

Configuration Guide

This guide covers all configuration options for the SOaC Framework.

Table of Contents

- [Environment Variables](#)
 - [Backend Configuration](#)
 - [Frontend Configuration](#)
 - [Database Configuration](#)
 - [Device Connectors](#)
 - [Detection Engine](#)
 - [SOAR Playbooks](#)
 - [Security Settings](#)
 - [Advanced Configuration](#)
-

Environment Variables

Backend Environment Variables

Location: `backend/.env` or set in your environment

```

# Database Configuration
DATABASE_URL=postgresql://soac_user:soac_password@postgres:5432/soac_db

# JWT Authentication
SECRET_KEY=your-secret-key-change-this-in-production-64-chars-minimum
ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=1440 # 24 hours

# Application Settings
ENVIRONMENT=development # development, staging, production
DEBUG=true # Enable debug mode (development only!)
LOG_LEVEL=INFO # DEBUG, INFO, WARNING, ERROR, CRITICAL

# CORS Settings
FRONTEND_URL=http://localhost:3000
CORS_ORIGINS=["http://localhost:3000","https://yourdomain.com"]

# API Settings
API_PREFIX=/api/v1
API_VERSION=1.0.0

# Event Ingestion
EVENT_COLLECTION_INTERVAL=300 # seconds (5 minutes)
EVENT_BATCH_SIZE=1000
EVENT_RETENTION_DAYS=90

# Detection Engine
CORRELATION_TIME_WINDOW=3600 # seconds (1 hour)
MIN_CORRELATION_PHASES=3
CONFIDENCE_THRESHOLD=0.7

# SOAR Settings
AUTO_EXECUTE_PLAYBOOKS=false # Require manual approval
PLAYBOOK_TIMEOUT=300 # seconds
MAX_CONCURRENT_PLAYBOOKS=5

# Mock Mode (for testing without real devices)
MOCK_MODE=false

# Rate Limiting
RATE_LIMIT_PER_MINUTE=60
RATE_LIMIT_PER_HOUR=1000

# Celery (Background Tasks)
CELERY_BROKER_URL=redis://redis:6379/0
CELERY_RESULT_BACKEND=redis://redis:6379/0

# Redis Cache
REDIS_URL=redis://redis:6379/1
CACHE_TTL=3600 # seconds

# Monitoring
ENABLE_METRICS=true
METRICS_PORT=9090

# Email Notifications (optional)
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASSWORD=your-app-password
SMTP_FROM=noreply@soacfabric.org

```

```
# Slack Notifications (optional)
SLACK_WEBHOOK_URL=https://hooks.slack.com/services/YOUR/WEBHOOK/URL

# Threat Intelligence (optional)
MISP_URL=https://your-misp-instance.com
MISP_API_KEY=your-misp-api-key
VIRUSTOTAL_API_KEY=your-virustotal-api-key
```

Frontend Environment Variables

Location: `frontend/.env` or set in your environment

```
# API Configuration
VITE_API_BASE_URL=http://localhost:8000

# Application Settings
VITE_APP_NAME="S0aC Framework"
VITE_APP_VERSION=1.0.0

# Feature Flags
VITE_ENABLE_DARK_MODE=true
VITE_ENABLE_NOTIFICATIONS=true

# Monitoring (optional)
VITE_SENTRY_DSN=https://your-sentry-dsn
VITE_GOOGLE_ANALYTICS_ID=UA-XXXXXXXXXX-X

# Development Settings
VITE_DEBUG=false
```

