Securing Integration of Cloud Services in Cross-Domain Distributed Environments

Bojan Suzic

Graz University of Technology, Austria

SUNFISH This work has been supported partially by the SUNFISH project (N.644666) funded by the European Commission H2020 Program

Overview

- Cloud Integration – Motivation and Approaches
- Goal and Methodology
- Integration Framework
- Authorization Protocols
- OAuth 2.0 vs UMA Flows and Controls
- Security Assessment
- Summary and Further Work

Cloud Integration – Motivation

- Broad adoption of cloud paradigm ⇒ organizational data assets are increasingly getting transferred to and consumed from the cloud
- Characteristics of cloud services:
  - run by external organizations
  - based on different layers and XaaS models
  - geographically dispersed premises
  - subjected to various jurisdictions
  - functionalities or resources partially exposed to the clients (APIs)
- Traditionally, integration assumes point-to-point or point-to-multipoint integration from organizational premises
- What happens when this process starts from the cloud?

Cloud Integration – Approaches

- Integration Platform as a Service *(IPaaS)* - a new approach that transfers the complete business process execution to the cloud
- IPaaS defined by Gartner as:

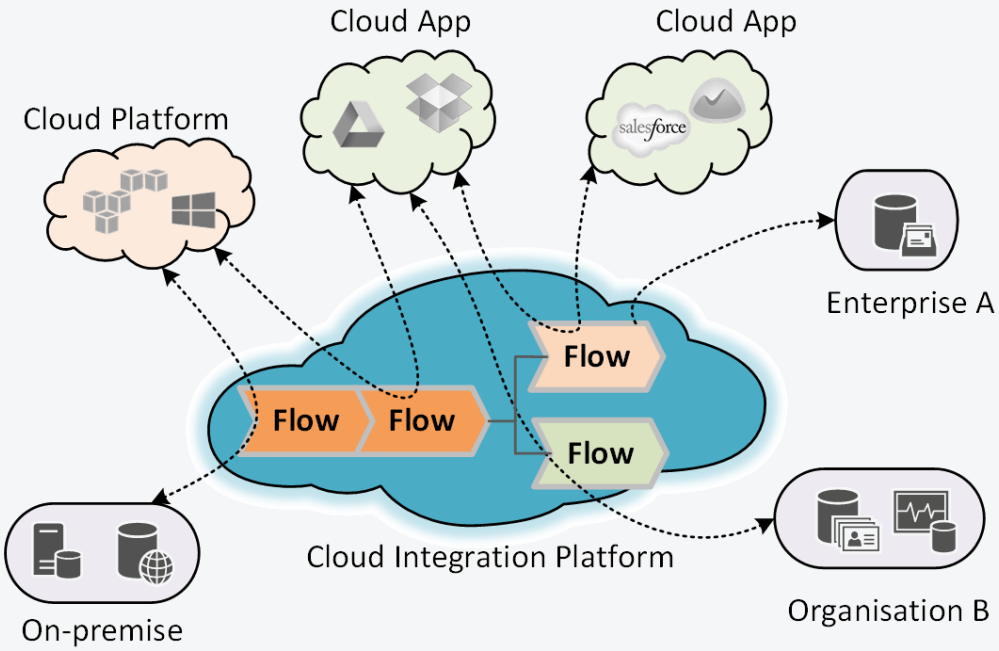
> *… a suite of cloud services enabling development, execution and governance of integration flows connecting any combination of on-premises and cloud-based processes, services, applications and data within individual, or across multiple organizations* [1)]

- Integration of (cloud) services:
  - Builds on, reuses and extends traditional approaches
  - Enterprise Application Integration (EAI)
  - Enterprise Service Bus (ESB)
  - Enterprise Integration Patterns[2)] (EIP)
  - Service Oriented Architecture (SOA)

[1)] M. Pezzini and B. Lheureux. *Integration platform as a service: moving integration to the cloud*. Gartner, 2011.
[2)] G. Hohpe and B. Woolf. *Enterprise Integration Patterns*. 9th Conference on Pattern Language of Programs, 2002.

