RSA CONFERENCE ASIA PACIFIC 2013

SECURING REST APIS— PROTECTING THE NEW WORLD OF MOBILE AND CLOUD SERVICES, USING OAUTH AND API KEYS

John Thielens
Axway

Session ID: SPO-W03A

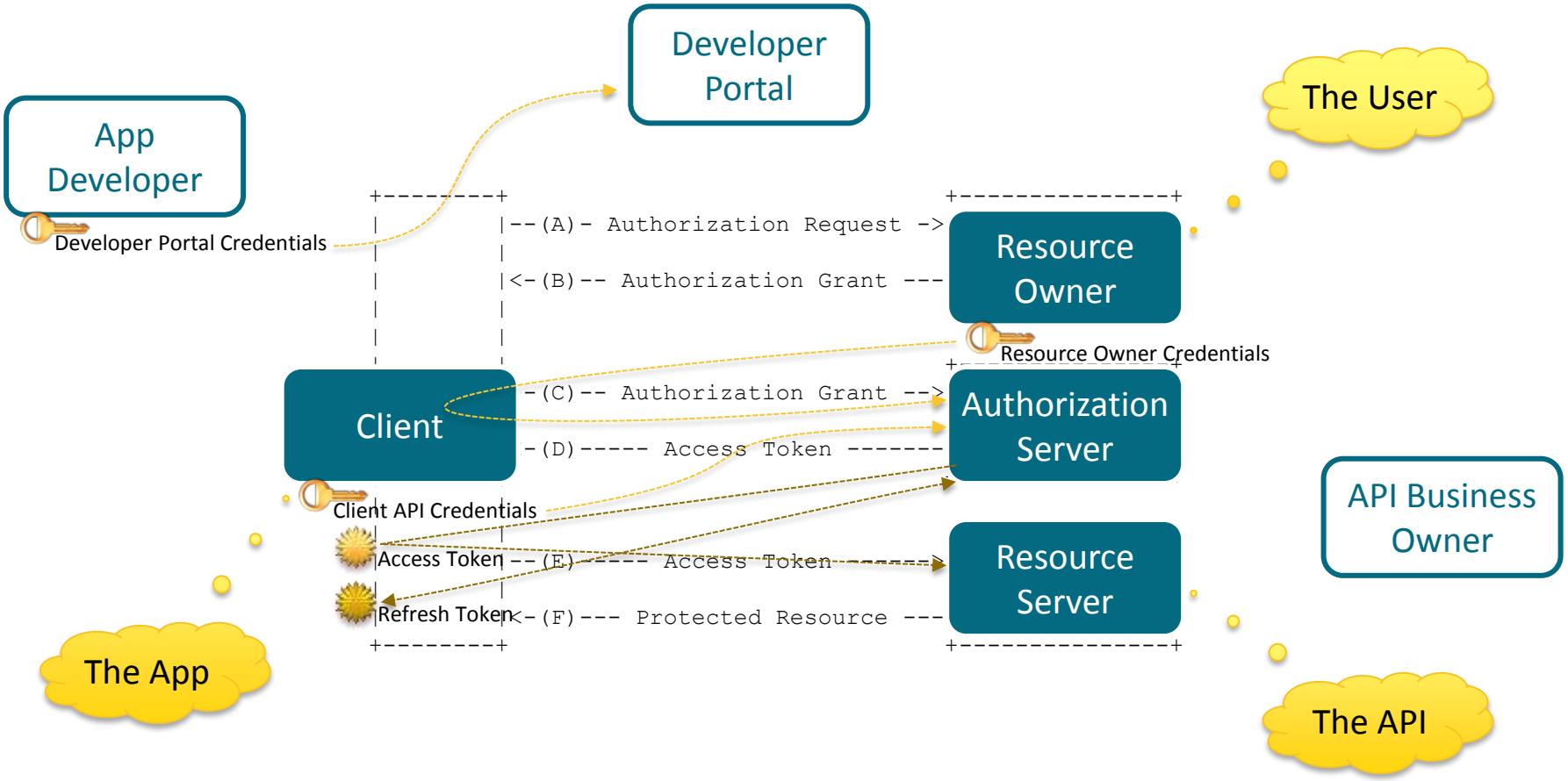
Session Classification: Intermediate

REST—The Cultural Evolution

- REST Culture (as evolved from ESB)
  - Developer self-service – open lifecycle
  - Usage context – Open API, Enterprise API, Vendor API
  - Deployment context – public cloud, private cloud, on premise
  - Focus on explicit state management / stateless APIs (nouns)
  - Constrained interaction model (verbs)

- REST Security (as evolved from Web Applications)