Backend Configuration

Application Configuration

Create `backend/app/config.py` for advanced settings:

```

from pydantic_settings import BaseSettings
from typing import List

class Settings(BaseSettings):
    # Database
    DATABASE_URL: str
    DATABASE_POOL_SIZE: int = 20
    DATABASE_MAX_OVERFLOW: int = 0

    # JWT
    SECRET_KEY: str
    ALGORITHM: str = "HS256"
    ACCESS_TOKEN_EXPIRE_MINUTES: int = 1440

    # Application
    ENVIRONMENT: str = "development"
    DEBUG: bool = False
    LOG_LEVEL: str = "INFO"

    # CORS
    FRONTEND_URL: str
    CORS_ORIGINS: List[str] = ["http://localhost:3000"]

    # Event Ingestion
    EVENT_COLLECTION_INTERVAL: int = 300
    EVENT_BATCH_SIZE: int = 1000
    EVENT_RETENTION_DAYS: int = 90

    # Detection
    CORRELATION_TIME_WINDOW: int = 3600
    MIN_CORRELATION_PHASES: int = 3
    CONFIDENCE_THRESHOLD: float = 0.7

    # SOAR
    AUTO_EXECUTE_PLAYBOOKS: bool = False
    PLAYBOOK_TIMEOUT: int = 300
    MAX_CONCURRENT_PLAYBOOKS: int = 5

    # Celery
    CELERY_BROKER_URL: str = "redis://redis:6379/0"
    CELERY_RESULT_BACKEND: str = "redis://redis:6379/0"

    # Redis
    REDIS_URL: str = "redis://redis:6379/1"
    CACHE_TTL: int = 3600

    class Config:
        env_file = ".env"
        case_sensitive = True

settings = Settings()

```

Logging Configuration

Create `backend/app/logging_config.py` :

```

import logging
from logging.handlers import RotatingFileHandler
import sys

def setup_logging(log_level: str = "INFO"):
    """Configure application logging"""

    # Create formatter
    formatter = logging.Formatter(
        '%(asctime)s - %(name)s - %(levelname)s - %(message)s',
        datefmt='%Y-%m-%d %H:%M:%S'
    )

    # Console handler
    console_handler = logging.StreamHandler(sys.stdout)
    console_handler.setFormatter(formatter)

    # File handler (rotating)
    file_handler = RotatingFileHandler(
        'logs/soac-framework.log',
        maxBytes=10485760,  # 10MB
        backupCount=10
    )
    file_handler.setFormatter(formatter)

    # Configure root logger
    root_logger = logging.getLogger()
    root_logger.setLevel(getattr(logging, log_level.upper()))
    root_logger.addHandler(console_handler)
    root_logger.addHandler(file_handler)

    # Suppress verbose libraries
    logging.getLogger("urllib3").setLevel(logging.WARNING)
    logging.getLogger("sqlalchemy.engine").setLevel(logging.WARNING)

```

Frontend Configuration

Vite Configuration

Edit `frontend/vite.config.ts`:

```

import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
  server: {
    host: '0.0.0.0',
    port: 3000,
    proxy: {
      '/api': {
        target: 'http://backend:8000',
        changeOrigin: true,
      },
    },
  },
  build: {
    outDir: 'dist',
    sourcemap: false,
    rollupOptions: {
      output: {
        manualChunks: {
          vendor: ['react', 'react-dom', 'react-router-dom'],
          mui: ['@mui/material', '@mui/icons-material'],
        },
      },
    },
  },
})
  
```

Theme Configuration

Edit frontend/src/theme/index.ts :

```

import { createTheme } from '@mui/material/styles';

export const lightTheme = createTheme({
  palette: {
    mode: 'light',
    primary: {
      main: '#1976d2',
    },
    secondary: {
      main: '#dc004e',
    },
  },
});

export const darkTheme = createTheme({
  palette: {
    mode: 'dark',
    primary: {
      main: '#90caf9',
    },
    secondary: {
      main: '#f48fb1',
    },
  },
});
  
```

Database Configuration

Connection Pool Settings

For production, tune PostgreSQL connection pool:

```
# In backend/.env
DATABASE_URL=postgresql://soac_user:soac_password@postgres:5432/soac_db?pool_size=20&max_overflow=10
```

Or in `backend/app/database.py`:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

engine = create_engine(
    DATABASE_URL,
    pool_size=20, # Number of connections to maintain
    max_overflow=10, # Additional connections when needed
    pool_pre_ping=True, # Verify connections before using
    pool_recycle=3600, # Recycle connections after 1 hour
)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
```

Database Maintenance

Create maintenance script `scripts/db-maintenance.sh`:

```
#!/bin/bash
# Database maintenance tasks

# Vacuum and analyze
docker-compose exec postgres psql -U soac_user -d soac_db -c "VACUUM ANALYZE;"

# Reindex
docker-compose exec postgres psql -U soac_user -d soac_db -c "REINDEX DATABASE soac_db;"

# Check table sizes
docker-compose exec postgres psql -U soac_user -d soac_db -c "
    SELECT
        schemaname,
        tablename,
        pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename)) AS size
    FROM pg_tables
    WHERE schemaname = 'public'
    ORDER BY pg_total_relation_size(schemaname||'.'||tablename) DESC;
"
```

Device Connectors

Palo Alto NGFW Configuration