Cloud Integration – Approaches

- Typical integration scenarios:
  - Cloud to on-premises
  - Cloud to cloud
  - On-premises to on-premises
- Resources are stored, transferred and processed entirely in the cloud, which becomes the central point of integration



Goal and Methodology

- Goal - assessing the security of service integrations in the cloud
- Scope:
  - Processes present in integration platforms, spanning multiple organizations
  - Web-API (RESTful) approaches relying on web authorization protocols
- Web authorization protocols:
  - OAuth 2.0: broadly adopted web authorization framework
  - UMA: OAuth profile introduced by Kantara Initiative
    Recently stabilized (2015), missing support among integration frameworks
- Methodology:
  - Evaluate protocol capabilities and flows with the help of test environment
  - Assess security using RMIAS[3)] framework based on seven categories

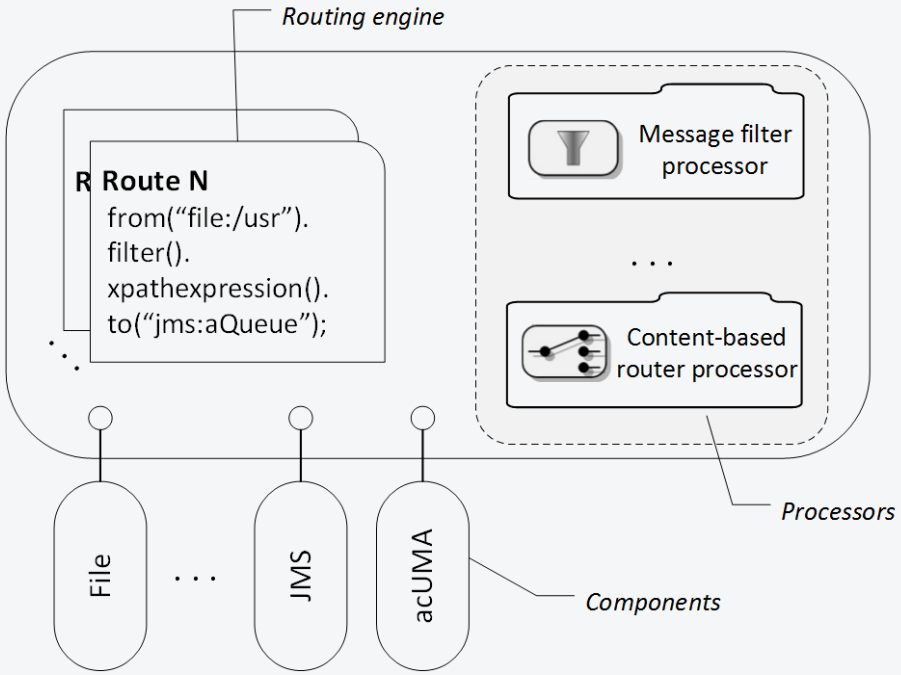
[3)] Y. Cherdantseva and J. Hilton. *A Reference Model of Information Assurance & Security.* (2013)

Integration Framework

- Apache Camel:
  - Java-based framework, Apache 2 license
  - Deployed as standalone application or integrated into other frameworks
  - Used in other products such as JBoss FUSE, OpenESB, Talend suite
- acUMA:
  - A component that enables consumption of UMA-protected endpoints
  - Manages communication with resource server, authorization server, supports additional claim-based flows
- Prototype and test environment used to examine cross-domain integration flows of resource sharing based on OAuth and UMA

