SOaC Framework Architecture

Overview

The SOaC Framework is built on a modern, microservices-inspired architecture that enables scalable, maintainable, and extensible security operations automation.

Architecture Diagram

```
┌──────────────────────────────────────────────────────────────────────┐
│                       Security Infrastructure                        │
│                                                                      │
│  ┌────────────┐  ┌────────────┐  ┌────────────┐  ┌────────────┐      │
│  │ Palo Alto  │  │ Entra ID   │  │   SIEM     │  │  Falcon    │ ...  │
│  │   NGFW     │  │  (Azure)   │  │ (Splunk)   │  │   EDR      │      │
│  └─────┬──────┘  └─────┬──────┘  └─────┬──────┘  └─────┬──────┘      │
└────────┼───────────────┼───────────────┼───────────────┼─────────────┘
         │               │               │               │
         └───────────────┴───────────────┴───────────────┘
                         │
                         ▼
         ┌─────────────────────────────────┐
         │   API Integration Layer         │
         │                                 │
         │  ┌──────────────────────────┐   │
         │  │ Device Connectors        │   │
         │  │ - Base Connector         │   │
         │  │ - PaloAlto Connector     │   │
         │  │ - EntraID Connector      │   │
         │  │ - SIEM Connector         │   │
         │  └──────────────────────────┘   │
         └───────────────┬─────────────────┘
                         │
                         ▼
         ┌─────────────────────────────────┐
         │   Data Processing Layer         │
         │                                 │
         │  ┌──────────────────────────┐   │
         │  │ Event Ingestion          │   │
         │  │ - Normalization          │   │
         │  │ - Validation             │   │
         │  │ - Enrichment             │   │
         │  └──────────────────────────┘   │
         └───────────────┬─────────────────┘
                         │
                         ▼
         ┌─────────────────────────────────┐
         │      Detection Layer            │
         │                                 │
         │  ┌──────────────────────────┐   │
         │  │ Correlation Engine       │   │
         │  │ - Multi-phase detection  │   │
         │  │ - Entity tracking        │   │
         │  │ - Temporal correlation   │   │
         │  │ - Confidence scoring     │   │
         │  └──────────────────────────┘   │
         │                                 │
         │  ┌──────────────────────────┐   │
         │  │ Operational Models       │   │
         │  │ - Ransomware             │   │
         │  │ - Data Theft             │   │
         │  │ - Intrusion              │   │
         │  │ - Fraud                  │   │
         │  │ - DoS                    │   │
         │  └──────────────────────────┘   │
         └───────────────┬─────────────────┘
                         │
                         ▼
         ┌─────────────────────────────────┐
         │  Incident Management Layer      │
         │                                 │
         │  ┌──────────────────────────┐   │
         │  │ Incident Manager         │   │
         │  │ - Case creation          │   │
         │  │ - Enrichment             │   │
         │  │  - Assignment            │  │
         │  │  - Workflow              │  │
         │  └──────────────────────────┘  │
         └────────────────┬────────────────┘
                          │
         ┌────────────────▼────────────────┐
         │    Response Layer (SOAR)        │
         │                                 │
         │  ┌──────────────────────────┐  │
         │  │  Playbook Engine         │  │
         │  │  - Decision matrix       │  │
         │  │  - Action execution      │  │
         │  │  - Multi-device actions  │  │
         │  │  - Approval workflows    │  │
         │  └──────────────────────────┘  │
         │                                 │
         │  ┌──────────────────────────┐  │
         │  │  Playbook Library        │  │
         │  │  - Endpoint containment  │  │
         │  │  - Identity lockdown     │  │
         │  │  - Network containment   │  │
         │  │  - Cloud mitigation      │  │
         │  └──────────────────────────┘  │
         └────────────────┬────────────────┘
                          │
         ┌────────────────▼────────────────┐
         │    Presentation Layer           │
         │                                 │
         │  ┌──────────────────────────┐  │
         │  │  REST API (FastAPI)      │  │
         │  │  - OpenAPI/Swagger       │  │
         │  │  - JWT Authentication    │  │
         │  │  - RBAC                  │  │
         │  └──────────────────────────┘  │
         │                                 │
         │  ┌──────────────────────────┐  │
         │  │  Web UI (React)          │  │
         │  │  - Dashboard             │  │
         │  │  - Device management     │  │
         │  │  - Incident investigation│  │
         │  │  - Configuration         │  │
         │  └──────────────────────────┘  │
         └─────────────────────────────────┘
                          │
         ┌────────────────▼────────────────┐
         │      Data Layer                 │
         │                                 │
         │  ┌──────────────────────────┐  │
         │  │  PostgreSQL Database     │  │
         │  │  - Devices               │  │
         │  │  - Rules                 │  │
         │  │  - Events                │  │
         │  │  - Incidents             │  │
         │  │  - Users                 │  │
         │  │  - Audit logs            │  │
         │  └──────────────────────────┘  │
         │                                 │
         │  ┌──────────────────────────┐  │
         │  │  Redis Cache             │  │
         │  │  - Task queue            │  │
         │  │  - Session storage       │  │
         │  │  - Rate limiting         │  │
```

```
          │  └──────────────────────────┘  │
          └────────────────────────────────┘
```

