

User-centered Security Management of API-based Data Integration Workflows

Bojan Suzic

Graz University of Technology, Austria



Overview

- Cloud Integration – Introduction and Approaches
- API-based Integration: Use Case, Protocols, Practice
- Proposed Approach
- Integration in the Workflow
- Discussion
- Summary and Further Work

Cloud Integration – Introduction

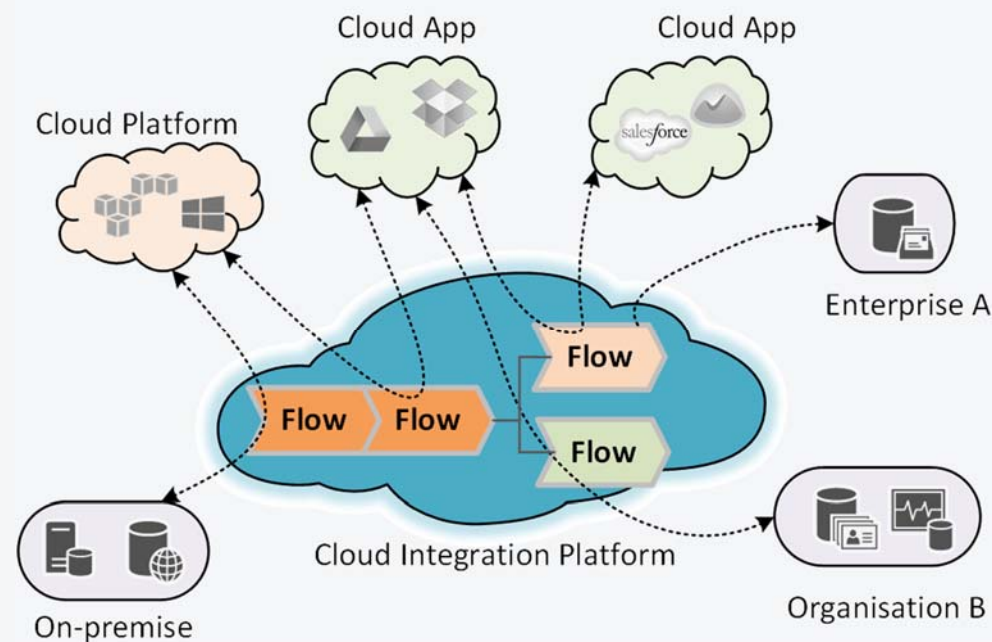
- Broad adoption of cloud paradigm \Rightarrow organizational data assets are increasingly getting transferred to and consumed from the cloud
- Traditionally, integration assumes point-to-point or point-to-multipoint connections originating from organizational premises
- What happens when this process starts from the cloud?
- 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 organizations¹⁾

¹⁾ M. Pezzini and B. Lheureux. *Integration platform as a service: moving integration to the cloud*. Gartner, 2011.