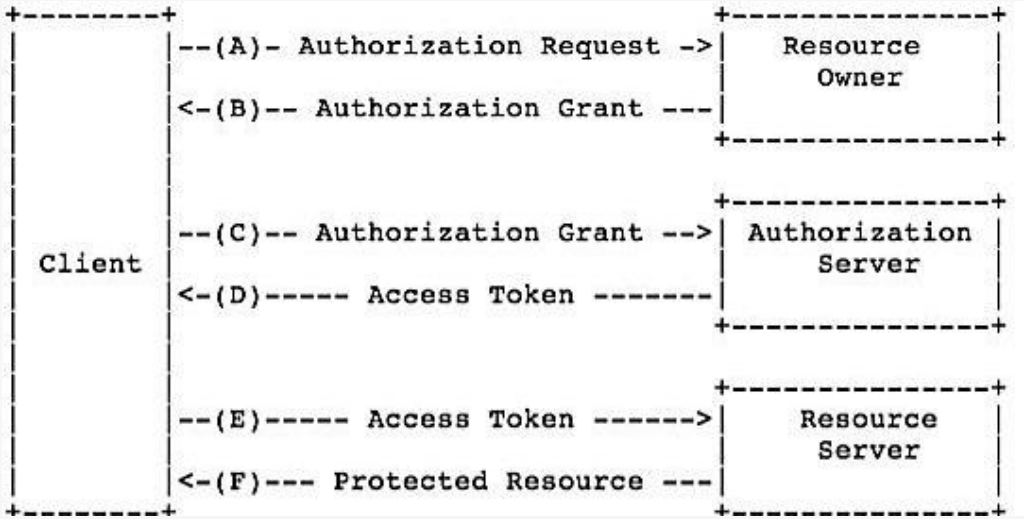
Integration Framework

- Apache Camel:
  - routing and mediation engine: interprets DSL-based specifications, executes EIPs and manages the traversal and processing of messages
  - processors: perform processing and implement EIPs
  - components: establish the connectivity with other systems, expose producers and consumers for interaction



Authorization Protocols

- OAuth 2.0:
  - Authorization framework
  - Standardized in 2012, builds on OAuth concepts (2010)
  - Aims to enable clients to access server resources on behalf of a resource owner based on its consent
  - Entities: *resource owner*, *client*, *resource server*, *authorization server*

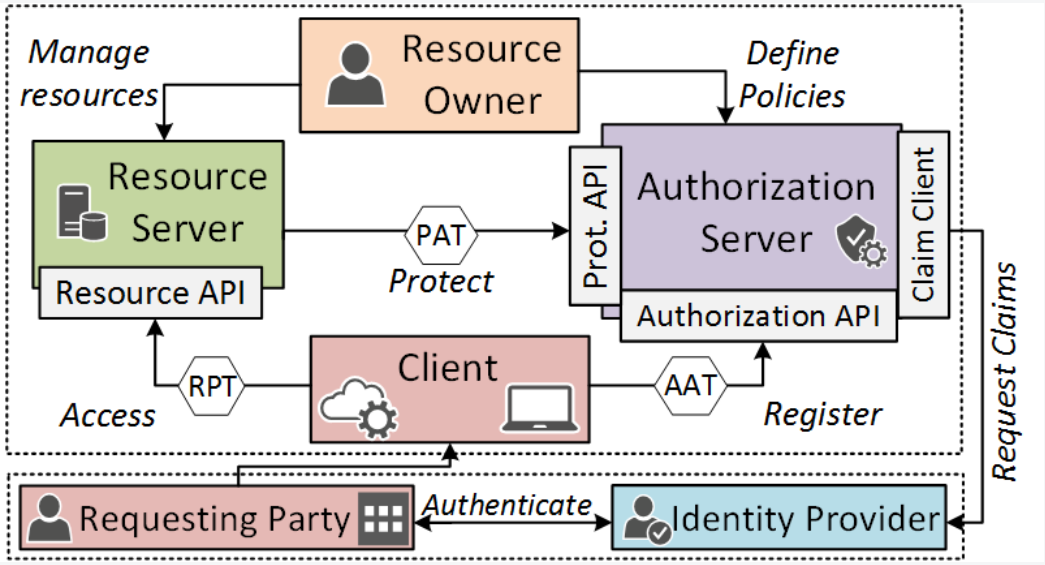


4)