```
# Device configuration in UI or via API
{
  "name": "PA-FW-01",
  "type": "paloalto",
  "config": {
    "hostname": "firewall.company.com",
    "api_key": "your-api-key",
    "port": 443,
    "verify_ssl": true,
    "timeout": 30
  }
}
```

Microsoft Entra ID Configuration

```
# Device configuration
{
  "name": "AzureAD-Tenant",
  "type": "entraid",
  "config": {
    "tenant_id": "your-tenant-id",
    "client_id": "your-client-id",
    "client_secret": "your-client-secret",
    "verify_ssl": true,
    "timeout": 30
  }
}
```

SIEM Configuration (Splunk)

```
# Device configuration
{
  "name": "Splunk-SIEM",
  "type": "siem",
  "config": {
    "url": "https://splunk.company.com:8089",
    "token": "your-api-token",
    "verify_ssl": true,
    "timeout": 60,
    "max_results": 1000
  }
}
```

See: [Device Integration Guide](#) (./DEVICE_INTEGRATION.md)

Detection Engine

Correlation Settings

Edit operational model configuration:

```
{
  "correlation_settings": {
    "time_window": 3600, // seconds
    "min_phases": 3, // minimum phases to match
    "entity_fields": ["user", "computer", "ip"],
    "confidence_weights": {
      "phase_count": 0.4,
      "source_diversity": 0.3,
      "temporal_proximity": 0.3
    }
  }
}
```

Custom Operational Models

Create custom models in `data/operational_models/` :

```
{
  "name": "Custom Threat Detection",
  "description": "Detect custom threat pattern",
  "use_case": "custom_threat",
  "phases": [
    {
      "name": "initial_access",
      "sources": ["entraid"],
      "queries": [...]
    },
    {
      "name": "execution",
      "sources": ["falcon"],
      "queries": [...]
    }
  ],
  "correlation": {
    "fields": ["user", "computer"],
    "time_window": 1800,
    "min_phases": 2
  }
}
```

See: [Operational Models Guide \(./OPERATIONAL_MODELS.md\)](#)

SOAR Playbooks

Playbook Configuration

Edit `backend/app/playbooks/config.py` :

```

PLAYBOOK_CONFIG = {
    "ransomwareContainment": {
        "enabled": True,
        "autoExecute": False, # Require approval
        "timeout": 300,
        "steps": [
            {
                "action": "isolateEndpoint",
                "deviceType": "falcon",
                "timeout": 60
            },
            {
                "action": "disableAccount",
                "deviceType": "entraID",
                "timeout": 30
            }
        ]
    }
}

```

Custom Playbooks

Create custom playbook in `backend/app/playbooks/custom_playbook.py`:

```

from .base_playbook import BasePlaybook

class CustomPlaybook(BasePlaybook):
    """Custom response playbook"""

    @async def execute(self, incident):
        """Execute playbook steps"""

        # Step 1: Custom action
        await self.custom_action(incident)

        # Step 2: Another action
        await self.another_action(incident)

        return {
            "status": "success",
            "actions_taken": [...]
        }

    @async def custom_action(self, incident):
        """Implement custom logic"""
        pass

```

See: [SOAR Playbooks Guide](#) (`./SOAR_PLAYBOOKS.md`)

Security Settings

JWT Configuration

```
# Strong secret key (64+ characters)
SECRET_KEY=$(python -c "import secrets; print(secrets.token_urlsafe(64))")

# Token expiration
ACCESS_TOKEN_EXPIRE_MINUTES=1440 # 24 hours
REFRESH_TOKEN_EXPIRE_DAYS=30

# Token algorithm
ALGORITHM=HS256
```

CORS Configuration

```
# Allowed origins
CORS_ORIGINS=["https://soac.company.com", "https://app.company.com"]

# Allowed methods
CORS_METHODS=["GET", "POST", "PUT", "DELETE", "PATCH"]

# Allowed headers
CORS_HEADERS=["Content-Type", "Authorization"]

# Allow credentials
CORS_CREDENTIALS=true
```

Rate Limiting

```
# backend/app/middleware/rate_limit.py
from fastapi import Request
from fastapi.responses import JSONResponse
import redis

redis_client = redis.from_url(Redis_URL)

async def rate_limit_middleware(request: Request, call_next):
    """Rate limiting middleware"""

    client_ip = request.client.host
    key = f"rate_limit:{client_ip}"

