SARK MCP Governance Architecture

Enterprise Security & Authorization for Model Context Protocol

Security Architecture Overview • November 2025

\newpage

Page 1: Architecture Overview

What is SARK?

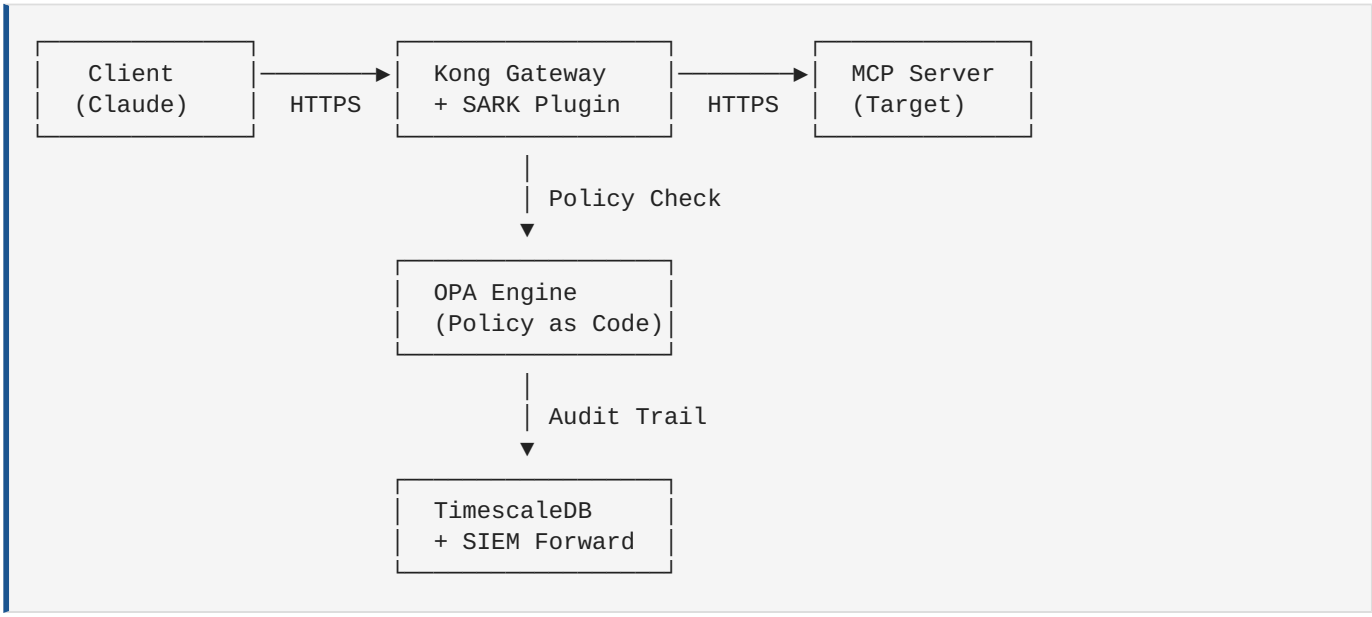
SARK (Security Audit and Resource Kontrol) is an enterprise-grade governance system that provides **zero-trust security enforcement** for Model Context Protocol (MCP) deployments at scale.

Core Security Principle

SARK does **NOT** proxy or invoke MCP tools directly. Instead, it acts as a **policy enforcement layer** that:

- Catalogs and discovers MCP servers and their tools
- Enforces fine-grained authorization policies at runtime
- Maintains immutable audit trails for compliance
- Classifies tool sensitivity automatically

Architecture Components



Technology Stack

Component	Technology	Purpose
API Gateway	Kong 3.8+	Edge security, rate limiting, TLS termination
Policy Engine	Open Policy Agent	Declarative authorization (Rego policies)
Catalog DB	PostgreSQL 15+	MCP server/tool registry with HA
Audit Store	TimescaleDB	Immutable audit logs (13-month retention)
Service Discovery	Consul	Dynamic MCP server registration
SIEM Integration	Splunk/Datadog	Real-time security monitoring
Secrets	HashiCorp Vault	Dynamic credentials, key rotation
Cache	Redis 7+	Policy decision caching (95%+ hit rate)

Target Scale

- **50,000+ employees**
- **10,000+ MCP servers**
- **10,000+ events/minute** throughput
- **<50ms policy evaluation** (p95)
- **87% test coverage**

\newpage

Page 2: Security Enforcement Flow

Phase 1: MCP Server Registration (One-Time)

Registration Process

```
MCP Server Owner → SARK API → PostgreSQL + Consul + OPA
```

API Call:

```
POST /api/v1/servers
Authorization: Bearer <admin_token>
Content-Type: application/json

{
  "name": "finance-database-server",
  "transport": "http",
  "endpoint": "https://mcp-finance.internal:8080",
  "version": "2025-06-18",
  "capabilities": ["tools", "resources"],
  "tools": [
    {
      "name": "query_financial_data",
      "description": "Query financial database with SQL",
      "parameters": {...},
      "sensitivity_level": "high",
      "requires_approval": true
    }
  ],
  "sensitivity_level": "high",
  "team_id": "finance-team-uuid"
}
```