4) RFC 6749

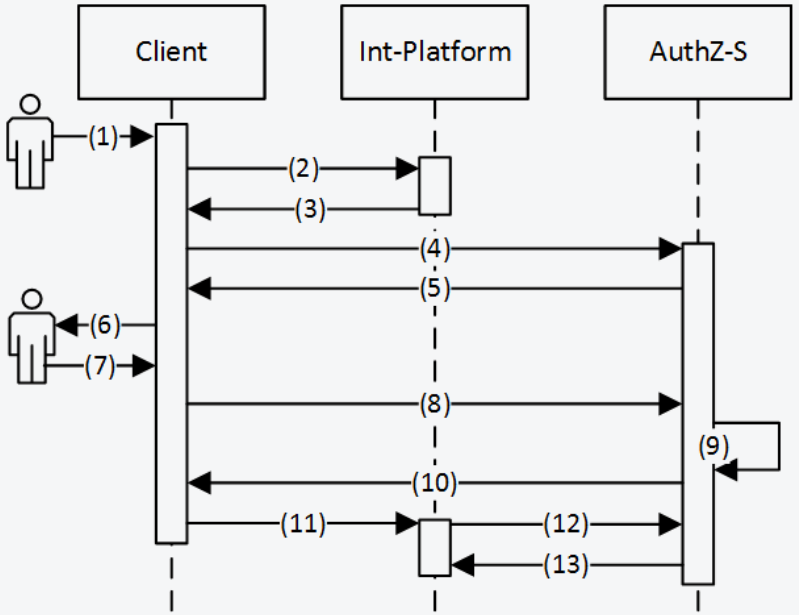
Authorization Protocols

- UMA (User-managed access):
  - Extends OAuth 2.0, stable UMA-Core (2015)
  - Defines flows, processes and APIs that govern access to protected resources in distributed environment
  - Introduces *resource-owner policies, centralized authorization server, claims-gathering flow* and *requesting party*
  - Specifies separate access token types for access to resource server (RPT) and authorization server (AAT and PAT)

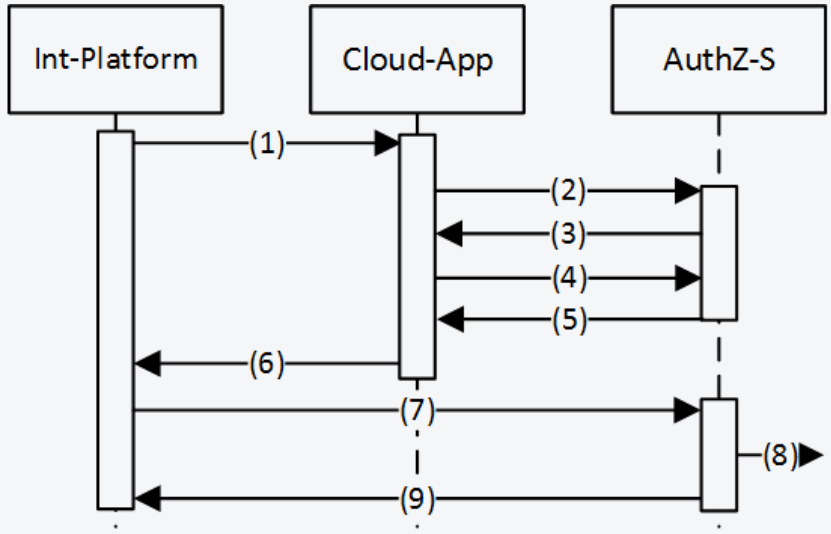


OAuth 2.0 vs UMA Flows

- Authorizing integration platform
  Establishes relationship and provides authorization for future interactions



