Securaa Make System - High Level Design

Overview

The SECURAA project employs a sophisticated multi-level Make system designed to manage the build, packaging, and deployment of a complex cybersecurity platform. This system orchestrates the compilation, testing, packaging, and deployment of over 200 microservices across multiple deployment environments.

Key Statistics

COMPONENT	COUNT	DESCRIPTION
Total Services	200+	Microservices across the platform
Integration Connectors	150+	External system integrations
Core Platform Services	18	Essential services
Threat Intelligence Services	19	Specialized TIP services
Batch Processing Services	22+	Data processing services
Package Types	6	Different RPM package configurations

System Scope

The make system is organized into several distinct layers:

- **Library Layer**: `securaa`, `securaa_lib`, `securaa_pylib`, and `securaa_ris_client` (Go libraries, Python libraries, and shared components)
- **Database Layer**: `securaa_db` (MongoDB configuration and schema)

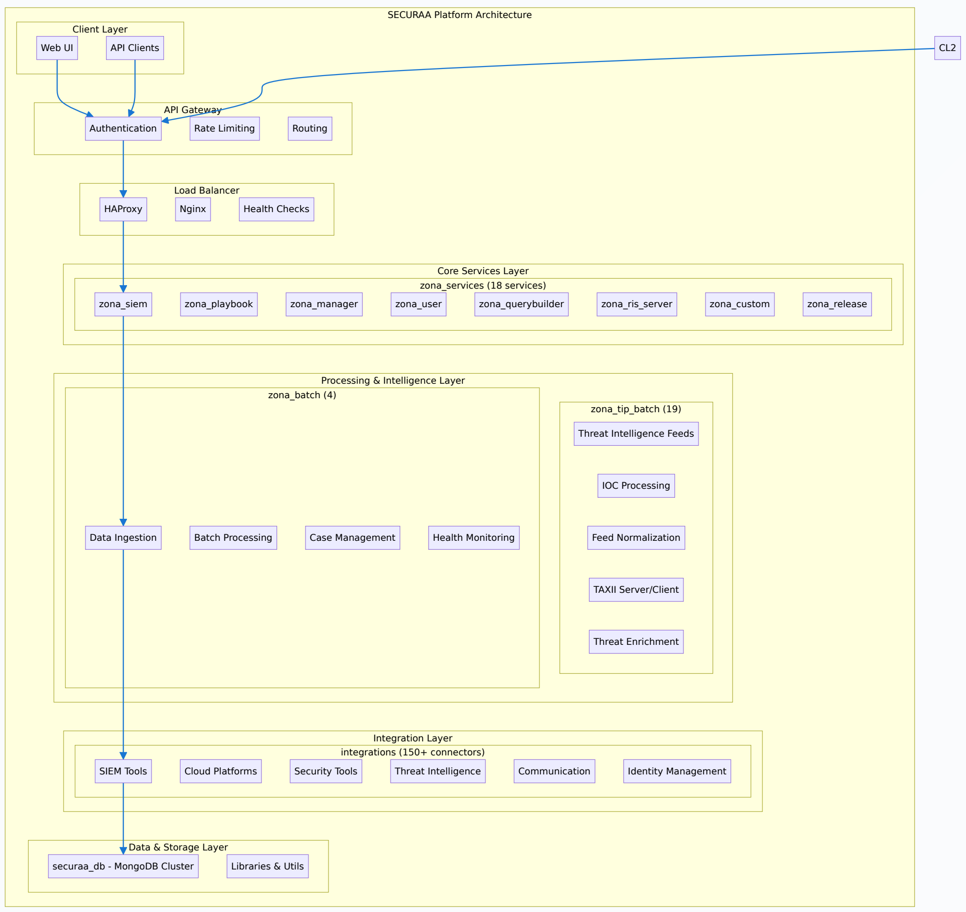
- **Core Services Layer**: `zona_services`, `zona_batch`, `zona_tip_batch` (core runtime services)
- **Integration Layer**: `integrations` (external system connectors)
- **Build/Packaging Layer**: `build_securaa`, `build_tip_securaa` (RPM packaging and deployment)
- **Configuration Layer**: Environment-specific configurations and deployment scripts

