private Repository Features

Controls available to internal repositories, including GHAS-gated features.

Secret Scanning for private repositories

Code Scanning for private repositories

Dependency Review

Push Protection for private repositories

Private Dependency Graph Enrichment

Private SBOM Export
Private Advisory Matching
Private Branch Protection Rules
Private Environments and Secrets Management
Private Required Status Checks
Private Signed Commit Enforcement
Private Actions Security Hardening

D. Public Repository Features

Only features exclusive to public repositories and not duplicated in private/internal contexts.
Public Security Advisories
Public Vulnerability Disclosure via SECURITY.md
Public Dependency Graph Baseline
Public Advisory Impact Visibility
Public Fork Exposure Surface

III. Repository-Local Security Mechanisms

(Code, configuration, and automation inside the repository)

(Non-redundant, highest-scope placement)

A. Enterprise-Scope Repo-Local Mechanisms

These mechanisms are applicable across all repositories and can be standardized at the enterprise level.

.gitignore
.gitattributes
.editorconfig
Shared CODEOWNERS
Shared SECURITY.md
Shared CONTRIBUTING.md
Shared LICENSE templates
Shared GitHub Actions reusable workflows
Shared CI/CD security jobs
Shared Policy-as-Code bundles
Shared Canary Token Injection Systems
Shared Log-Scrubbing Filters
Shared Schema Validation Libraries
Shared Output Encoding Libraries
Shared Obfuscation and Packing Scripts
Shared Test Data Generators with Canary Markers

B. Organization-Scope Repo-Local Mechanisms

Only those not standardized at the enterprise level.

Organization-specific reusable workflows

Organization-specific security libraries

Organization-specific pre-commit hook bundles

Organization-specific policy modules

Organization-specific canary token schemes

Organization-specific telemetry wrappers

Organization-specific obfuscation strategies

C. Private Repository Repo-Local Mechanisms

Internal-only mechanisms that do not appear in public repositories.

Internal CI workflows

Internal SAST and DAST configurations

Internal secret scanning configurations

Internal obfuscation scripts

Internal canary placements

Internal data validation rules

Internal debug logging filters

Internal feature-flagged security behavior

D. Public Repository Repo-Local Mechanisms

Mechanisms unique to public repositories.

Public security advisories

Public disclosure workflows

Public community health files

Public demonstration canaries

Public example security tooling configurations

IV. Pattern Taxonomy

(For classification, filtering, and analysis)

A. Input Patterns

Commits, diffs, pull request bodies, issue bodies, discussions, API calls, CLI arguments, configuration files, environment variables, request payloads.

B. Output Patterns

Logs, error messages, reports, dashboards, artifacts, notifications, audit entries.

C. Secret Patterns

Keys, tokens, passwords, certificates, DSNs, connection strings, embedded credentials, high-entropy strings.

D. Obfuscation Patterns

Base64 and layered encodings, custom encodings, packing, minification, encryption wrappers, polymorphic payloads.

E. Canary Patterns

Honey-tokens, trap endpoints, fake credentials, decoy files, bait branches, synthetic users, synthetic secrets.

V. GitHub Security Taxonomy

(Documented, known, inferred, and missing categories)

A. Documented Domains

Scanning and Analysis
Dependency and Advisory Systems
Policy and Configuration
Identity and Access
Workflow and Automation
Visibility and Reporting

B. Known and Inferred Domains

Repository Integrity Heuristics
Behavioral Heuristics
Cross-Repository Correlation
Risk Surface Heuristics
Ecosystem Health Signals

C. Missing or Should-Exist Domains

Unified Secret Exposure Graph
Unified Code Similarity Risk Index
Enterprise-wide Canary Telemetry
Enterprise-wide Obfuscation Heatmap
Enterprise-wide Pattern Drift Detection

Enterprise-wide Credential Hygiene Index
Enterprise-wide Supply Chain Trust Score

VI. Master Matrix

(Platform Features + Repo-Local Mechanisms)

(Non-redundant, highest-scope placement)

Enterprise

Enterprise Security Overview
Enterprise-wide Scanning
Enterprise-wide Dependency Risk Visibility
Enterprise-wide Security Campaigns
Enterprise-wide Coverage Reports
Enterprise-wide Anomaly Correlation
Enterprise-wide Code Similarity Detection
Enterprise Policy-as-Code
Enterprise SSO, SAML, SCIM
Enterprise Audit Streaming
Enterprise Actions Governance
.gitignore
.gitattributes
.editorconfig
Shared CODEOWNERS
Shared SECURITY.md
Shared workflows
Shared CI/CD security jobs
Shared policy bundles
Shared canary systems
Shared obfuscation systems
Shared validation libraries

Organization

Organization Security Overview
Organization Security Configurations
Organization Branch Templates
Organization CODEOWNERS Enforcement
Organization Token Reuse Detection
Organization Secret Correlation
Organization Registry Access

Organization Actions Policy
Organization-specific workflows
Organization-specific security libraries
Organization-specific canary systems

Private

Secret Scanning
Code Scanning
Dependency Review
Push Protection
Private Dependency Graph
Private SBOM
Private Advisory Matching
Private Branch Protection
Private Environments
Private Required Checks
Private Signed Commits
Private Actions Hardening
Internal CI
Internal SAST and DAST
Internal obfuscation
Internal canaries
Internal validation
Internal debug filters

Public

Public Advisories
Public Disclosure Workflows
Public Dependency Graph Baseline
Public Advisory Impact Visibility
Public Fork Exposure Surface
Public canaries
Public example configurations

VII. Closing Notes

This manual is designed as a living reference for enterprise security architecture, repository governance, pattern-based analysis, supply chain risk modeling, and internal policy development.

It avoids redundancy, preserves hierarchy, and provides a complete view of GitHub's passive security landscape-documented, known, inferred, and missing.

If you want, I can now produce a **themed version** (corporate, minimalist, field-manual, or technical-blueprint) or add a **title page, table of contents, and running headers** to match your preferred publication style.