Apache Airavata Security Manager

Authentication & Authorization Implementation for a Multi-Tenant e-Science Framework

Supun Nakandala, Hasini Gunasinghe, Suresh Marru and Marlon Pierce

Science Gateways Research Center

Indiana University

Outline

- Introduction
- Problem Definition
- Solution Overview
- Solution in Detail
- Implementation Details
- Conclusions

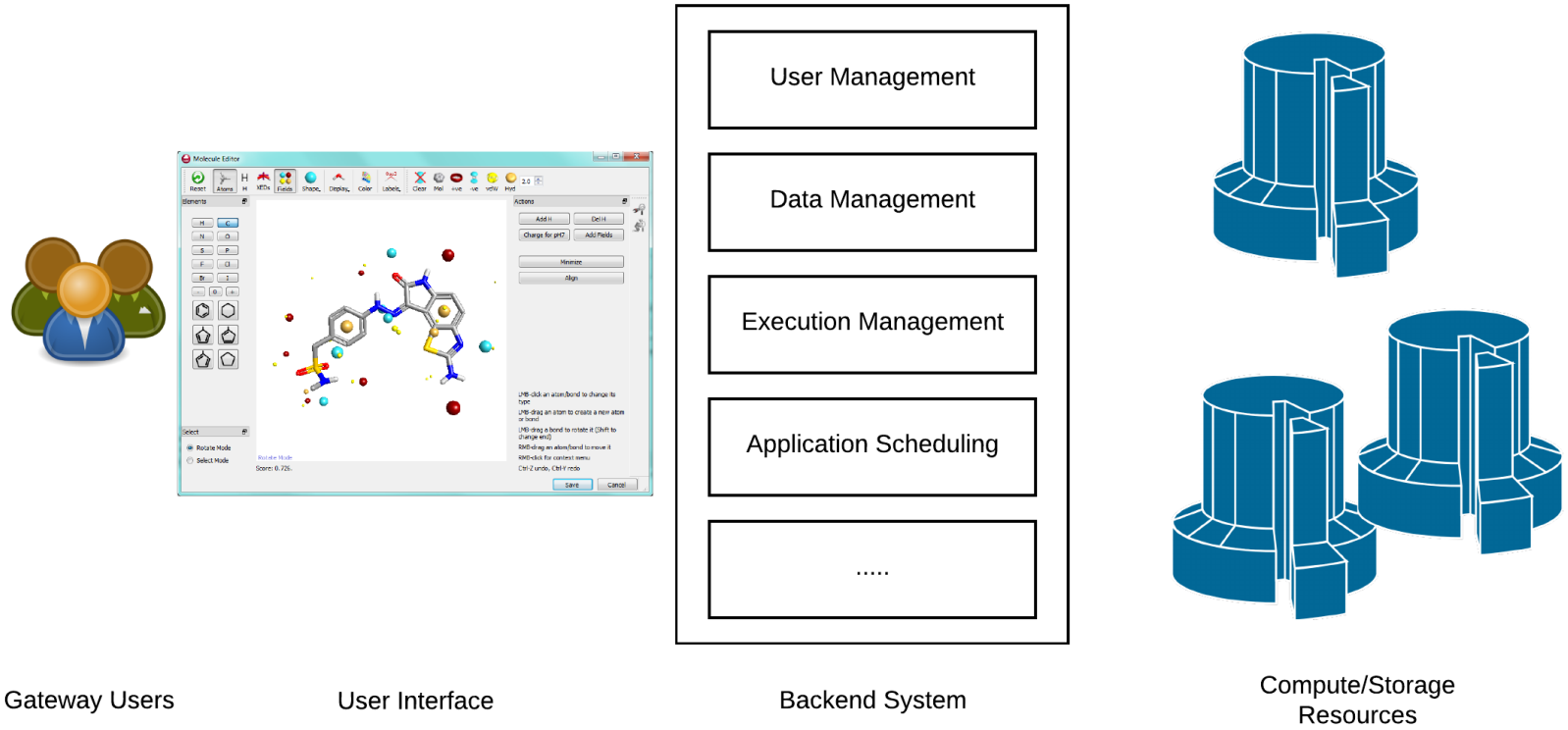
Introduction

- Solving **identity management** challenges in **multi-tenanted** eScience middleware that needs to support multiple **diverse virtual organizations**.
- Implementation is based on Apache Airavata Science Gateways middleware. But the concepts are equally applicable to other similar systems too.

Introduction – What are Science Gateways?