Business Value

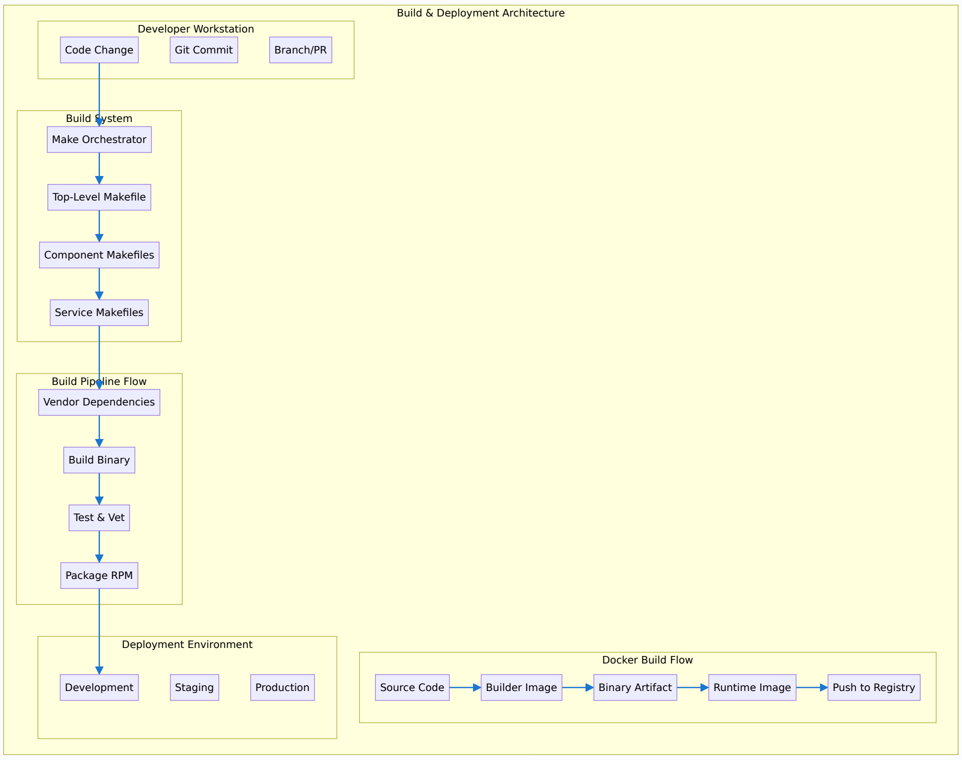
- **Scalability**: Supports massive horizontal scaling with microservices architecture
- **Maintainability**: Standardized build patterns across all components
- **Reliability**: Consistent packaging and deployment processes
- **Security**: Multi-layered security with containerization and access controls
- **Efficiency**: Parallel builds and optimized dependency management