Core Components

1. API Integration Layer

Purpose: Communicate with external security devices and collect data.

Components:
- **Base Connector**: Abstract base class with common functionality
- **Device-Specific Connectors**: Implementations for each platform
- PaloAlto Connector (PAN-OS REST API)
- EntraID Connector (Microsoft Graph API)
- SIEM Connector (Splunk/Elastic REST APIs)

Key Features:
- Connection pooling and retry logic
- Rate limiting and throttling
- Credential management
- Health checking
- Mock mode for testing

2. Data Processing Layer

Purpose: Normalize, validate, and enrich security events from different sources.

Components:
- **Event Ingestion Service**: Background service for continuous event collection
- **Normalization Engine**: Convert vendor-specific formats to common schema
- **Validation Service**: Ensure data quality and completeness
- **Enrichment Service**: Add context (geolocation, threat intel, asset info)

Common Event Schema:

```json
{
  "id": "uuid",
  "source": "device_name",
  "source_type": "paloalto|entraid|siem",
  "timestamp": "2025-11-14T10:00:00Z",
  "event_type": "authentication|network|process|file",
  "severity": "critical|high|medium|low|info",
  "entities": {
    "user": "john.doe@company.com",
    "host": "LAPTOP01",
    "ip": "192.168.1.100",
    "file": "malware.exe",
    "hash": "sha256:abcd1234..."
  },
  "raw_data": {...}
}
```

3. Detection Layer

Purpose: Correlate events across time and sources to detect complex attacks.

Components:

Correlation Engine

- **Multi-Phase Detection**: Track attack progression through multiple stages
- **Entity Tracking**: Follow users, hosts, IPs across events
- **Temporal Windowing**: Correlate events within time windows (5 min - 24 hours)
- **Confidence Scoring**: Calculate confidence based on matched phases

Correlation Logic:

```
# Pseudocode for correlation
def correlate_events(events, time_window, min_phases):
    entities = extract_entities(events)

    for entity in entities:
        event_chain = get_events_by_entity(entity, time_window)
        matched_phases = match_phases(event_chain, operational_model)

        if len(matched_phases) >= min_phases:
            confidence = calculate_confidence(matched_phases)
            create_incident(entity, event_chain, confidence)
```

Operational Models

Pre-built detection patterns for common attacks:
- **Ransomware**: Delivery → Execution → Encryption → Impact
- **Data Theft**: Collection → Staging → Exfiltration
- **Intrusion**: Foothold → Privilege Abuse → Lateral Movement
- **Fraud**: Compromise → Transaction → Exfiltration
- **DoS**: Flood → Degradation → Exhaustion

Model Structure:

```
{
  "name": "Ransomware Detection",
  "phases": [
    {
      "name": "Delivery",
      "sources": ["proofpoint", "email_gateway"],
      "indicators": ["attachment with .exe", "suspicious links"]
    },
    {
      "name": "Execution",
      "sources": ["falcon", "endpoint_edr"],
      "indicators": ["powershell -enc", "suspicious process tree"]
    },
    {
      "name": "Encryption",
      "sources": ["falcon", "file_monitoring"],
      "indicators": ["mass file rename", ".locked extension"]
    }
  ],
  "correlation_fields": ["user", "computer", "ip"],
  "time_window": "60 minutes",
  "min_phases": 3
}
```

4. Incident Management Layer

Purpose: Manage security incidents through their lifecycle.