  - Explicit actor model – add developer and app to user and API
  - “public” security model – assume browser or app is compromised
  - Tiered defensive strategy

OAUTH Actor Model



API Identities and Keys

- Identities and Credentials
  - The User (Resource Owner)
    - Issued by…federated in many scenarios
  - The App (Client)
    - Issued through the Developer Portal by the API Owner
  - The Developer
    - Issued through the Developer Portal by the API Owner
  - The API Owner
    - Federated to the Enterprise / Web Site Owner
- Keys/Tokens
  - Access Tokens (typically “short”-lived)
  - Refresh Tokens (long-lived or indefinite)

The TLS Slide

- REST Security in general and OAuth mechanisms in particular depend on TLS
  - Use TLS 1.0, 1.2 if you can
  - Handle certificates properly, even in sample code

Threats

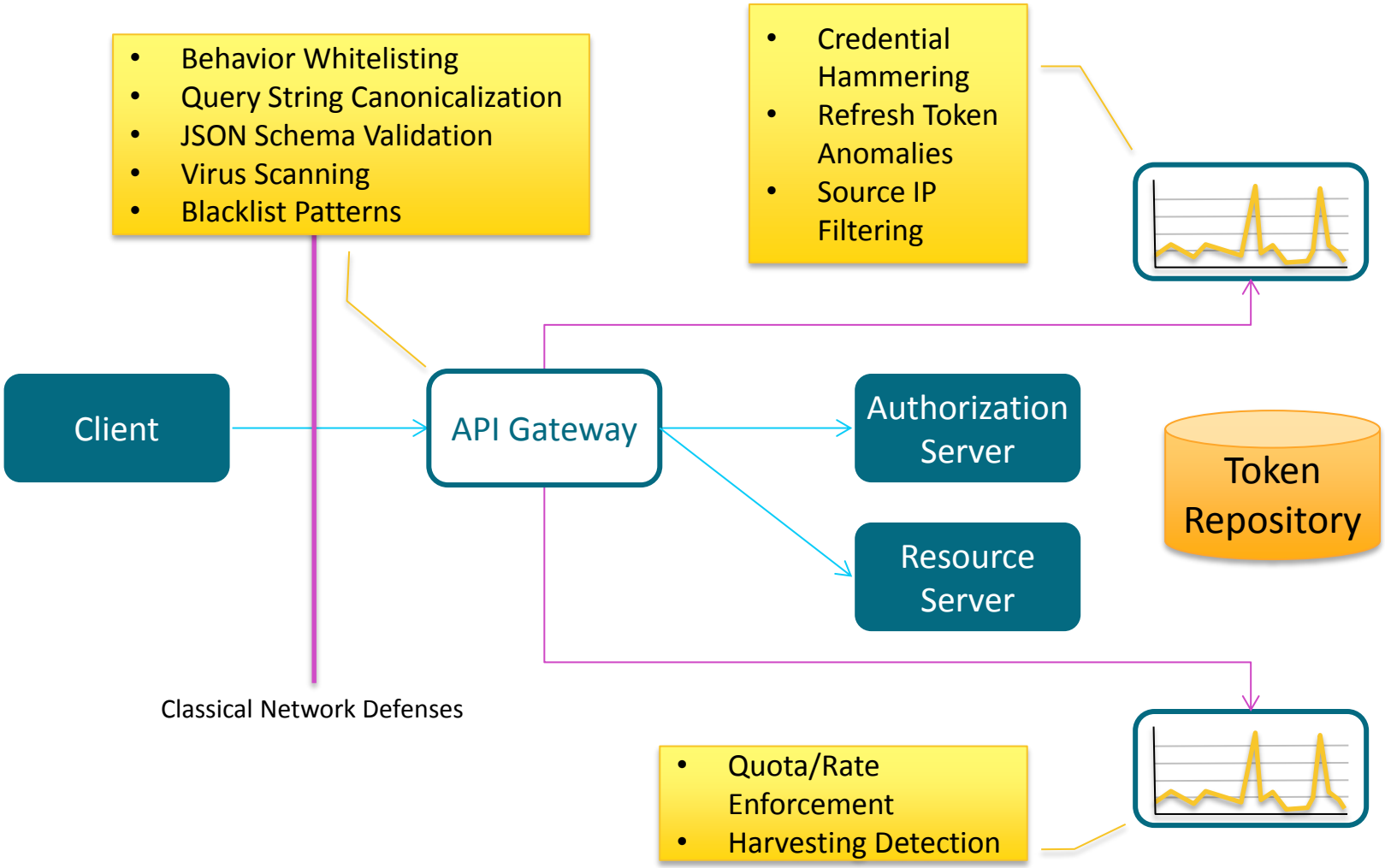
- Service Provider
  - Access control violations (data protection)
  - Availability (scale, responsiveness)
  - Identity management (life cycle)
- Application Developer
  - Client (application) keys, identifiers
  - Integrity of access and refresh tokens
  - Application and Developer reputation
- Application / Service Consumer
  - Man-in-the-middle vulnerability
  - Malicious responses