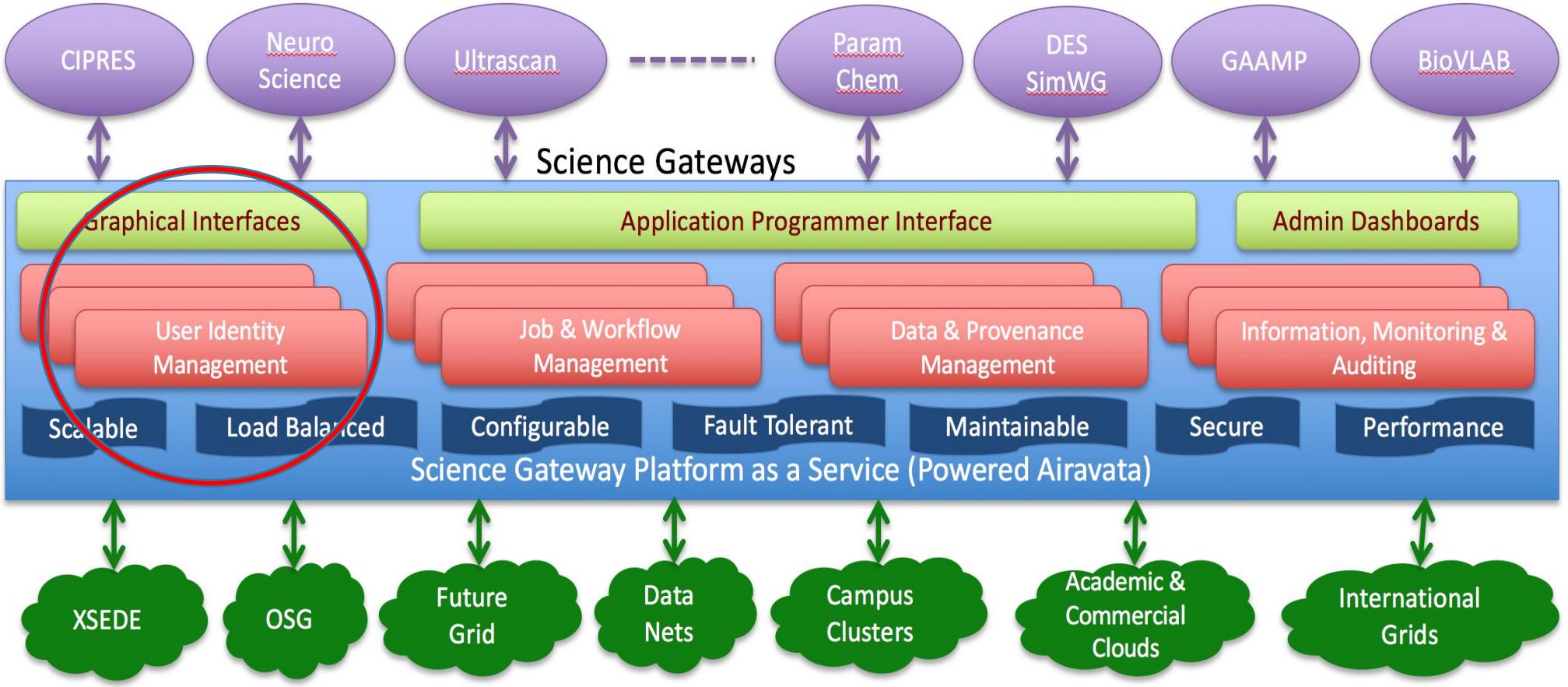
- A Science Gateway is a **community-developed** set of tools, applications, and data that are integrated via a portal or a suite of applications, usually in a graphical user interface, that is further **customized to meet the needs of a specific community**. (*XSEDE)*

What are these customized requirements ?



Gateways share many of these requirements!

Apache Airavata



Outline

- Introduction
- Problem Definition
- Solution Overview
- Solution in Detail
- Implementation Details
- Conclusions

Problem Definition

- Previously
  - Security model was based on the trust relationship between gateway software and Airavata middleware by restricting access to the Airavata API only from pre-validated web-based gateway clients.
  - Mutual trust between the gateways and Airavata server was established using TLS mutual authentication and enforcing firewall commands.
  - End users who interacted with Airavata API through gateways were only authenticated and authorized at the gateway level, and no validation was done at the Airavata API level. Hence no explicit user notion in Airavata.

Problem Definition (Continued…)

- Previous approach was reviewed by the Center for Trustworthy Scientific Cyberinfrastructure and was determined to be operationally acceptable [1].
- However,
  - This approach does not scale to a large number of gateways.
  - It does not address the issue of securing native client (desktop and mobile) access to the Airavata API.
  - It does not enable a uniform approach to user-level tracking of API calls.
  - Not satisfying from the architectural point of view.

[1] - R. Heiland, J. Basney, and V. Welch, “Suggested security practices for SciGaP: A preliminary report,” http://hdl.handle.net/2022/20811.

So why can't we implement identity management in Airavata ?

- Three different identity management scenarios that needs to be considered
  - Scenario 1 - The gateway client does not have a user store and would like to depend on Airavata to provide user management features.
  - Scenario 2 - The gateway has a user store and in-house identity management mechanisms.
  - Scenario 3 - The gateway does not have a dedicated user store but authenticates users into the gateway using some federated identity provider.