    # Check rate limit
    count = redis_client.incr(key)
    if count == 1:
        redis_client.expire(key, 60) # 1 minute window

    if count > 60: # 60 requests per minute
        return JSONResponse(
            status_code=429,
            content={"detail": "Too many requests"}
        )

    response = await call_next(request)
    return response
```

Advanced Configuration

Multi-Tenancy (Coming in v1.1)

```
# Enable multi-tenancy
MULTI_TENANT=true

# Tenant isolation level
TENANT_ISOLATION=strict # strict, shared

# Default tenant
DEFAULT_TENANT=tenant1
```

High Availability

```
# docker-compose.ha.yml
services:
  backend:
    deploy:
      replicas: 3
      update_config:
        parallelism: 1
        delay: 10s
      restart_policy:
        condition: on-failure

  postgres:
    deploy:
      replicas: 1
      placement:
        constraints:
          - node.role == manager
```

Monitoring & Metrics

```
# Prometheus metrics
ENABLE_METRICS=true
METRICS_PORT=9090
METRICS_PATH=/metrics

# Health checks
HEALTH_CHECK_INTERVAL=30 # seconds
HEALTH_CHECK_TIMEOUT=10 # seconds
```

Configuration Files Reference

Complete .env.example

```

# =====
# Database Configuration
# =====
DATABASE_URL=postgresql://soac_user:soac_password@postgres:5432/soac_db
DATABASE_POOL_SIZE=20
DATABASE_MAX_OVERFLOW=10

# =====
# JWT Authentication
# =====
SECRET_KEY=your-secret-key-change-this-in-production-64-chars-minimum
ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=1440
REFRESH_TOKEN_EXPIRE_DAYS=30

# =====
# Application Settings
# =====
ENVIRONMENT=production
DEBUG=false
LOG_LEVEL=INFO
API_PREFIX=/api/v1
API_VERSION=1.0.0

# =====
# CORS Settings
# =====
FRONTEND_URL=https://soac.company.com
CORS_ORIGINS=["https://soac.company.com"]
CORS_CREDENTIALS=true

# =====
# Event Ingestion
# =====
EVENT_COLLECTION_INTERVAL=300
EVENT_BATCH_SIZE=1000
EVENT_RETENTION_DAYS=90
MOCK_MODE=false

# =====
# Detection Engine
# =====
CORRELATION_TIME_WINDOW=3600
MIN_CORRELATION_PHASES=3
CONFIDENCE_THRESHOLD=0.7

# =====
# SOAR Settings
# =====
AUTO_EXECUTE_PLAYBOOKS=false
PLAYBOOK_TIMEOUT=300
MAX_CONCURRENT_PLAYBOOKS=5

# =====
# Background Tasks
# =====
CELERY_BROKER_URL=redis://redis:6379/0
CELERY_RESULT_BACKEND=redis://redis:6379/0

# =====
# Cache
# =====

```

```

REDIS_URL=redis://redis:6379/1
CACHE_TTL=3600

# =====
# Rate Limiting
# =====
RATE_LIMIT_PER_MINUTE=60
RATE_LIMIT_PER_HOUR=1000

# =====
# Monitoring
# =====
ENABLE_METRICS=true
METRICS_PORT=9090

# =====
# Notifications (Optional)
# =====
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=
SMTP_PASSWORD=
SMTP_FROM=noreply@soacframework.org

SLACK_WEBHOOK_URL=

# =====
# Threat Intelligence (Optional)
# =====
MISP_URL=
MISP_API_KEY=
VIRUSTOTAL_API_KEY=

```

Next Steps

- [Device Integration Guide](#) ([./DEVICE_INTEGRATION.md](#))
 - [Operational Models Guide](#) ([./OPERATIONAL_MODELS.md](#))
 - [SOAR Playbooks Guide](#) ([./SOAR_PLAYBOOKS.md](#))
 - [Security Best Practices](#) ([./SECURITY.md](#))
 - [Monitoring Guide](#) ([./MONITORING.md](#))
-

For deployment-specific configuration, see [Deployment Guide](#) ([./DEPLOYMENT.md](#))