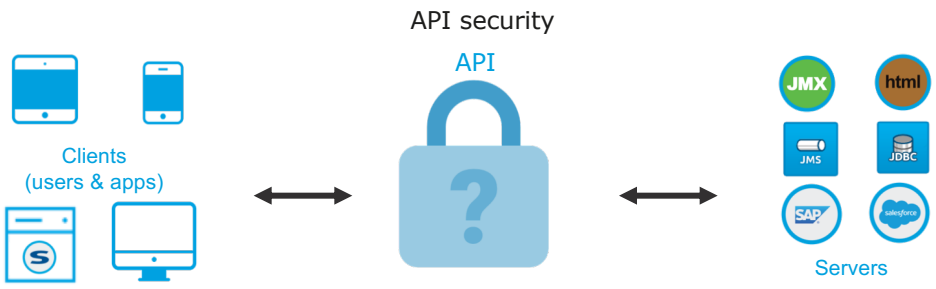
MuleSoft

Module 11: Securing APIs

Goal

MuleSoft



At the end of this module, you should be able to

- Define API security requirements
- Use security schemes to apply resource and method level policies
- Define custom security scheme for APIs
- Apply an OAuth2.0 external provider policy to resource methods

Introducing API security requirements

Requirements for API security

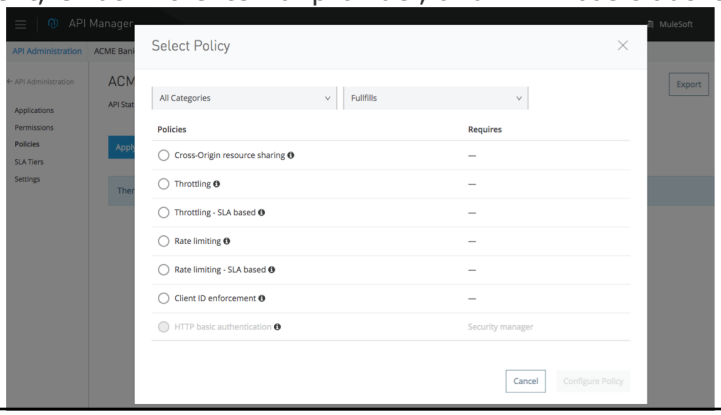
MuleSoft

- Identity
  - Core of security for an API
  - Helps recognize apps and users that consume the API, the servers that the API makes calls to
  - API should identify itself to both app and servers
- Confidentiality
  - Information processed by the API is made visible only to users, apps, and servers that are authorized to consume it
- Integrity
  - The message received by the API is verified as being the one sent by the app
  - The same applies for when the API acts as client to a server
- Availability
  - An API must never lose information, it should handle requests and process them in a reliable fashion

Anypoint Platform security capabilities

MuleSoft

- API Manager in Anypoint Platform offers a suite of security policies
  - A policy is a mechanism for enforcing security (and other filters) on traffic
  - Some policies are inherently dependent on a mechanism to verify incoming tokens
  - Some of the policies that address these security requirements are Client ID enforcement, OAuth2.0 external provider, and HTTP basic authentication



Defining security schemes in RAML 1.0

MuleSoft

- RAML supports a range of built-in security scheme types
  - The schemes must conform to a specified standard of declaration to support the policies
- The security scheme node is a map that contains key-value pairs
  - type: OAuth1.0/ OAuth2.0/Basic Authentication/ Digest Authentication …
  - displayName
  - description
  - describedBy: A map of key-value pairs including
    - headers
    - queryParameters
    - Responses
  - settings: authorizationUri, accesstokenUri, authorzationGrants and scopes

Specifying the security scheme to be used by a resource

MuleSoft

- Use the securedBy node in the RAML root or inside a specific resource
  - At the root, it applies to all the methods in the API
  - In a resource, it applies to only that resource
    - Overrides any security scheme applied in the root

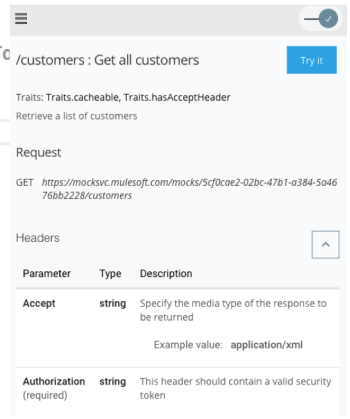
```
31  /customers:
32    type:
33      collection:
34        customErrorDataType: CustomErrorMessage
35    post:
36    get:
37      description: Retrieve a list of customers
38      displayName: Get all customers
39      securedBy: oauth2_0
40      is:
41        - Traits.cacheable
42        - Traits.hasAcceptHeader:
43            customErrorDataType : CustomErrorMess
```

Walkthrough 11-1: Define a custom security scheme for an API

MuleSoft

- Create a custom security scheme file
- Reference the custom security scheme in the main RAML API definition
- Apply the security scheme to certain resource methods

```
24  securitySchemes:
25      customTokenSecurity: !include securitySchemes/customTo
26
27  securedBy: customTokenSecurity
28
29  /customers:
30    type:
31      collection:
32        customErrorDataType: CustomErrorMessage
```