Formal Problem Definition

- How to provide a **unified identity management solution** that can meet the standard security requirements for the above **three usecases** and be able to **seamlessly adopted** by all types of gateways including web based and native (desktop and mobile) clients

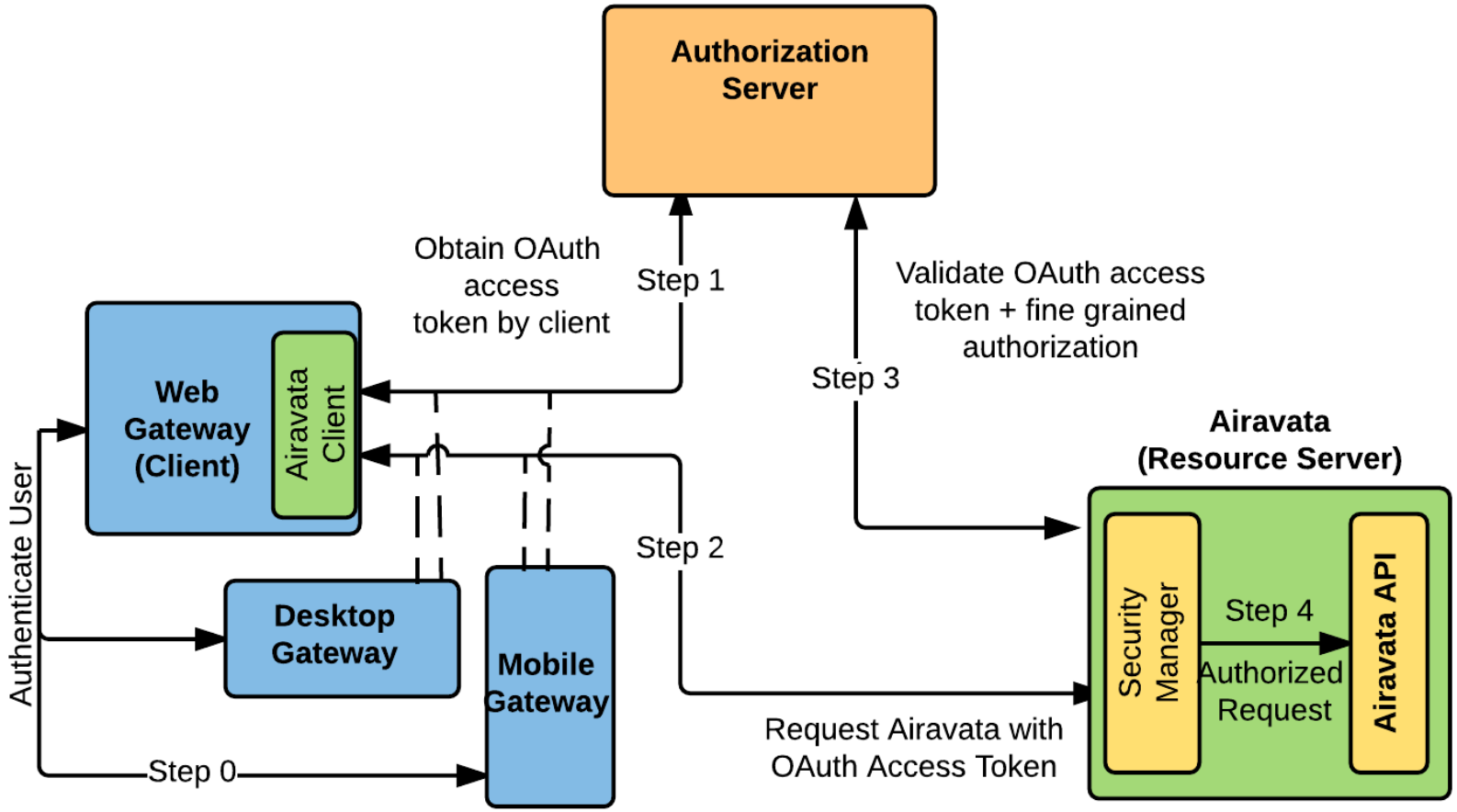
Outline

- Introduction
- Problem Definition
- Solution Overview
- Solution in Detail
- Implementation Details
- Conclusions

Solution Overview

- We use standard security protocols and standards in our design.
- OAuth 2.0 based **authorization delegation** for the **user authenticated at the gateway**.
- OAuth access tokens are generated by a **separate** dedicated **authorization server**.
- We map specific OAuth grant types for our requirements.
- OpenID-Connect which runs on top of OAuth 2.0 for user authentication.
- Role based fine grained customized authorization is done using XACML.

High level Solution



High level Solution

- Only the user authentication and access token retrieval will change for each use case scenario.
- Depending on the client type and usage scenario appropriate OAuth grant type should be used to obtain an access token.