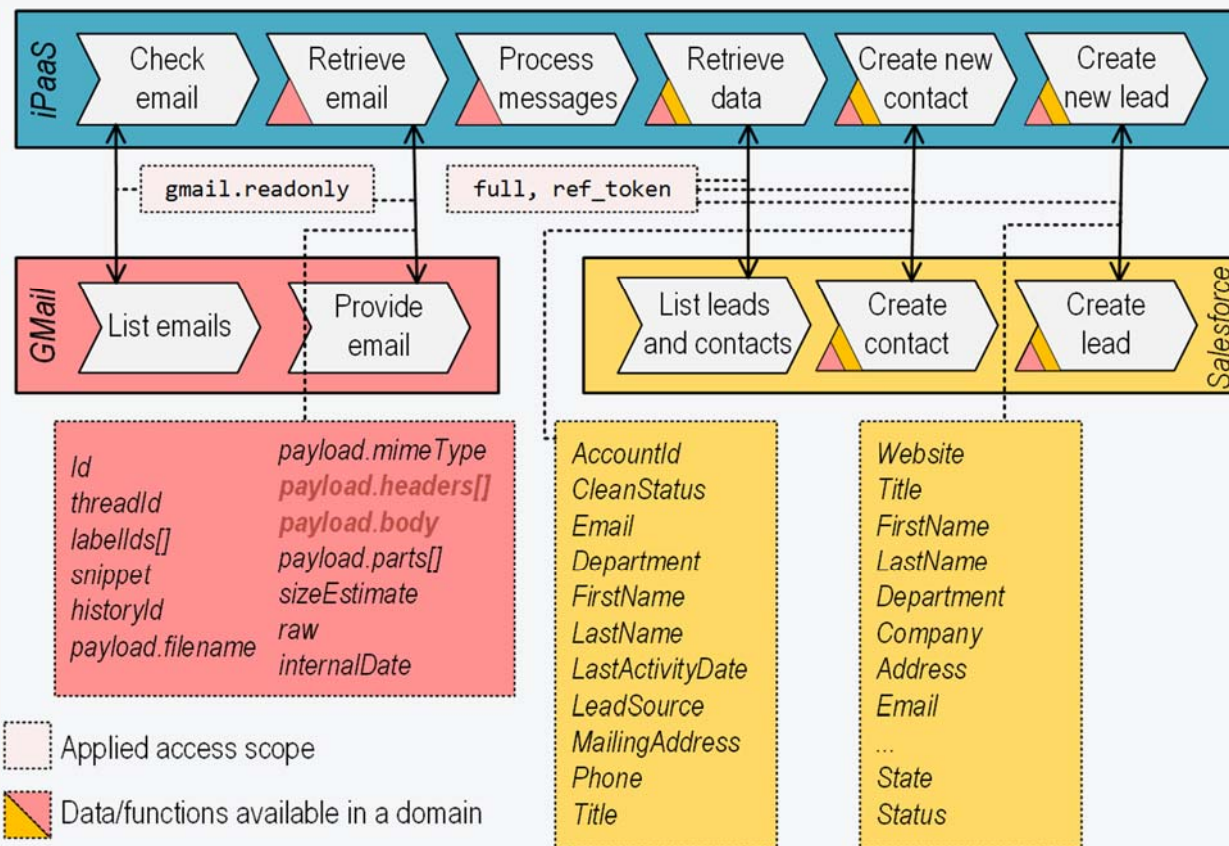
Cloud Integration – Approaches

- 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
- Like standard data sharing, but: increased complexity, many actors



API-based Integration – Use Case

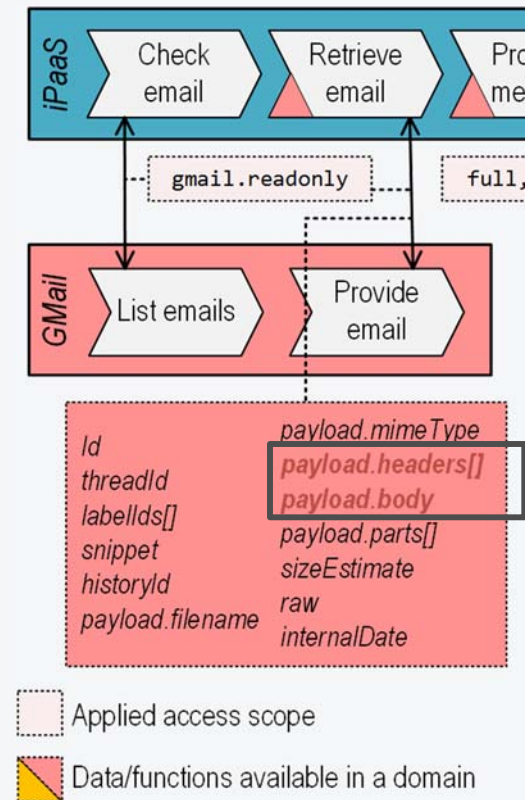
- Organization owns the accounts at three clouds: iPaaS, Gmail, Salesforce
 - iPaaS (platform): periodically checks, retrieves and processes email, creating the leads and contacts at Salesforce
 - Based on REST-API interactions, secured using OAuth 2.0 and *access scopes*



API-based Integration – Use Case

- *Authorization scope*: abstract term, non-structured, out-of-the-band, opaque
- Gmail: supports a limited set of static scopes
- Able to read all emails and metadata in an account
- There is no specialized view, data filtering, limitation on fields...