Web Threats, REST Edition

- Injection attacks (XSS, SQL, Xpath, Xquery)
- Buffer Overflow
- (D)DoS attacks
- XML attacks
- JSON attacks
- Session attacks / CSRF

- APIs are responsible for managing inbound and outbound threats

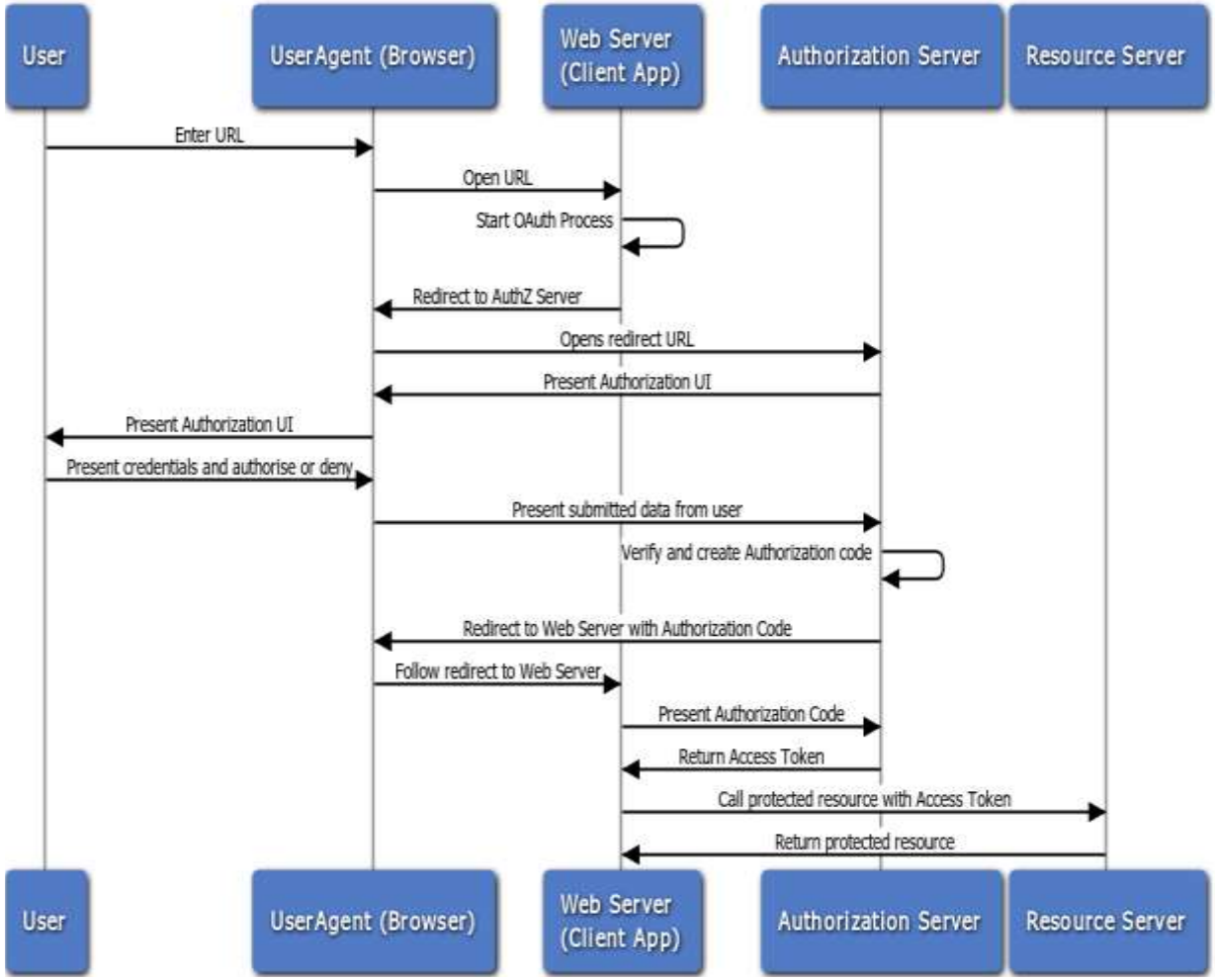
Tiered Countermeasures



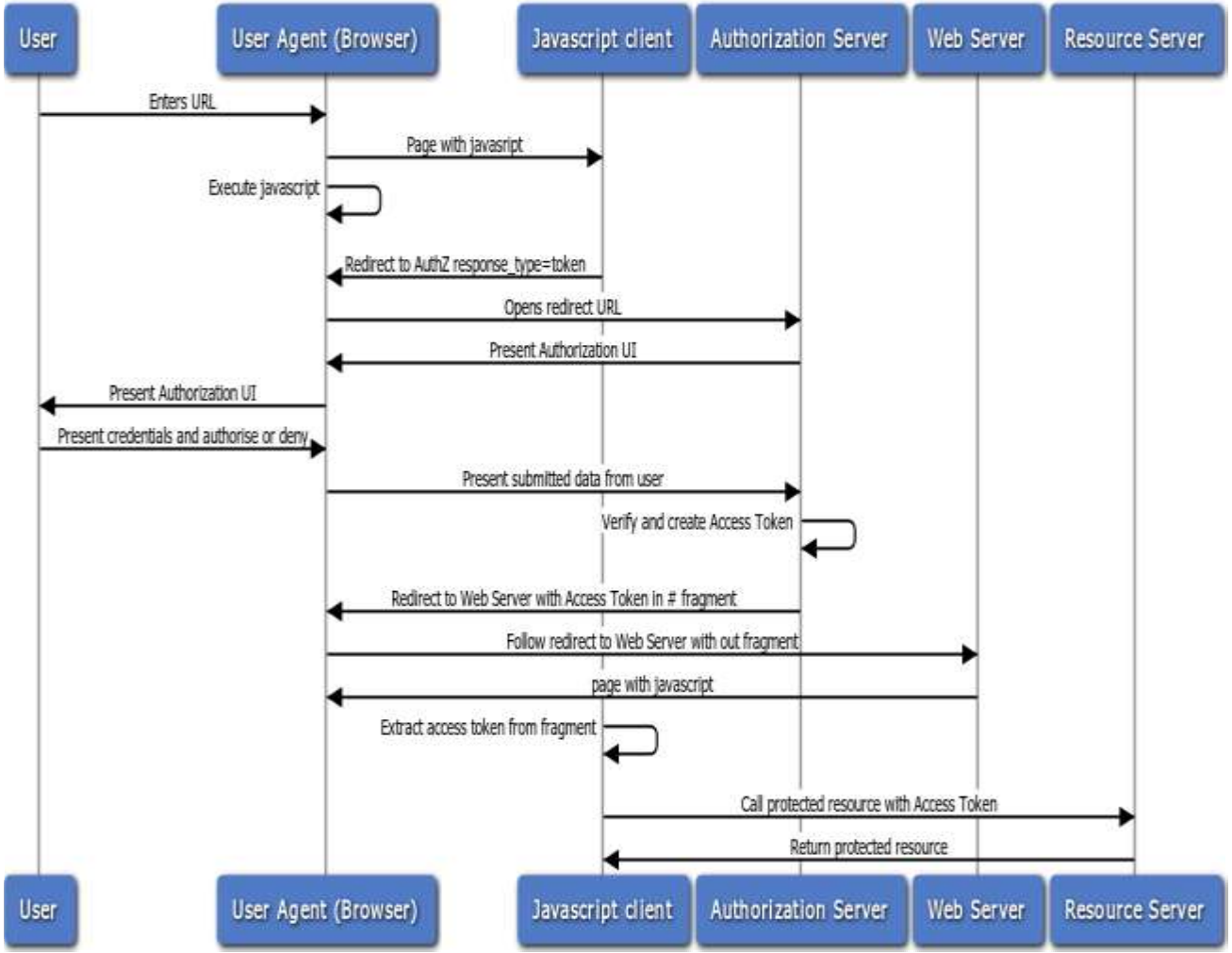
Authorization

- OAuth manages authorization using *scopes*
  - Scope management can be delegated to an API Gateway
- JSON Web Tokens
  - Can be used in cross-domain contexts
  - Requires signatures, and associated certificate management
  - Alternate mechanism to obtain access tokens, possibly with scope