Components:
- **Incident Manager**: CRUD operations for incidents
- **Workflow Engine**: State transitions and approvals
- **Assignment Engine**: Auto-assign based on severity/type
- **Enrichment Service**: Add threat intel and context

Incident Lifecycle:

```
New → Assigned → Investigating → Contained →
  Resolved → Closed
```

Incident Structure:

```
{
  "id": "INC-12345678",
  "title": "Ransomware Attack Detected",
  "severity": "critical",
  "confidence": "high",
  "status": "investigating",
  "assignee": "analyst@company.com",
  "operational_model": "ransomware",
  "matched_phases": ["delivery", "execution", "encryption"],
  "entities": {
    "user": "john.doe@company.com",
    "host": "LAPTOP01",
    "ip": "192.168.1.100"
  },
  "events": [...],
  "timeline": [...],
  "playbooks_executed": [...]
}
```

5. Response Layer (SOAR)

Purpose: Automate security response actions across multiple platforms.

Components:

Playbook Engine

- **Decision Matrix**: Determine which playbooks to execute
- **Action Executor**: Execute actions across devices
- **Approval Workflow**: Manual approval for sensitive actions
- **Rollback**: Revert actions if needed

Playbook Library

Standard playbooks for common responses:

1. **Endpoint Containment**
   - Isolate host from network
   - Kill malicious processes

   - Capture memory/disk for forensics
   - Tag device in CMDB

2. **Identity Lockdown**
   - Disable user account
   - Revoke active sessions
   - Reset MFA
   - Force password change

3. **Network Containment**
   - Block IP/domain at firewall
   - Enable packet capture
   - Rate limit connections
   - Create quarantine VLAN

4. **Cloud Mitigation**
   - Revoke API keys/tokens
   - Lock cloud resources
   - Snapshot for forensics
   - Roll back configuration

Playbook Structure:

```
{
  "name": "Ransomware Containment",
  "trigger_conditions": {
    "operational_model": "ransomware",
    "confidence": "high",
    "matched_phases": [">=3"]
  },
  "steps": [
    {
      "action": "isolate_endpoint",
      "device": "falcon_edr",
      "params": {"host": "${incident.entities.host}"}
    },
    {
      "action": "disable_account",
      "device": "entraid",
      "params": {"user": "${incident.entities.user}"}
    },
    {
      "action": "block_ip",
      "device": "paloalto_ngfw",
      "params": {"ip": "${incident.entities.ip}"}
    },
    {
      "action": "create_ticket",
      "device": "servicenow",
      "params": {"incident_id": "${incident.id}"}
    }
  ],
  "approval_required": false,
  "auto_execute": true
}
```

6. Presentation Layer

Purpose: Provide user interfaces for security operations.

Components:

REST API (FastAPI)

- **OpenAPI/Swagger Documentation**: Auto-generated API docs
- **JWT Authentication**: Stateless authentication
- **Role-Based Access Control**: Admin, Analyst, Viewer roles
- **Rate Limiting**: Prevent abuse
- **Audit Logging**: Track all API calls

Key Endpoints:
- `/api/v1/auth/*` - Authentication
- `/api/v1/devices/*` - Device management
- `/api/v1/rules/*` - Detection rules
- `/api/v1/events/*` - Event management
- `/api/v1/incidents/*` - Incident management
- `/api/v1/detection/*` - Detection engine
- `/api/v1/playbooks/*` - Playbook execution

Web UI (React + TypeScript)

- **Dashboard**: Real-time metrics and status
- **Device Management**: Configure and monitor devices
- **Rule Management**: Create and manage detection rules
- **Event Browser**: Search and filter events
- **Incident Investigation**: Full incident details and timeline
- **Operational Models**: View and configure detection patterns
- **Playbook Management**: Configure and test playbooks

7. Data Layer

Purpose: Persistent storage for all framework data.

Components:

PostgreSQL Database

Primary data store with tables for:
- **devices**: Security device configurations
- **rules**: Detection rules and mappings
- **events**: Collected security events
- **incidents**: Security incidents
- **users**: User accounts and roles
- **playbooks**: Response playbook definitions
- **audit_logs**: Complete audit trail

Redis Cache

Used for:
- **Task Queue**: Celery background jobs
- **Session Storage**: User sessions

- **Rate Limiting**: API rate limit counters
- **Caching**: Frequently accessed data

Data Flow

Event Collection Flow

```
1. Scheduled Task (every 5 min)
   ↓
2. Sync Service calls Device Connector
   ↓
3. Device Connector fetches events via API
   ↓
4. Events normalized to common schema
   ↓
5. Events validated and enriched
   ↓
6. Events saved to PostgreSQL
   ↓
7. Events sent to Correlation Engine
```