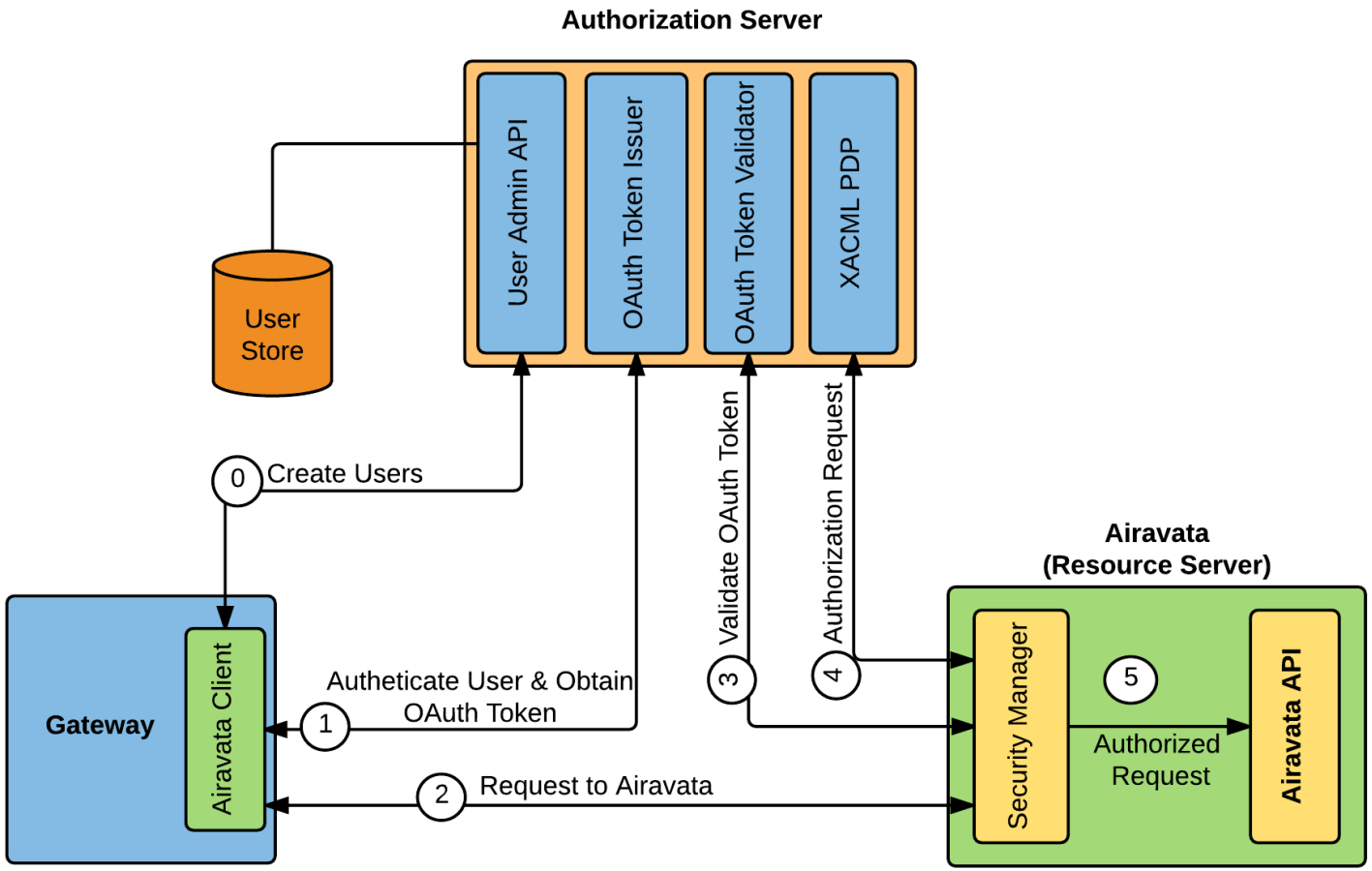
Outline

- Introduction
- Problem Definition
- Solution Overview
- Solution in Detail
- Implementation Details
- Conclusions

OAuth 2.0 Grant Types

- Authorization code grant – Client app is web based (or can spawn a web browser) (e.g. Web applications) and can maintain a client credential.
- Implicit grant – Client app is web based (or can spawn a web browser) but cannot keep it's credentials secret (e.g. Thick web clients)
- Resource owner password grant – User trusts the client application (e.g. Gateway provided desktop clients)
- Client credential grant – Machine to machine communication. No user involvement.
- Refresh code grant – Retrieve new access token when current token expired

