Securing the Full API Stack
The API Stack

- Includes runtime and design time components
- Not all API products will involve all components
- Some components may appear more than once in practice
  - Firewalls
  - Facades
- Not all possible systems are included
  - Log Management and SIEM
  - Visualization
  - Monitoring
CDN

**Role:** Static content delivery, edge cache, DDoS protection

**Threats and Risks**
- Provider overload
- False positives
- Improper caching

**Mitigations**
- Only store non-sensitive data
- Delegate to API Gateway
- Stringent Cache-Control headers
OAuth Server

**Role:** Issuance and validation of Bearer and ID Tokens

**Threats and Risks**
- Brute force attacks
- Information leakage
- Web site attacks (CSRF, XSS, etc.)

**Mitigations**
- Delegate lockouts to identity provider
- Throttle OAuth endpoints
- *Use OpenID Connect*
- Keep it simple
Identity Provider

Role: Authentication, attestation of claims

Threats and Risks
- Brute force attacks
- Information leakage
- Incorrect configuration

Mitigations
- Lockouts and true 2FA
- Limit to API access from OAuth Server
- Hash all credentials
Developer Portal

**Role:** Developer engagement, self-service API access

**Threats and Risks**
- Granting access to too many APIs
- Granting access to wrong environment
- Web site attacks (CSRF, XSS, etc.)

**Mitigations**
- Workflow for approvals to production environment
- Patch, patch, patch
API Gateway

**Role:** Access control, transformation, analytics

**Threats and Risks**
- Denial of service attacks
- Improper access to APIs

**Mitigations**
- OAuth2 and OpenID Connect for authentication validation
- Role-based access control using claims
- Throttle traffic and employ whitelists and blacklists
- Limit complexity of policies
- Certificate pinning and **TLS**
Gateway Backside Cache

**Role:** Response and data caching

**Threats and Risks**
- Incorrect caching and invalidation
- Cross-API cache data access

**Mitigations**
- Authorize *before* reading from cache
- Use separate caches for sensitive data
- Send and enforce Cache-Control headers
Firewalls

**Role:** Network boundary enforcement

**Threats and Risks**
- Cross-network access
- Unauthorized service traffic

**Mitigations**
- Define clear API network boundaries
- Enforce using mutual TLS and IP restrictions (when possible)
- Implement IDS to detect anomalous traffic
# Facade

**Role:** Transformation, orchestration, business logic

## Threats and Risks
- Unauthorized service traffic
- Invalid data inputs
- Unauthorized access to data

## Mitigations
- Pass claims from Gateway to enforce attribute-based access control
- Use mutual TLS to enforce access only from Gateway
- **Data Validation Is Everyone’s Responsibility**
Microservice Aligned Data Store

Role: CQRS query handling

Threats and Risks
• Data exfiltration
• Stale data access
• Lost writes

Mitigations
• Use TLS and authentication mechanisms
• Only store data you actually need
• Update with triggers or use as write-through cache
**Backend Service**

**Role**: Business logic, legacy system access

**Threats and Risks**
- Unauthorized service traffic
- Invalid data inputs
- Unauthorized access to data
- Limited ability to change

**Mitigations**
- Use mutual TLS to enforce access only from facades
- Delegate ABAC to façade layer
- **Data Validation Is Everyone’s Responsibility**
**Backend Data Store**

**Role:** System of record for data

**Threats and Risks**
- Invalid data
- Improper data access
- Limited ability to change

**Mitigations**
- Enforce authentication and authorization of backend service callers
- Block access with network segmentation
- Encrypt data at rest and in motion
General principles

- Don’t reinvent the wheel
- Don’t be insecure just because your clients are
- Don’t use old stuff when newer solutions exist
- Don’t fly blind without monitoring
- Don’t trust anyone!