Detection Flow

```
1. New event arrives
   ↓
2. Correlation Engine extracts entities
   ↓
3. Query recent events for same entities
   ↓
4. Match events against Operational Models
   ↓
5. Calculate confidence score
   ↓
6. If threshold met, create Incident
   ↓
7. Incident triggers Decision Matrix
   ↓
8. Appropriate Playbooks executed
   ↓
9. Actions performed on devices
   ↓
10. Incident updated with results
```

Scalability Considerations

Horizontal Scaling

- **Backend API**: Stateless design allows multiple instances
- **Event Ingestion**: Can be split by device type
- **Correlation Engine**: Can process events in parallel
- **Database**: Read replicas for queries

Performance Optimizations

- **Database Indexing**: Optimized indexes on frequently queried fields
- **Caching**: Redis for frequently accessed data

- **Background Jobs**: Celery for async processing
- **Connection Pooling**: Reuse connections to devices and database

High Availability

- **Load Balancing**: Distribute traffic across backend instances
- **Database Replication**: PostgreSQL streaming replication
- **Redundant Workers**: Multiple Celery workers
- **Health Checks**: Kubernetes liveness/readiness probes

Security Architecture

Authentication & Authorization

- **JWT Tokens**: Short-lived access tokens
- **Refresh Tokens**: Long-lived refresh tokens
- **Role-Based Access**: Admin, Analyst, Viewer roles
- **API Keys**: For service-to-service communication

Secrets Management

- **Environment Variables**: For development
- **AWS Secrets Manager**: For AWS deployments
- **Azure Key Vault**: For Azure deployments
- **Kubernetes Secrets**: For K8s deployments

Network Security

- **TLS/SSL**: All communication encrypted
- **Network Segmentation**: Isolated network zones
- **Firewall Rules**: Whitelist-based access
- **VPN**: Secure access to management interfaces

Data Security

- **Encryption at Rest**: Database encryption
- **Encryption in Transit**: TLS for all connections
- **Credential Encryption**: Device credentials encrypted in DB
- **Audit Logging**: Complete audit trail

Technology Choices

Why FastAPI?

- Modern Python web framework
- Automatic OpenAPI documentation
- Built-in data validation (Pydantic)
- Async support for performance
- Type hints for better code quality

Why React + TypeScript?

- Modern, component-based UI
- Type safety reduces bugs

- Large ecosystem of libraries
- Excellent developer experience
- Strong community support

Why PostgreSQL?

- ACID compliance
- Powerful query capabilities
- JSON support for flexible schemas
- Strong ecosystem
- Proven reliability

Why Docker/Kubernetes?

- Consistent environments
- Easy deployment
- Horizontal scaling
- Service isolation
- Industry standard

Monitoring & Observability

Application Metrics

- Request rate and latency
- Error rates
- Event ingestion rate
- Incident creation rate
- Playbook execution success rate

Infrastructure Metrics

- CPU and memory usage
- Database connections
- Queue depth
- Network I/O
- Disk usage

Logging

- Structured JSON logs
- Centralized log aggregation
- Log levels (DEBUG, INFO, WARNING, ERROR)
- Correlation IDs for request tracking

Alerting

- Application health alerts
- Database performance alerts
- Integration failures
- Security events

Deployment Architecture

Docker Compose (Development)

```
┌─────────────────┐
│   Frontend      │ :3000
│   (React)       │
└────────┬────────┘
         │
┌────────▼────────┐
│   Backend       │ :8000
│   (FastAPI)     │
└────────┬────────┘
         │
    ┌────┴────┐
    ▼         ▼
┌────────┐ ┌──────┐
│Postgres│ │Redis │
└────────┘ └──────┘
```

Kubernetes (Production)

```
┌──────────────────────────┐
│     Ingress Controller   │
└────────┬─────────────────┘
         │
    ┌────┴────┐
    ▼         ▼
┌────────┐ ┌─────────┐
│Frontend│ │Backend  │ (3 replicas)
│Service │ │Service  │
└────────┘ └────┬────┘
                │
       ┌────────┴────────┐
       ▼                 ▼
┌─────────────┐   ┌──────────┐
│PostgreSQL   │   │Redis     │
│StatefulSet  │   │Deployment│
└─────────────┘   └──────────┘
```

Future Architecture Enhancements

Version 1.1

- Message queue (RabbitMQ/Kafka) for event streaming
- Elasticsearch for event search
- Grafana for advanced dashboards
- Multi-tenancy support

Version 2.0

- AI/ML for anomaly detection
- Graph database for entity relationships
- Real-time streaming analytics
- Advanced threat hunting

For implementation details, see the Development Guide (./DEVELOPMENT.md)