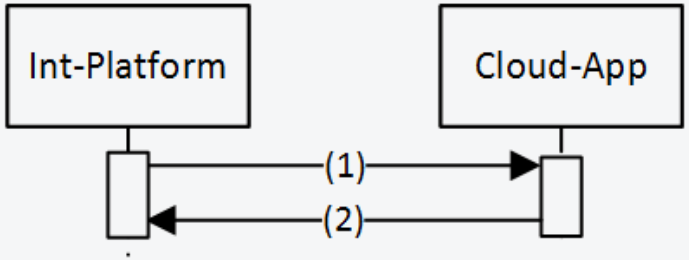
OAuth 2.0



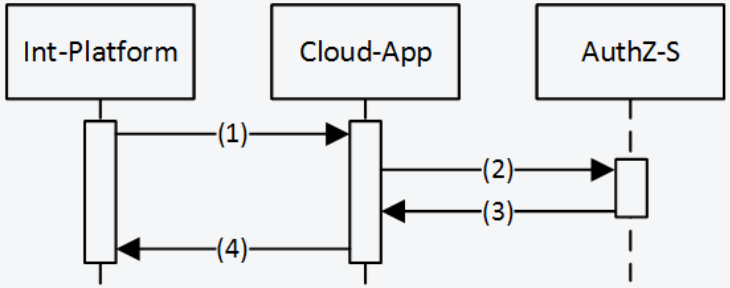
UMA

OAuth 2.0 vs UMA Flows

- Consuming cloud resources
  Repetitive interactions



OAuth 2.0



UMA

OAuth 2.0 vs UMA Controls

Authorization of peers

- OAuth 2.0: *synchronous* flow tight to access request ⇒ *explicit* authorization
- UMA: separate flow ⇒ *asynchronous* and *implicit* authorization

Architectural blocks

- OAuth 2.0: methods and flows for access request evaluation not specified
  - Increases coupling between AS and RS
  - Restricts AS and RS to the same organization
- UMA: enables distributed access control
  - Introduces protection API and resource set registration
  - Structured evaluation of access requests
  - AS and RS decoupled, enabling deployment in separate environments
  - One AS can control resources distributed in different domains

OAuth 2.0 vs UMA Controls

Authorization scope and granularity

- OAuth 2.0 relies on a *scope* to allow access to a resource
  - Out-of-band defined abstract construct incorporating several dimensions
  - Implicitly states allowed resources (or depends on context)
  - References resources as categories, not as instances
  - Example scope: *drive.file, gmail.labels, gcalendar.readonly*

- UMA introduces *resource set* and *permissions*
  - Scope represents an action (extent) over an object (resource set)
  - Enables more structured representation and evaluation of access scopes
  - Scope granularity relates to the level of abstraction of a resource set
  - Granularity possible on arbitrary level, down to a particular instance of a resource
  - Example permission: *["view","http://drive.example.com/dir1"]*

OAuth 2.0 vs UMA Controls

Evaluation of access requests

- OAuth 2.0 – static, does not foresee any form of policies
  - *Explicit*, *consent-based* access delegation without further evaluation
- UMA uses *authorization policies*
  - Definition and enforcement of policies separated (resembles XACML)
  - Owners define policies at *authorization server*
  - *Resource server* checks requests against *authorization server*
  - *AS* evaluates requests against policies, deciding dynamically
  - Policies can involve other variables, enabling *context-based* authorization
  - Flow and architecture are defined, but not the structure, semantics or evaluation process of policies
  - Policies are not portable; coupled with particular organization and implementation

OAuth 2.0 vs UMA Controls

Error handling and claims gathering

- OAuth 2.0 requires presence and involvement of resource owner for consents
- UMA does not require the presence of resource owner
  - Access decision can be performed based on available policies
  - The consent of resource owner can be gathered in out-of-band process asynchronously
- Due to opaque property of standard access tokens, in both cases clients are not able to determine the reach of the token validity
- *Claims-gathering* flow in UMA enables to request additional information from clients
  - Can depend on the type of access or target resource, or context
  - Additional authentication token may be requested, using e.g. OpenID Connect
  - Has potential to entail other capabilities, such as *obligations*