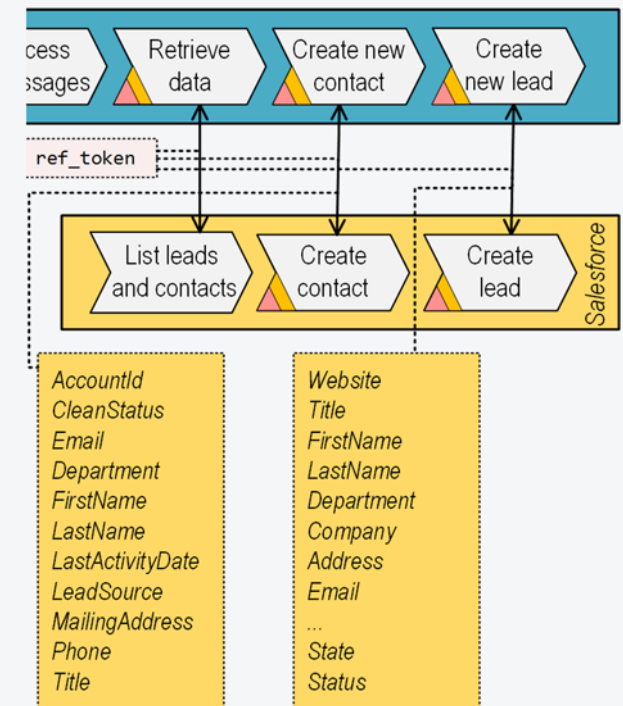
Scope	Meaning
gmail.readonly	<i>Read all resources and their meta-data</i>
gmail.compose	<i>Create, read, update and delete drafts, including sending of messages and drafts</i>
gmail.send	<i>Sending of messages, without read or update privileges</i>
gmail.insert	<i>Inserting and importing messages</i>
gmail.labels	<i>CRUD of labels</i>
gmail.modify	<i>All R/W operations except perm. deletion of threads and messages</i>
https://mail.google.com/	<i>Full access to the account</i>



API-based Integration – Use Case

- *Authorization scope*: abstract term, non-structured, out-of-the-band, opaque
- Salesforce: in many cases full scope required
- Able to perform a broad set of operations well beyond the required use case
- Coarse-grained scopes - *Least privilege principle?*

Scope	Meaning
api	Access to user's account using API. Encompasses chatter_api scope
chatter_api	Access to Chatter API resources
custom_permissions	Access to the custom permissions in a client-associated organization
full ←	Full data available to the user. Encompasses all other scopes
id	Access to the identity URL service
openid	Unique identifier of user in OpenID Connected applications. Can be used to retrieve token conforming to OIDC specifications
refresh_token	Allows provision of refresh tokens to eligible users, enabling the client to interact with the resource in offline mode.
visualforce	Provides access to Visualforce
web	Allows to use the token on the web, incl. access to Visualforce pages



API-based Integration in Practice

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

The screenshot displays two OAuth 2.0 authorization cards. The top card is for IFTTT, showing access to Google Docs, Google Drive, and basic account info. The bottom card is for Zapier, showing access to Google Calendar, Google Docs, and Google Drive. A red box highlights the Google Docs and Google Drive permissions for IFTTT, with text explaining that these permissions grant full access to any spreadsheet, calendar, or file/document.

IFTTT
Has access to Google Docs, Google Drive, basic account info **REMOVE**

IFTTT has access to:

- Google Docs**
View and manage your spreadsheets in Google Drive
- Google Drive**
View and manage the files in your Google Drive
View and manage any of your documents and files in Google Drive
- Basic account info**
View your email address
View your basic profile info

Authorization date: March 25, 2015

Zapier
Has access to Google Calendar, Google Docs, Google Drive **REMOVE**

Zapier has access to:

- Google Calendar**
Manage your calendars
- Google Docs**
View and manage your spreadsheets in Google Drive
- Google Drive**
View and manage the files in your Google Drive