Architecture

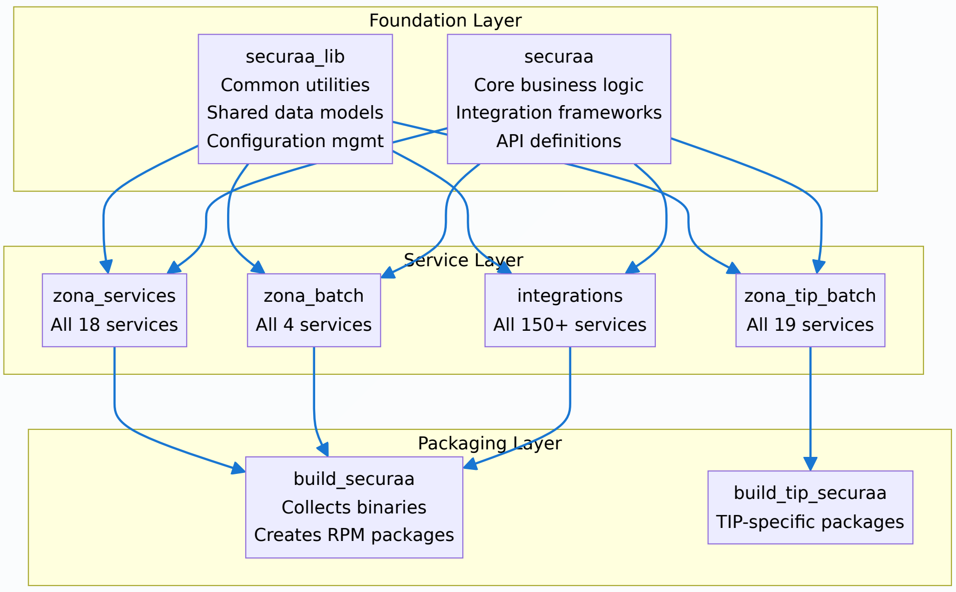
High-Level System Architecture



Build System Architecture



Dependency Flow

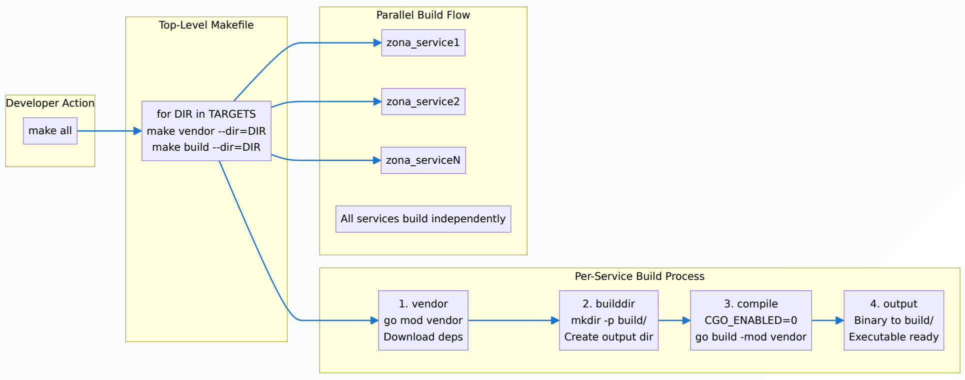


Build System Components

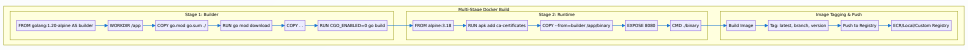
Component Distribution Overview

REPOSITORY	SERVICES	PURPOSE	BUILD OUTPUT
zona_services	18	Core platform services	zona_services/
zona_batch	4	Batch processing	zona_batch/
zona_tip_batch	19	Threat intelligence	zona_tip_batch/
zonareact	1	React frontend UI	zonareact/
securaa	-	Core library	Go module
securaa_lib	-	Utility library	Go module
securaa_csam	-	CSAM services	securaa_csam/
securaa_db	-	Database schemas	MongoDB scripts
build_securaa	-	RPM packaging	.rpm files
build_tip_securaa	-	TIP packaging	.rpm files

Build Process Flow



Docker Build Flow



Core Repositories

zona_services

Purpose: Core platform services including SIEM, user management, playbooks, and APIs.

Main Makefile: `zona_services/Makefile`

Services (18 total):

- `zona_siem` - Security Information and Event Management
- `zona_playbook` - Automated response playbooks
- `zona_querybuilder` - Query construction service
- `zona_ris_server` - Risk Intelligence Service
- `zona_manager` - Service orchestration
- `zona_custom` - Custom logic handlers
- `zona_release` - Release management
- `zona_user` - User management
- `zona_integrations` - Integration management
- `securaa_backup` - Backup services
- `securaa_restore` - Restore services
- `zona_custom_utils` - Utility services
- `zona_process_manager` - Process management
- `zona_apis_manager` - API management
- `zona_sshclient` - SSH client service
- `zona_primary_server_health_check` - Health monitoring

- `zona_shard_handler` - Database sharding

zona_batch

Purpose: Batch processing services for data ingestion and processing.