Security Assessment

Confidentiality

- Only authorized users can access information
- Refined with the *principle of least privilege* :
  provide a minimal range of privileges to accomplish a task
- OAuth 2.0 uses coarse-grained scopes, providing more information than necessary
- UMA enables more granular definition of permissions
  - Possibility to reference a range of resources or a particular resource instance
  - Two dimensions: extent (scope) and a resource

Example scopes (8) and actions (34) for GDrive:

- drive, drive.file, drive.readonly, drive.appdata, drive.metadata.readonly, drive.metadata, drive.photos.readonly, drive.scripts
- Files.copy, Files.delete, Files.create, Files.get, Files.update, Comments.get, Comments.list, Replies.update, Changes.watch, About.get, ...

Example permissions:

- ["view","http://drive.example.com/dir1"]
  ["update","http://drive.example.com/dir2"]

Security Assessment

Integrity

- Assurance of data integrity and absence of unauthorized modifications
- In this context: ability of a client to alter the data on a resource server
- OAuth 2.0: ability to modify resources depends on access scope

> Google Drive:
> drive.file: *Comments.create, Comments.update, Files.create, Files.delete, Permissions.create, Permissions.delete, Replies.create, Revisions.update …*
>
> Microsoft Calendar:
> calendars.readwrite: *create event, decline event, create attachment, delete attachment, create calendar, update calendar, create calendar group …*

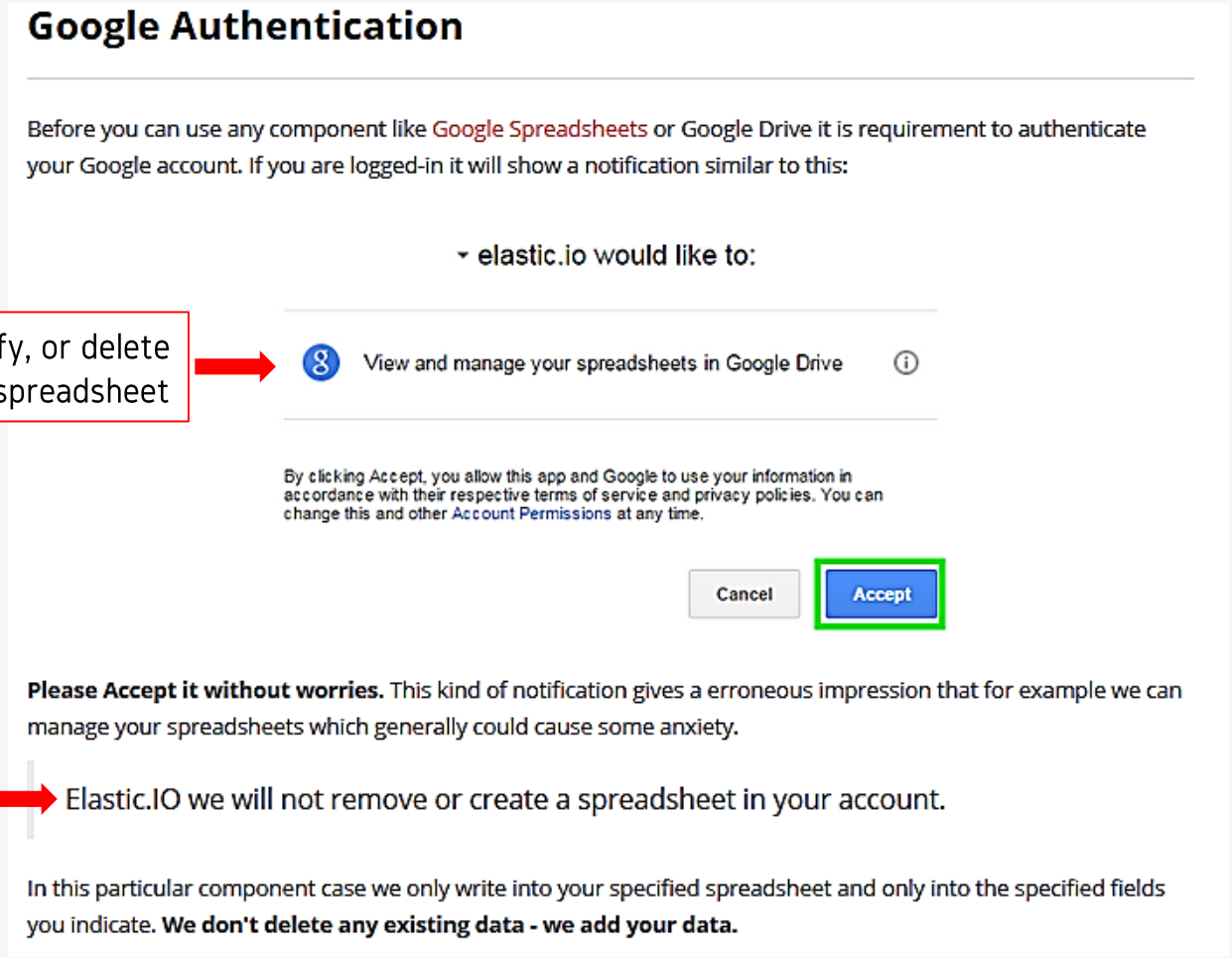
- UMA: more restrictive, as type of actions can be separated from resource type or abstract category of actions