Authorization date: February 16, 11:16 AM

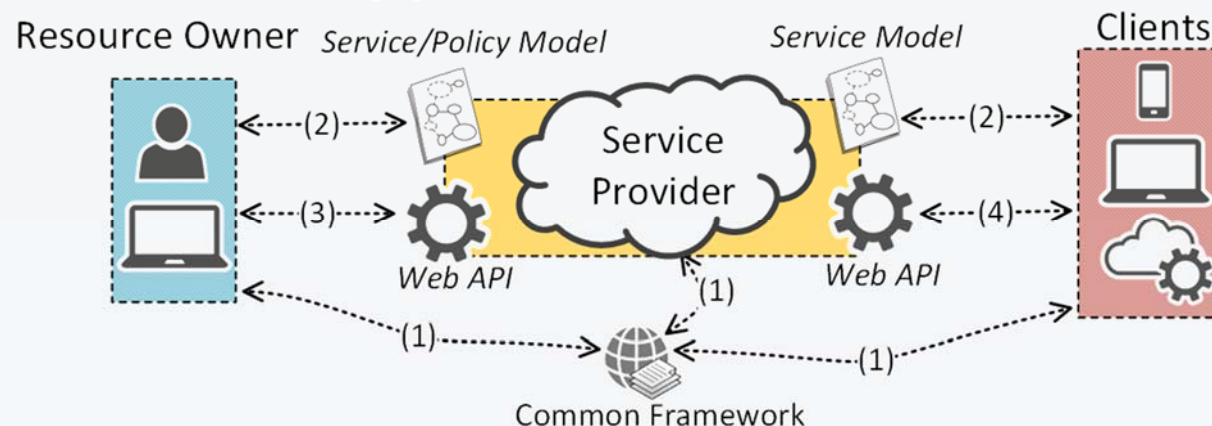
Full access to any spreadsheet (GDocs), Calendar (GCal) or file/document (GDrive)

Issues and Challenges

- Issues
 - Unilateral definition of security concepts, APIs, management interfaces
 - Manual and proprietary management of security functions
 - Tools or approaches cannot be reused (many-to-many)
- Management of security requirements from the perspective of resource owner?
 - *Interoperable*: apply the same approach among different providers/platforms
 - *Automated*: support autonomous agents
 - *Dynamic*: allow dynamic definition of owners' policies and clients' requests
 - *Granular constraints*: restrict access to resources or their parts relevant for an activity – principle of least privilege
 - *Contextual views*: enforcement and resource transformations based on a context
- Goals
 - Enabling cooperative, dynamic and granular access management
 - Considering complex API-based interactions in cross-domain environments

Proposed Approach

- Actors:
 - Service provider (SP): exposes the service, such as XaaS
 - Resource owner (RO): consumes services, outsources data and processing
 - Clients (C): consume the resources provided by SP and owned by RO
- Steps:
 - Establish relationship with the common framework {SP, RO, C} [1]
 - Expose and consume model of services – resources and capabilities {SP, RO, C} [2]
 - Expose model of security policies {SP, C} [2]
 - Update and manage security policies {RO, SP} [3]
 - Request resources {C, SP} [4]



Exposing Services and Capabilities

- Characterizing APIs exposed by service provider
 - Web-APIs – there is a broad range of RESTful approaches and models
 - Exposed APIs specific to provider's use-cases, environment, data models, processes, with different maturity levels
 - Development and maintenance overhead: clients need to be developed and maintained separately for each service provider through the whole lifecycle
 - Hard-wiring: out-of-the band contracts, underlying semantic unknown
- Service description
 - Lightweight model based on *semantic interoperability layer*
 - Modeling the resources and operations *from security perspective*
 - Relies on an external or common model to represent the exposed services, resources and operations that can be managed
 - Reusing existing blocks

Exposing Services and Capabilities

- Core and domain-specific vocabularies
 - *Core vocabulary* provides the general concepts
 - Can be reused to expose general concepts
 - *Domain-specific* vocabularies extend the core, optional
 - Refine domain-related concepts
- Categories and application of concepts:
 - Entity classes - represented in object hierarchies
 - Object properties - establishing relations between the entities
- Next: excerpts of classes and properties

Entity Classes (Excerpt)