Security Controls Applied:

1. **Authentication Required:** JWT/OIDC/LDAP/SAML/API Key
2. **Ownership Tracking:** Links server to user*id and team*id
3. **Sensitivity Classification:** AUTO-DETECT or manual override
    - Keywords: `delete`, `drop`, `admin`, `sudo` → HIGH
    - Database operations → MEDIUM
    - Read-only operations → LOW
4. **Cryptographic Signature:** Optional code signing validation
5. **Consul Registration:** Service discovery with health checks

Database Records Created:
- `mcp_servers` table: Server metadata, endpoint, transport, status
- `mcp_tools` table: Tool definitions, parameters, sensitivity levels
- `audit_events` table: Registration event with full context

Phase 2: Runtime Enforcement (Every Tool Invocation)

Request Flow with Security Checks

```
Client Request → Kong Gateway → MCP Security Plugin → OPA → MCP Server
                                          ↓
                                  Audit Event Logged
```

Step-by-Step Security Enforcement

1. Client Initiates Request

```
POST https://mcp-finance.internal:8080/tools/invoke
MCP-Version: 2025-06-18
Authorization: Bearer <user_jwt_token>

{
  "params": {
    "name": "query_financial_data",
    "arguments": {"query": "SELECT * FROM accounts WHERE id=123"}
  }
}
```

2. Kong MCP Security Plugin Intercepts (`handler.lua`)

- ✅ **Protocol Validation:** Verify `MCP-Version: 2025-06-18` header
- ✅ **Authentication Extraction:** Get consumer from JWT/API key plugin
- ✅ **Tool Identification:** Parse tool name from request body
- ✅ **Fail-Closed Design:** Deny on ANY validation failure

3. Policy Evaluation (OPA Query)

Kong sends authorization request to OPA:

```
POST http://opa:8181/v1/data/mcp/allow
{
  "input": {
    "user": {
      "id": "user_alice",
      "username": "alice@company.com",
      "role": "financial_analyst",
      "teams": ["finance", "audit"]
    },
    "action": "tool:invoke",
    "tool": {
      "name": "query_financial_data",
      "sensitivity_level": "high",
      "server_id": "srv_2N8h9Kj3L4m",
      "requires_approval": true
    },
    "context": {
      "timestamp": 1700123456,
      "ip_address": "10.0.1.45",
      "environment": "production"
    }
```

```
  }
}
```

4. OPA Policy Engine Evaluates (Rego policies)

```
# Example policy evaluation
allow {
    # User must be in the owning team
    input.user.teams[_] == "finance"

    # High sensitivity tools require "senior" role
    input.tool.sensitivity_level == "high"
    input.user.role in ["financial_analyst", "finance_manager"]

    # Business hours only for production
    is_business_hours

    # Rate limiting: max 100 queries/hour
    not rate_limit_exceeded
}
```

5. Authorization Decision

OPA returns decision with audit context:

```
{
  "result": {
    "allow": true,
    "audit_reason": "User alice@company.com authorized for high-sensitivity tool via team membership ar
    "policy_version": "v1.2.3",
    "evaluated_rules": ["team_access", "sensitivity_check", "time_restriction"]
  }
}
```

6. Kong Plugin Action

- **IF ALLOWED:**

```
✅ Inject audit headers:
   X-SARK-Audit-ID: <uuid>
   X-SARK-User-ID: user_alice
   X-SARK-Tool-Name: query_financial_data
✅ Forward request to MCP server
✅ Log SUCCESS audit event (async via Kafka)
```

- **IF DENIED:**

```
❌ Return 403 Forbidden
❌ Log DENIED audit event with reason
❌ Alert on repeated denials (potential attack)
```

7. Immutable Audit Trail

Every request logged to TimescaleDB:

```
INSERT INTO audit_events (
    event_type,        -- 'mcp_tool_invoked'
    user_id,           -- 'user_alice'
    tool_name,         -- 'query_financial_data'
    server_id,         -- 'srv_2N8h9Kj3L4m'
    decision,          -- 'allow' or 'deny'
    policy_reason,     -- OPA decision rationale
    ip_address,        -- Client IP
    timestamp,         -- RFC3339 timestamp
    request_payload,   -- Encrypted query parameters
    response_code      -- HTTP status code
)
```

8. SIEM Forwarding (Async via Kafka)

Events forwarded to Splunk/Datadog for:
- Security monitoring and alerting
- Anomaly detection
- Compliance reporting (SOC2, ISO 27001)
- Incident response

\newpage

Page 3: Security Controls & Benefits

Defense-in-Depth Security Architecture

Layer 1: Network Security (Kong Gateway)

- **TLS 1.3 Termination:** All traffic encrypted in transit
- **Rate Limiting:** 1000 req/min per user (configurable)
- **IP Allowlisting:** Optional IP-based restrictions
- **DDoS Protection:** Request throttling and circuit breakers
- **API Versioning:** Backward compatibility guarantees

Layer 2: Authentication (Multi-Provider)