Scenario 1 – Gateway does not have existing user management



Scenario 1 – Gateway does not have existing user management

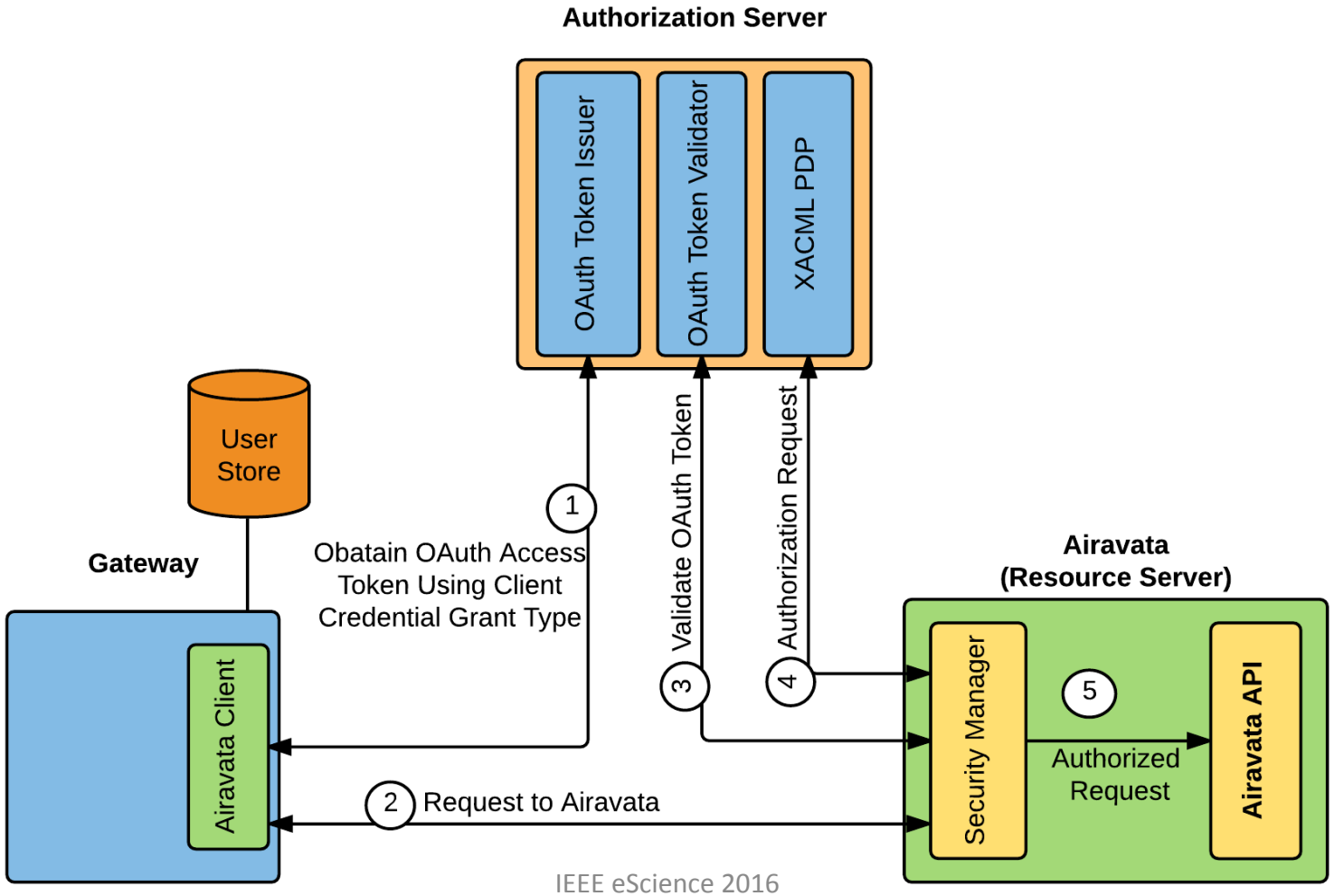
- Authorization Server maintains a user store for the gateway.
- Gateway will use Airavata SDK to invoke user management operations.
- Authorization Code, Implicit or Resource Owner Password can be used to obtain an access token.

Scenario 2 – Gateway has a user-store and its own in-house identity management mechanisms

- Case 1 – Gateway does not share any user information with Airavata.
- Case 2 – Gateway is willing to share user identity information but does not allow Airavata to connect to the gateway's user store.
- Case 3 – Gateway is willing to share user identity information.