- Service:
 - The primary point in service provider's offer to clients
 - Class members denote different types of services that correspond to a particular use case
 - Examples: *CloudService*, *EmailService*, *StorageService*
- Resource:
 - Class of entities exposed by a service, manageable and consumable
 - Represents entities at different levels of abstractions, allowing specification of the granularity of resource sharing and application of (pre)processing operations.
 - Examples: *Object*, *Drive*, *Directory*, *File*, *Document*, *Media*, *Email*, *MsgBody*, *MsgHeader*
- Operation:
 - Operations are actions that can be executed on resources prior to their sharing
 - Effectively provide a resource view adjusted to a particular context
 - Examples: *RemovePII*, *MaskPII*, *Encrypt*, *Mask*

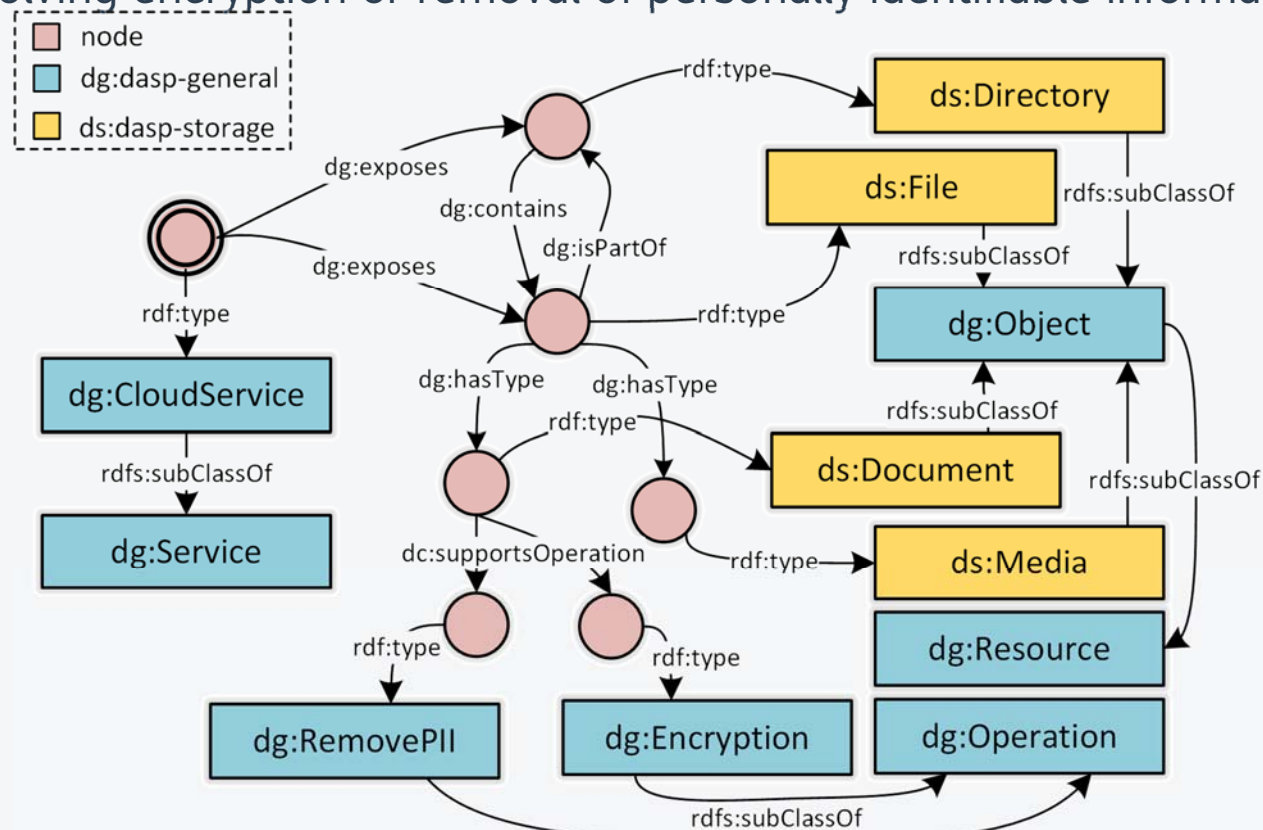
Object Properties (Excerpt)

- Connect entity instances of different concepts, establishing relations
- Behavioral and constitutional
- Examples:

<i>exposes</i>	Entity that service exposes to clients using web API
<i>contains, isPartOf</i>	Relationships between entities stating constitutive (hierarchical) relationship between instances
<i>subClassOf</i>	An abstract description of possible inter-class relations
<i>type</i>	Class type of particular entity instance
<i>hasType</i>	Supported class specialization for exposed resources
<i>requestsAccess</i>	Used to relate an access requests with a service
<i>supportsOperation</i>	Denotes operations that can be executed on entity
<i>acceptsOperation</i>	Denotes operations that are accepted by the client
<i>acceptsSubtype</i>	Subtype of resources satisfying a client request

Exposing Services and Capabilities

- Example model using *core* and *storage* vocabularies
 - *Nodes* are instances of concepts; labeling does not affect the workflow
 - Starts with *CloudService*, which exposes *File* and *Directory*
 - *File* may be further specialized (property) as a *Document* or *Media*
 - Prior to the delivery to the client, a *Document* can be subjected to operations involving encryption or removal of personally identifiable information



Security Policies Classes (Excerpt)

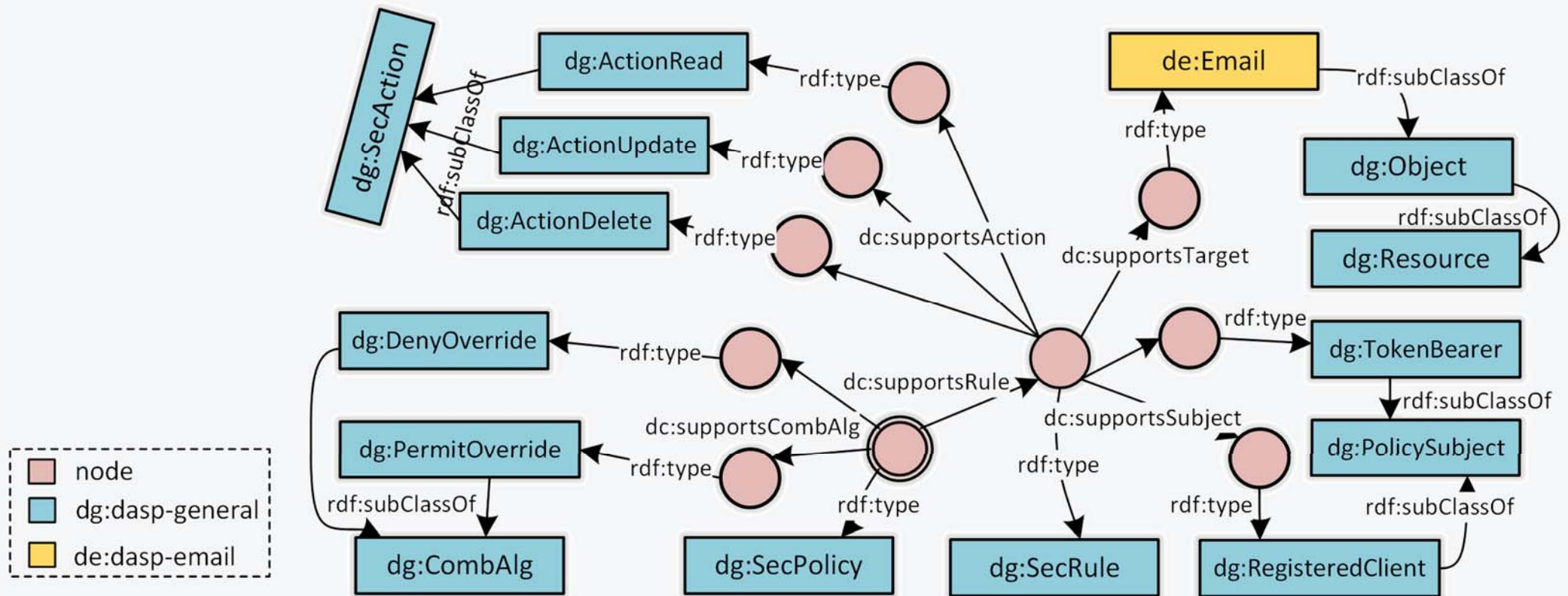
- SecPolicy:
 - The primary point in the definition of security policies.
 - One policy contains one or more rules with associated combination algorithm
 - Similar to the concept of XACML
- SecRule:
 - Defines a node used to connect policy subject, target, action, condition and obligation
- PolicySubject:
 - States a range of subjects or their attributes
 - Integration of different access control models and mechanisms such as OAuth 2.0
 - Examples: *RegisteredClient*, *TokenBearer*
- SecAction:
 - This class includes different actions that may be defined on a resource
 - Examples: *ActionDelete*, *ActionRead*, *ActionUpdate*, *ActionCreate*, *ActionPatch*