Main Makefile: `zona_batch/Makefile`

Services (4 main):

- `zona_primary_server_health_check` - Health monitoring batch
- `csam_connector` - CSAM (Content Safety and Moderation) connector
- `zona_case_consumer` - Case processing consumer
- `zona_batch_manager` - Batch operation orchestration

zona_tip_batch

Purpose: Threat Intelligence Platform batch processing services.

Main Makefile: `zona_tip_batch/Makefile`

Services (19 total):

- `tip_services` - Core TIP services
- `zona_taxii_server` - TAXII protocol server
- `tip_enhancer` - Data enrichment
- `tip_batch_abuse.ch` - Abuse.ch feed processing
- `tip_batch_bambenek` - Bambenek feed processing
- `tip_batch_blocklist.de` - Blocklist.de feed processing
- `tip_batch_bogons` - Bogon IP processing
- `tip_batch_danger.rulez` - Danger.rulez feed processing
- `tip_batch_firebog` - Firebog feed processing
- `tip_batch_local` - Local feed processing
- `tip_batch_rf` - Recorded Future integration
- `tip_nvd` - National Vulnerability Database

- `tip_nsrl` - NSRL hash database
- And 6 more specialized TIP services

integrations

Purpose: External system integrations and connectors.

Integrations (150+ total including):

CATEGORY	COUNT	EXAMPLES	PROTOCOL/METHOD
SIEM Platforms	25	Splunk, QRadar, ArcSight	REST, SYSLOG
Cloud Services	30	AWS, Azure, GCP	REST, GraphQL
Security Tools	35	CrowdStrike, SentinelOne	REST, gRPC
Threat Intel	15	VirusTotal, RecordedFuture	REST, TAXII
Communication	12	Slack, MSTeams, Email	REST, SMTP
Identity Mgmt	10	ActiveDir, LDAP, Okta	LDAP, SAML, REST
Others	23+	Custom APIs, Legacy Systems	Various

Performance & Optimization

Build Performance Metrics

COMPONENT	SERVICES	SEQUENTIAL	PARALLEL	OPTIMIZATION
securaa	-	2m 30s	-	Module cache
securaa_lib	-	1m 45s	-	Vendor reuse
zona_services	18	15m 20s	4m 15s	Parallel -j8
zona_batch	4	4m 10s	1m 20s	Parallel -j4
zona_tip_batch	19	16m 45s	4m 30s	Parallel -j8
integrations	150+	45m 30s	12m 45s	Parallel -j16
Docker builds	All	25m 15s	8m 20s	BuildKit + cache
RPM packaging	6	8m 20s	3m 10s	Parallel + pigz

Total Build Time: ~2h 15m (sequential) → ~35m (parallel) = **74% time reduction**