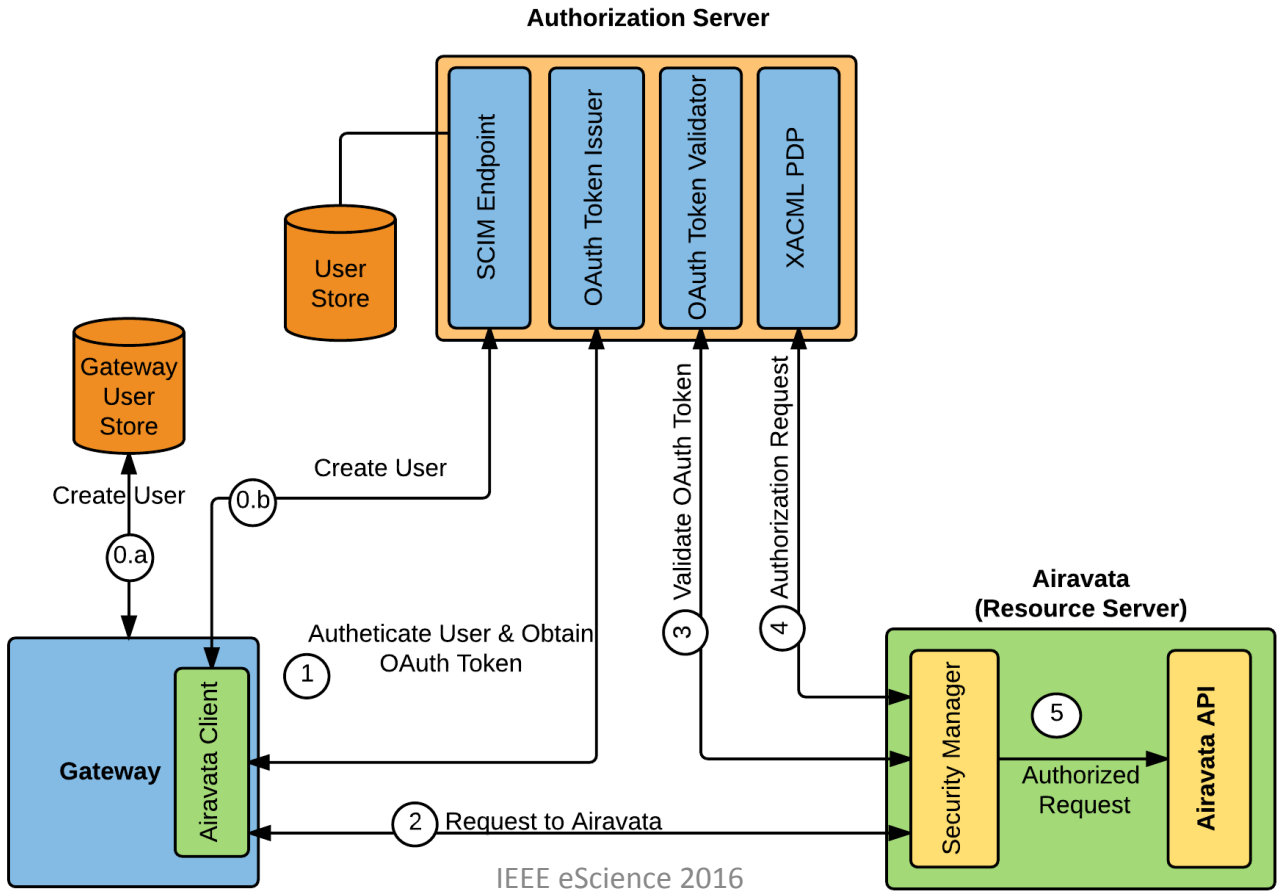
Scenario 2 – Gateway has a user-store and its own in-house identity management mechanisms

- Case 1 – Gateway does not share any user information with Airavata



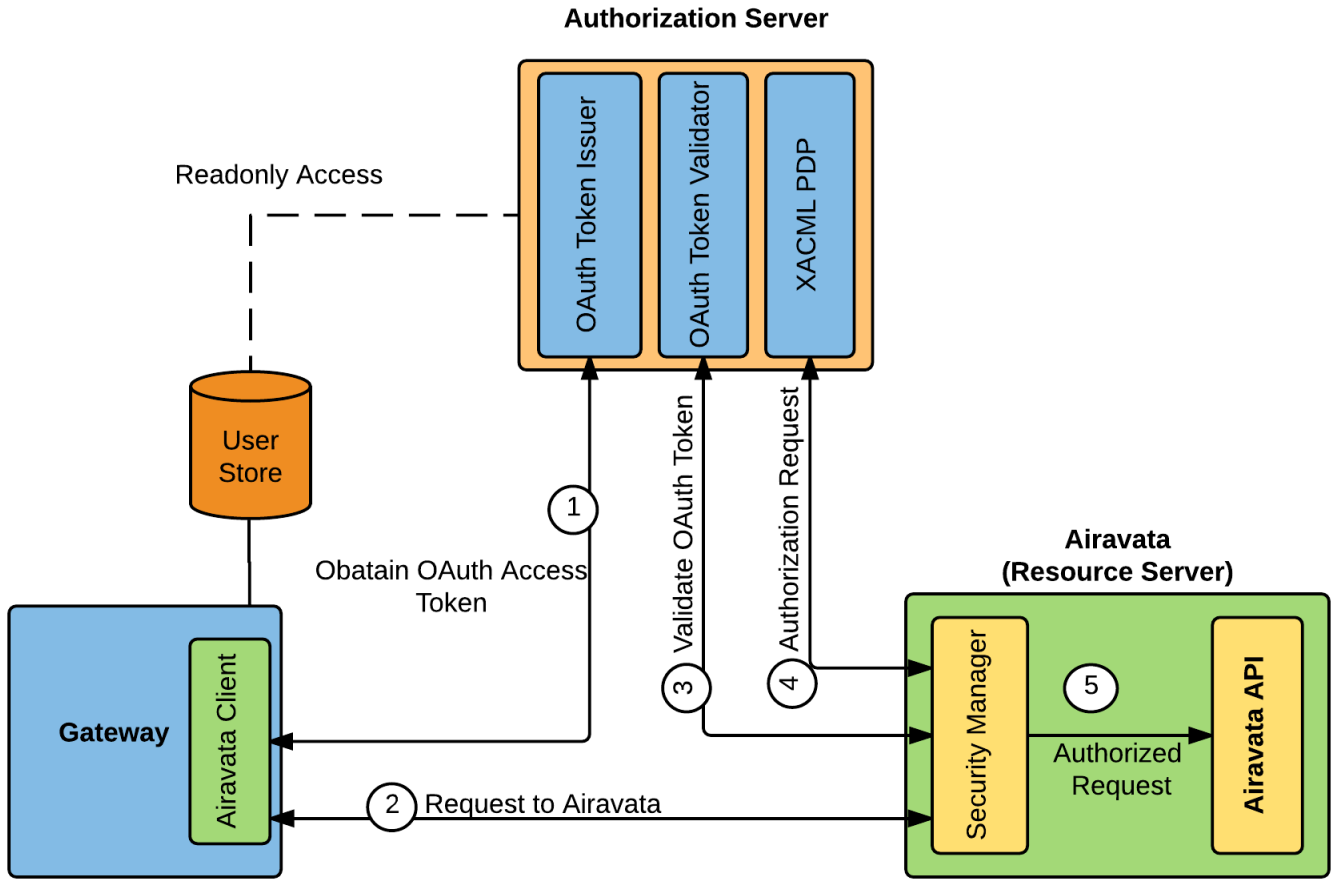
Scenario 2 – Gateway has a user-store and its own in-house identity management mechanisms