Introducing OAuth2.0 external provider policy

Introducing the OAuth2.0 token enforcement policy

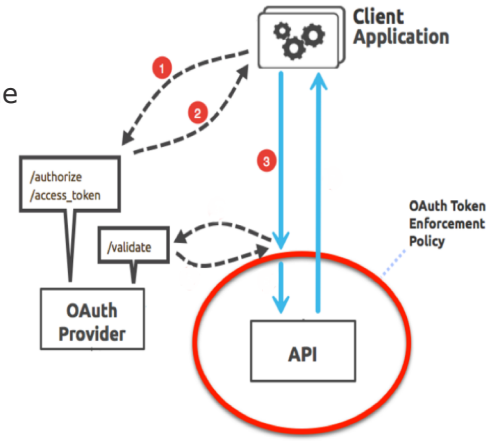
MuleSoft

- Enforces interaction between an external OAuth2.0 provider, the API, and the client application
- To apply this policy, an external OAuth2.0 provider application is required
- Anypoint Exchange contains a sample OAuth2.0 provider
  - Must be configured with the client credentials of the business group in which the API is registered
  - The client application must request access to the API to gain client credentials that should be included while sending a request to an API endpoint
- The API RAML definition should contain a securitySchemes node with the OAuth2.0 provider endpoint in the settings field

Controlling access to resources using an OAuth provider

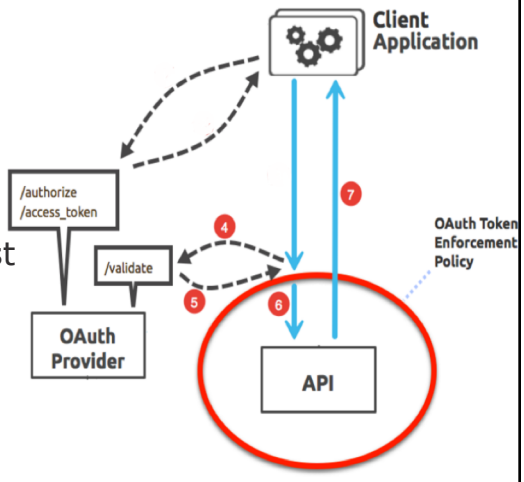
MuleSoft

1. The client app sends a request with the client ID in the authorization header to access data from the protected resource
   - Client app is redirected to a page supplied by the OAuth provider, that asks the API to use credentials to authenticate the client to receive an authorization grant from the OAuth provider
2. OAuth provider returns an access token
3. Client app sends a request to the API, appending the access token to the request URL



Controlling access to resources using an OAuth provider (cont)

4. The OAuth2.0 token enforcement policy intercepts the request and validates the token using the validate endpoint in the OAuth provider
5. If the access token is valid, the OAuth provider authenticates the client app
6. The OAuth provider forwards the request to the API
7. The API returns the response to the client app



The OAuth provider validates client apps using credentials

- Access token
  - Security credential that identifies a particular application
  - Authenticates a request from the application to an API resource
- An authorization grant is a method for a client application to acquire an access token
  - Types of grants include implicit, authorization code, client credentials, password

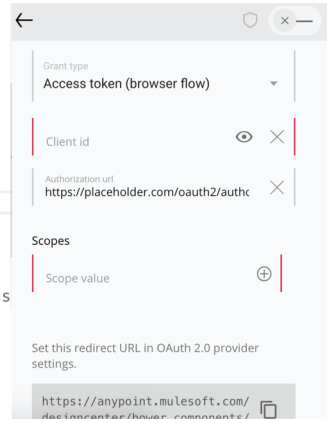
OAuth provider endpoints

- Authorization endpoint: /authorize
  - Used by the client to interact with the API to obtain authorization grant
  - The OAuth2.0 provider verifies the identity of the API using the username and password login
- Access token endpoint: /access_token
  - Used by the client application to exchange an authorization grant for an access token
- Validation endpoint: /validate
  - Used by the OAuth2.0 provider to validate the access token sent by the client in the request to the API resource
  - Validates the client's access token against a keystore

Walkthrough 11-2: Consume an OAuth2.0 security scheme for an API and secure API resources

- Consume an OAuth2.0 security scheme fragment file
- Reference the OAuth2.0 security scheme in the RAML API definition
- Apply the security scheme in the API resource methods

```
31  /customers:
32    type:
33      collection:
34        customErrorDataType: CustomErrorMessage
35    post:
36    get:
37      description: Retrieve a list of customers
38      displayName: Get all customers
39      securedBy: oauth2_0
40      is:
41        - Traits.cacheable
42        - Traits.hasAcceptHeader:
43            customErrorDataType : CustomErrorMess
```



Summary

Summary

MuleSoft

- Anypoint Platform enables API creators to build integrity, confidentiality, and reliability into the APIs in the design stage
- Security policies allow for APIs to be designed with security as a part of the initial design, and not an afterthought
- RAML helps define security policies using security scheme fragments