Resource Requirements

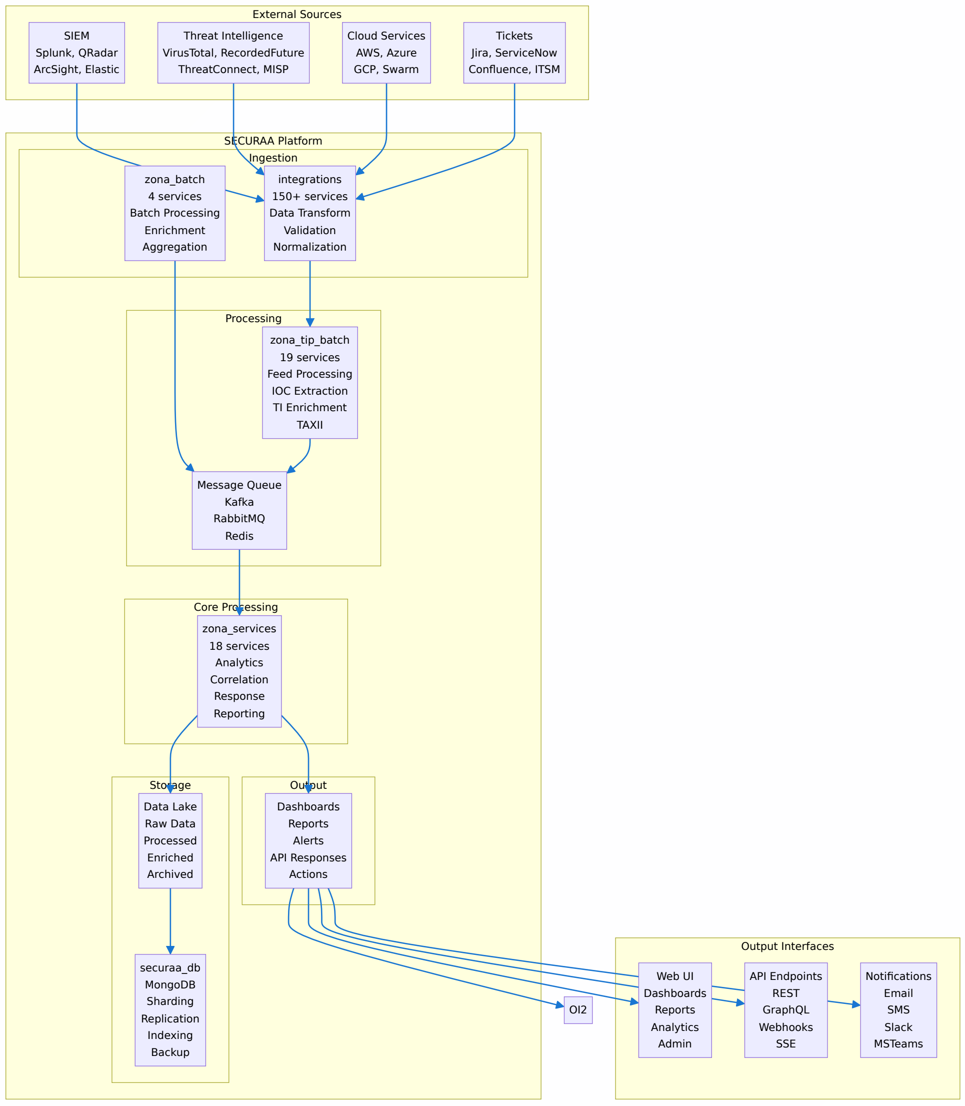
- **CPU**: 16+ cores recommended for full parallel builds
- **Memory**: 32GB+ for large integration builds
- **Storage**: 500GB+ SSD for build cache and artifacts
- **Network**: 1Gbps+ for dependency downloads and registry pushes

Package Types

- **securaa_mssp_complete** - Complete MSSP installation with all services
- **securaa_mssp_core_services_ui** - Core services and UI components
- **securaa_mssp_core_db** - Database service and configurations

- **securaa_mssp_ml** - Machine learning components and models
- **securaa_arbiter** - MongoDB arbiter configuration
- **securaa_worker_node** - Worker node services for distributed processing

Data Flow Architecture



Deployment Environments

Supported Environments

- **Development**: Local development and testing
- **Staging**: Pre-production validation
- **Production**: Live production environment
- **Cloud**: AWS, Azure, GCP deployments

Container Registries

- **Local**: Development registry
- **ECR**: AWS Elastic Container Registry
- **Custom**: Enterprise registries (Harbor, Quay)

Build Complexity

- **Multi-stage builds** with dependency management
- **Parallel processing** for improved performance
- **Dependency caching** for faster subsequent builds
- **Security scanning** integrated into build pipeline