- Case 2 – Gateway is willing to share user identity information but does not allow Airavata to connect to the gateway's user store

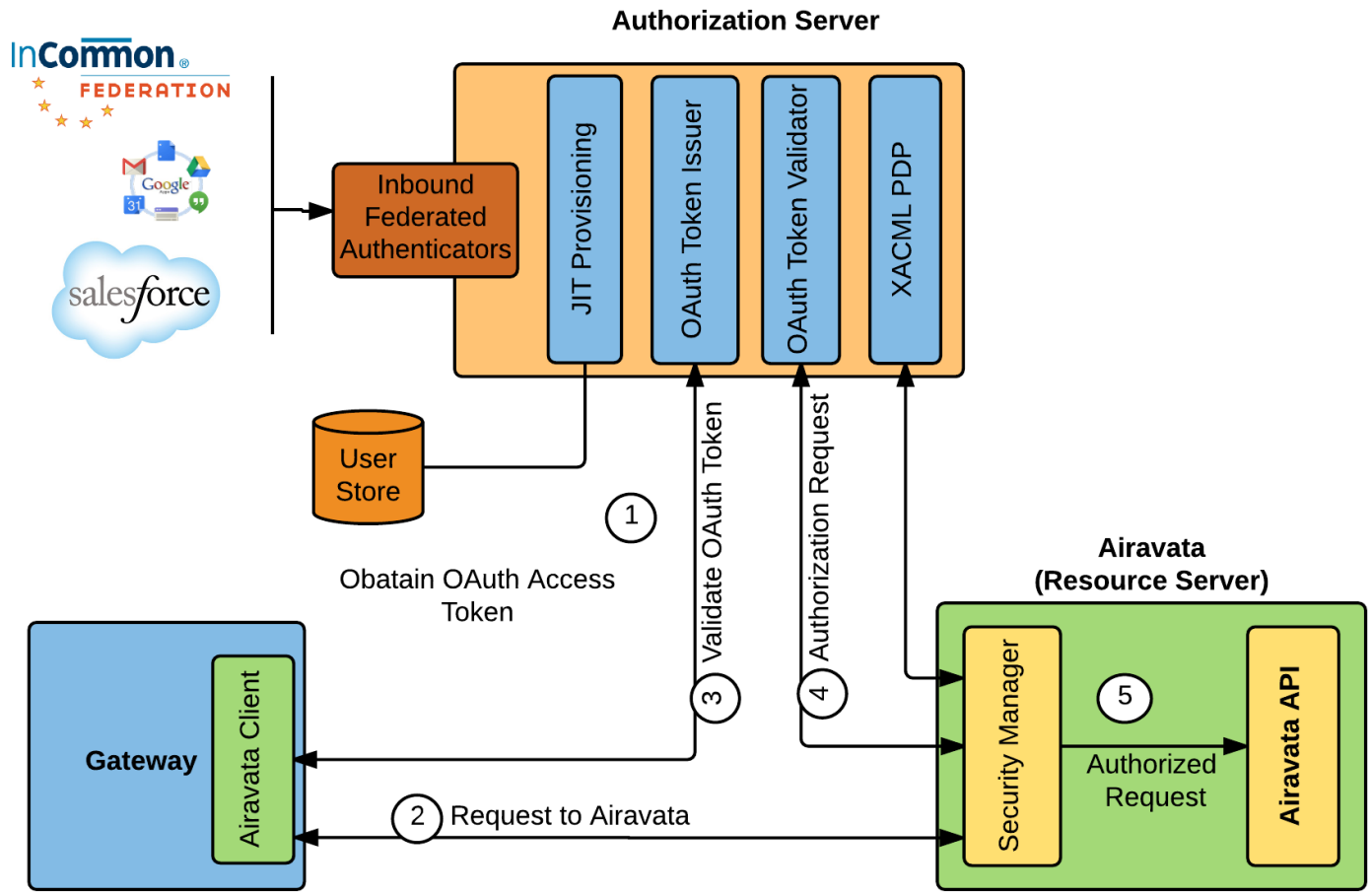


Scenario 2 – Gateway has a user-store and its own in-house identity management mechanisms

- Case 3 – Gateway is willing to share user identity information.