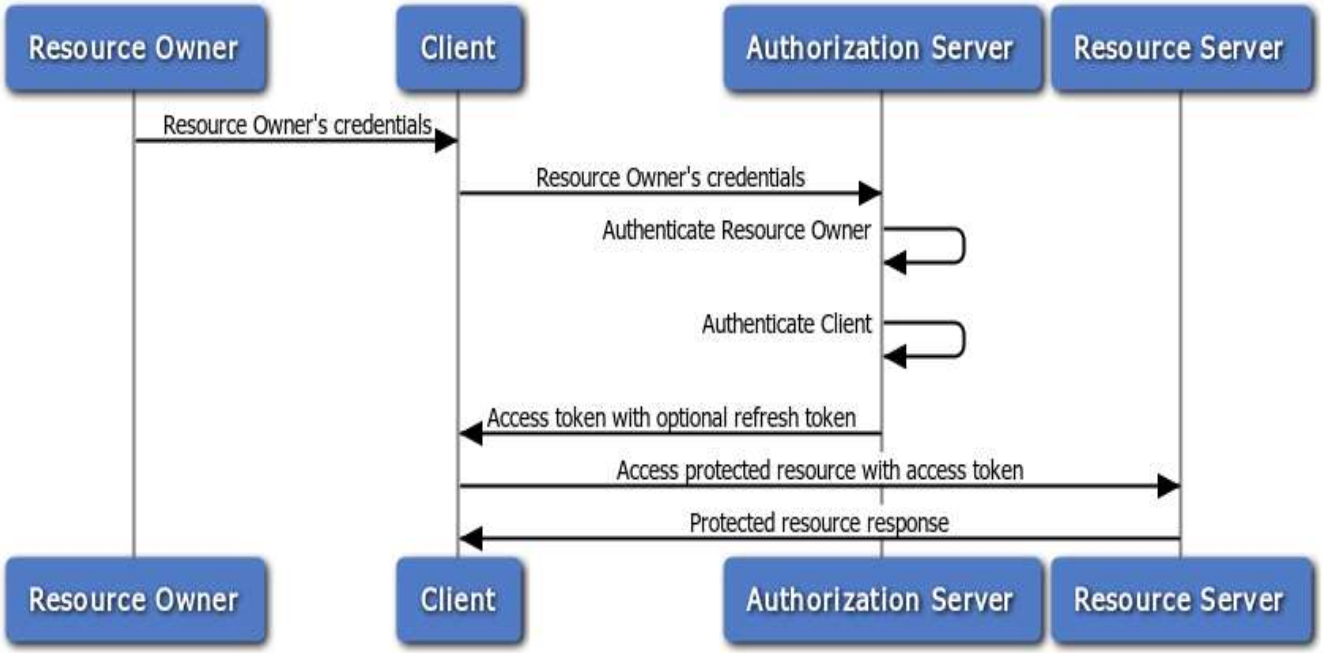
Example: OAuth 2.0 Auth Code Grant Flow



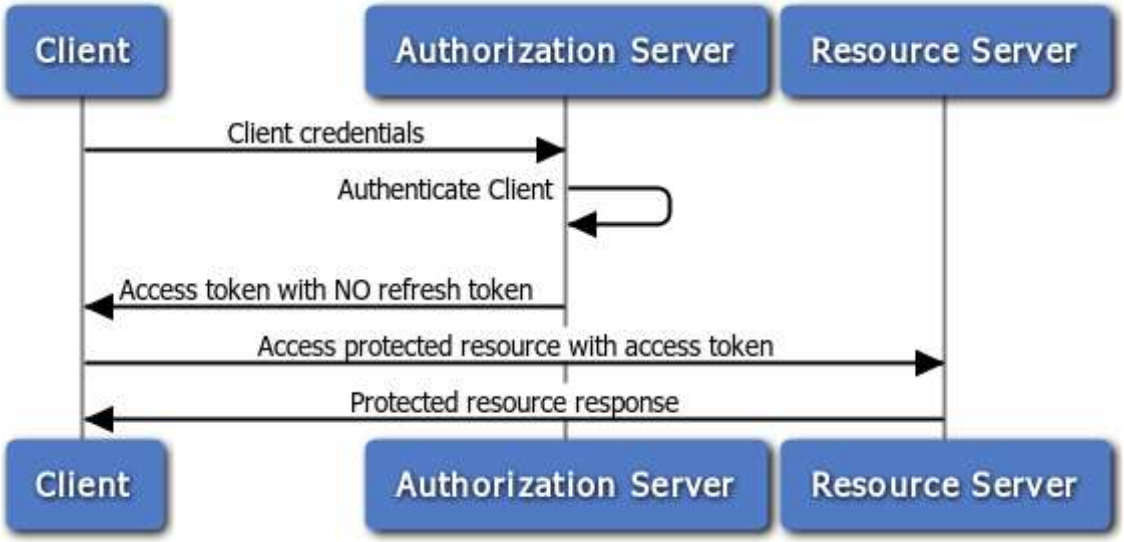
Example: OAuth 2.0 Implicit Grant Flow



Example: OAuth 2.0 Resource Owner Password Credentials Flow



Example: OAuth 2.0 Client Credentials Flow



Example: OAuth 2.0 JWT Flow

Summary

- REST Security…for whom?
  - API Provider, API Consumer, API Developer
- Life Cycle Processes (keep Self-Service in mind)
  - Credential registration, maintenance, revocation
  - Access Token and Refresh Token revocation
- Visibility and Monitoring
  - Authorization server abuses / hammering
  - API usage tracking (correlation with client_id)
- Threat prevention
  - Prefer whitelisting, where possible

RSA CONFERENCE
ASIA PACIFIC 2013