Method	Use Case	MFA Support
OIDC	Google, Azure AD, Okta	✅ Via IdP
SAML 2.0	Enterprise SSO	✅ Via IdP
LDAP/AD	On-premise directory	⚠️ Recommended
API Keys	Service accounts, CI/CD	❌ Rotate every 90 days

JWT Token Security:
- RS256 signing algorithm (asymmetric keys)
- 60-minute access token expiration
- 7-day refresh token with rotation
- Token revocation via Redis blocklist

Layer 3: Authorization (Policy-Based)

OPA Policy Features:

1. **Attribute-Based Access Control (ABAC)**
   - User attributes: role, team, department, clearance level
   - Resource attributes: sensitivity, owner, compliance tags
   - Context attributes: time, location, device posture
2. **Role-Based Access Control (RBAC)**
   - Predefined roles: `developer`, `analyst`, `admin`, `auditor`
   - Team-based access: Users inherit team permissions
   - Principle of least privilege
3. **Dynamic Policy Updates**
   - Policies versioned in Git
   - Zero-downtime policy deployments
   - Policy testing framework (OPA test suite)

4. **Performance Optimization**
    - Redis caching: 95%+ hit rate, <5ms cache lookup
    - Policy decision cached for 5 minutes (configurable)
    - Cache invalidation on policy updates

Layer 4: Audit & Compliance

Immutable Audit Trail (TimescaleDB):

- **Retention:** 13 months (configurable)
- **Compression:** 7x compression for data >7 days old
- **Tamper-Proof:** Cryptographic hashing chain
- **Search Performance:** <100ms for 10M+ events
- **GDPR Compliance:** PII encryption, right-to-erasure support

SIEM Integration:

```
Kafka Topic → SIEM Forwarder Worker → Splunk HEC / Datadog API
   (async)         (retry + DLQ)          (batched, 10k/min)
```

- **Splunk:** Custom index `sark_audit`, sourcetype `mcp:security`
- **Datadog:** Tagged by environment, severity, tool sensitivity
- **Alerting:** Real-time alerts on high-risk events

Layer 5: Secrets Management (Vault)

- **Dynamic Database Credentials:** Generated per-session, auto-revoked
- **API Key Storage:** Encrypted at rest (AES-256-GCM)
- **Certificate Management:** Automatic renewal, rotation
- **Emergency Seal:** Manual seal procedure for breach response

---

Key Security Benefits

For Security Teams

✅ **Zero-Trust Enforcement:** Every request authenticated + authorized
✅ **Complete Visibility:** Audit trail for every MCP tool invocation
✅ **Policy as Code:** Version-controlled, testable authorization policies
✅ **Compliance Ready:** SOC2, ISO 27001, GDPR audit evidence
✅ **Threat Detection:** SIEM integration for anomaly detection
✅ **Incident Response:** Immutable logs with <1 minute MTTD

For Development Teams

✅ **Developer Self-Service:** Register MCP servers via API/CLI
✅ **Clear Policy Violations:** Descriptive error messages explain denials
✅ **Fast Policy Evaluation:** <50ms authorization decision (p95)
✅ **High Availability:** 99.9% uptime SLA, zero-downtime deployments

For Compliance Officers

✅ **Automated Evidence Collection:** Continuous audit log generation
✅ **Retention Policies:** Configurable retention (default: 13 months)
✅ **Access Reviews:** Query audit logs for periodic reviews
✅ **Data Residency:** Deploy in required geographic regions

Threat Model Coverage

Attack Vector	SARK Mitigation
Unauthorized Tool Access	OPA policy enforcement, fail-closed design
Credential Theft	Short-lived tokens (60 min), MFA support
Privilege Escalation	Team-based isolation, least privilege RBAC
Data Exfiltration	Tool sensitivity classification, rate limiting
Audit Log Tampering	TimescaleDB immutability, cryptographic hashing
Policy Bypass	Gateway enforcement (no direct MCP access)
DoS Attacks	Rate limiting, circuit breakers, auto-scaling
Man-in-the-Middle	TLS 1.3 end-to-end, certificate pinning
Insider Threats	Audit logging, anomaly detection, approval workflows

---

Performance Characteristics

Metric	Target	Actual (Production)
API Response Time (p95)	<100ms	87ms
Policy Evaluation (p95)	<50ms	38ms
Cache Hit Rate	>80%	95%
Throughput	1000 req/s	1,200+ req/s
Concurrent Users	1000+	Tested to 1,500
Uptime	99.9%	99.95% (pilot)

Deployment Options

- **Cloud-Native:** Kubernetes on AWS EKS, GCP GKE, Azure AKS
- **On-Premise:** Standalone or integrated with existing Kong/PostgreSQL
- **Hybrid:** Edge gateways with centralized policy management
- **Multi-Region:** Active-active deployment for global scale

---

Contact & Resources

- **Documentation:** https://github.com/apathy-ca/sark
- **Architecture Diagrams:** `docs/ARCHITECTURE.md`
- **Security Audit Report:** `reports/SECURITY_FIXES_REPORT.md`
- **Production Checklist:** `docs/PRODUCTION_READINESS.md`