> ["update","http://drive.example.com/dir/file1.png"]
> ["delete","http://drive.example.com/dir/file2.png"]

Security Assessment

Coarse-grained OAuth 2.0 access scopes in practice (requesting authorization):

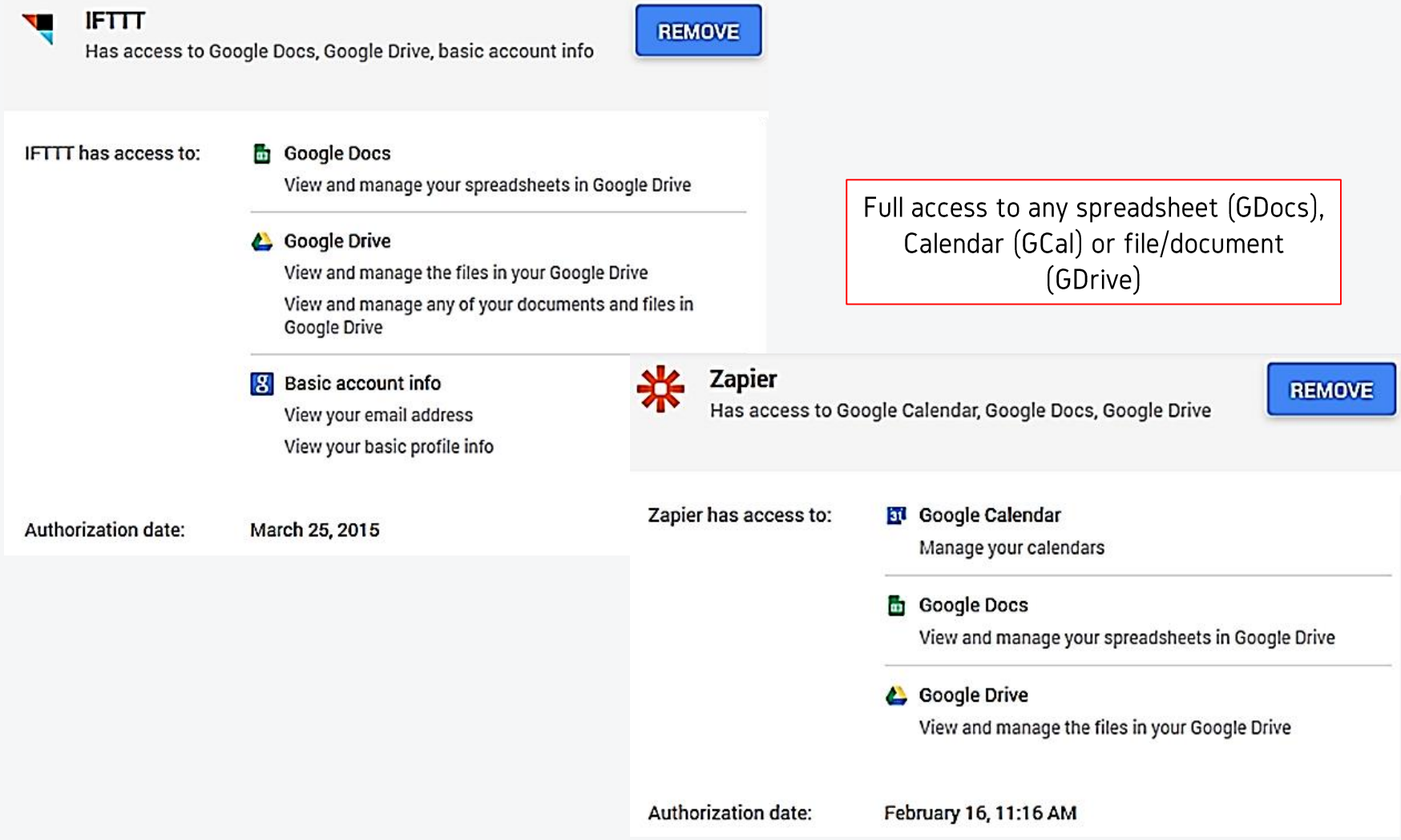
Read, modify, or delete any spreadsheet



A declarative statement, as security can not be fully enforced

Security Assessment

Coarse-grained OAuth 2.0 access scopes in practice (managing authorizations):



Security Assessment

Accountability / Auditability

- Capability of the system to hold users responsible for their actions
- Ensuring persistent and reliable monitoring of all actions performed within the system
- Monitoring of the use of information so that misuse can be determined and misbehaving parties held accountable
- Not considered in both protocols

Non-repudiation

- Ability of a system to prove (non-) occurrence of an event, as well as the (non-) participation of a party in a transaction
- Not a primary concern in both protocols
- *Claims-gathering flow* in UMA introduces the means to request additional information or processes to be performed
- Clients may involve requesting party to satisfy additional authorization requirements and initiate processes such as strong/step-up authentication

Security Assessment

Authenticity and Trustworthiness

- Verify the identity and establish trust in a third-party involved in a transaction
- *Implicit authorization grant* of OAuth 2.0: verification of identity of resource owner only; high-assurance techniques not part of the protocol
- UMA includes verification of *requesting party* and *resource server*
- Servers authenticated on a basis of TLS

Privacy

- Obeying privacy legislation and enabling individuals to control, where feasible, their personal information
- UMA framework approaches privacy by providing hints and recommendations to implementations and operators
- Distributed architecture and authorization policies enable resource owners to control sharing of their data using own infrastructure or trusted party
- Constrained degree of control, privacy still challenging area

Summary and Future Work

Summary

- Cloud-based integration adds complexity by executing processes and connecting resources hosted at various organizations
- Management of security and privacy spans across multiple entities and platforms
- Heterogeneity of systems, jurisdictions, APIs, metadata, authorizations, formats
- Both OAuth 2.0 and UMA focus on confidentiality
- Permissions or scopes are tied to the systems, defined at design-time by service provider, clients and resource-owners need to comply

Further work

- Structuring cross-organizational security management, considering interoperability and additional security dimensions and measures
- Applying semantic technologies to enable portability and understanding of data and management controls; exposing out-of-band specifications
- Open API: recent initiative by Cloud Security Alliance

Any questions?

Thanks for your attention!