Scenario 3 – Gateway authenticates users into the gateway using a federated identity provider.



Scenario 3 – Gateway authenticates users into the gateway using a federated identity provider.

- Authentication request is forwarded to configured federated identity provider.
- If the federated identity provider supports retrieval of user information a user account is created just-in-time.

Role based fine grained user authorization

- XACML is used to define customized role based API authorization decisions.
- Each gateway can have different policy on how they allow their users to access the API.

Role based fine grained user authorization

```
<Rule Effect="Permit" RuleId="GATEWAY-USER-PERMIT">
    <Target>
        <AnyOf>
                <Match>
                    <AttributeValue>^(?:(?!
                            /airavata/getAPIVersion|
                            /airavata/createProject|
                            /airavata/updateProject|
                            /airavata/getProject|
                            /airavata/createExperiment|
                            /airavata/updateExperiment|
                            /airavata/getExperiment|
                            /airavata/launchExperiment|
                            .....
                            .....
                            .....
                            ).)*$\r?\n?
                    </AttributeValue>
                </Match>
        </AnyOf>
    </Target>
    <Condition>
        <Apply>
            <AttributeValue>"GATEWAY-USER"</AttributeValue>
            <AttributeDesignator AttributeId="http://wso2.org/claims/role"
                    Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
        </Apply>
    </Condition>
</Rule>
```

Outline

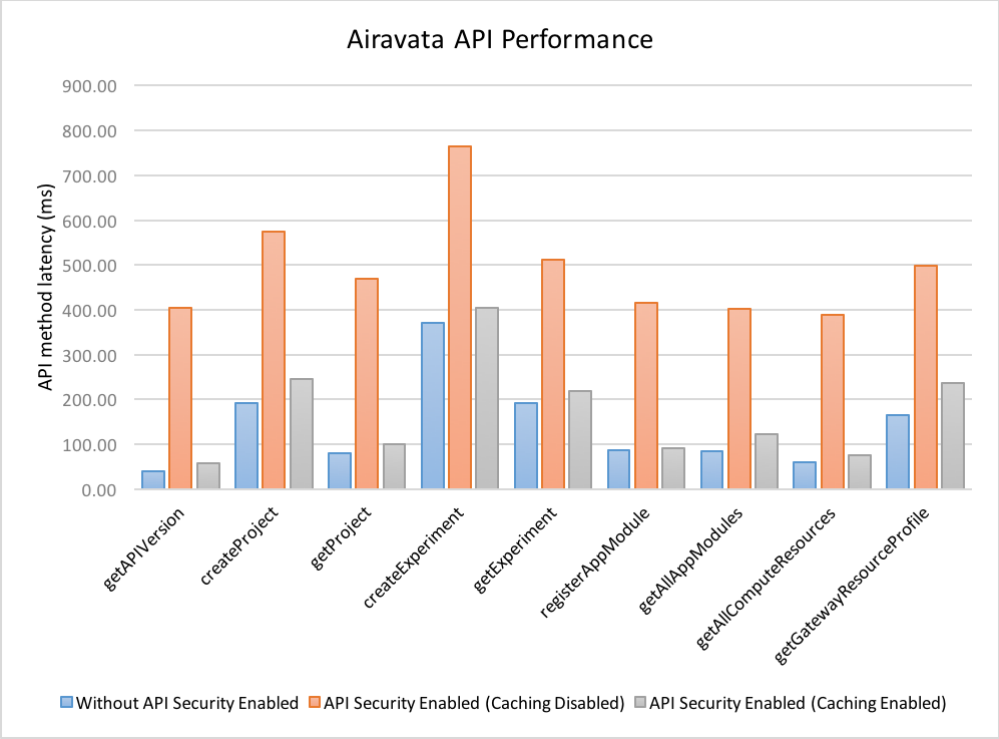
- Introduction
- Problem Definition
- Solution Overview
- Solution in Detail
- Implementation Details
- Conclusions

Implementation Details

- Most of the features that we use in our solution are based on standard security protocols.
- We use WSO2 Identity Server which is an open source (Apache V2 license) identity management system which supports multi-tenancy out of box and provide most of the required features.
- We extend the features available in IS in order to support custom user store and federated authenticator integration.
- A new component, Security Manager, is added to the Airavata API which manages the communication with Authorization server and validates user requests.

Implementation Details

- Additional security validation added some overhead on the overall Airavata API performance. But caching of authorization decisions improved it a lot.



Outline

- Introduction
- Problem Definition
- Solution Overview
- Solution in Detail
- Implementation Details
- Conclusions

Conclusions

- The most significant advance in gateway architectures over the last several years is the use of hosted, general purpose gateway platform services.
- We examined the over the-wire access patterns that exist between a wide range of gateway clients and multi-tenanted platform services like Apache Airavata.
- We map these patterns to widely accepted security standards and protocols and implement a solution that can support all the identified use cases.

Q & A
Thank You!