Security Policies Classes (Excerpt)

- CombAlg:
 - Policy contains one or more rules with associated combination algorithm
 - Defines a range of algorithms applied in the process of making a policy decision
 - Examples: *PermitOverride*, *DenyOverride*
- Context:
 - Establishes a range of conditions as multi-level classes whose instances may be evaluated in the process of making a policy decision
 - Examples: *TimeContext*, *SystemContext*, *RiskContext*
- Obligation:
 - Defines a range of actions that may be applied in the process of policy enforcement, both in pre and post resource delivery stages.
 - May include resource transformation using operations provided in the scope of resource sharing model
 - Examples: *LogPre*, *LogPost*, *CustomView*, *RemovePII*

Exposing Security Capabilities

- Example model using *core* and *email* vocabularies
 - *Nodes* are instances of concepts; labeling does not affect the workflow
 - Starts with *SecPolicy*, which refers to security rules and supported capabilities
 - The model is exposed for each endpoint or resource type separately, describing its specific capabilities

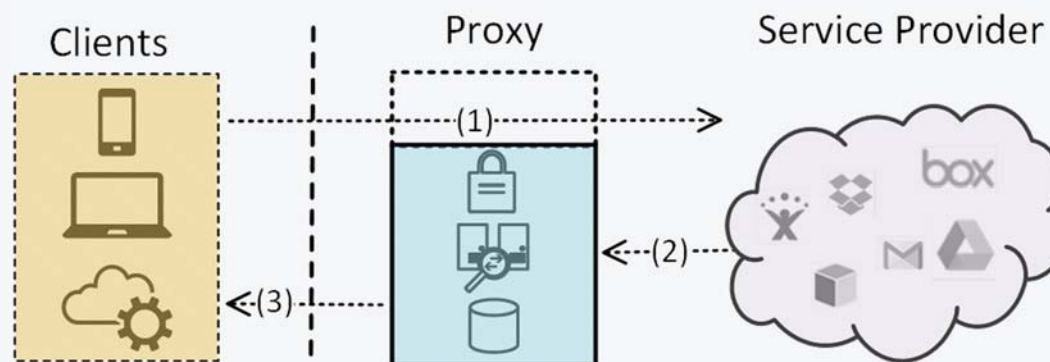


Defining Security Policies

- Discovering policy model
 - Client (resource owner's agent) fetches the model of policies applicable to particular resources or endpoints
 - The model represents a range of supported concepts (actions, subjects ...)
 - These concepts may be reused to define security policy
- Delivering the model
 - *Embedding models with resources*: retrieved together with resources by using web API endpoints and providing the models as separate descriptions
 - *Separate policy endpoints*: policy management is done using dedicated API endpoint
- Managing the policy
 - Based on retrieved model, the client instantiates the policy by applying provided concepts and relationships
 - These policies are retrieved or applied separately for each resource or endpoint

Integrating in the Workflow

- Limited proof-of-concept
 - Introduces additional management layer in existing OAuth 2.0 workflows
 - Transparent proxy between the service provider and the client
 - Complements OAuth 2.0 coarse-grained scopes by reducing the provided resource sets and enforcing client-specific resource view (resource transformation)
- Workflow steps
 - Client requests the resource using existing token [1]
 - Main application provides resource by relying on a token scope [2]
 - Based on a registered resource set and annotation returned by service provider, proxy performs second layer of security enforcement and delivers the resource [3]



Discussion

- Cross-entity messaging
 - Management of policies (RO), requesting resources (C), exposing models (SP)
 - Different possibilities: RDF or JSON-LD
 - JSON-LD - lightweight format, readable by humans (illustration at the bottom)
- Integrating in existing environments
 - Using additional HTTP headers for models and descriptions
 - Using HTTP methods (e.g. OPTIONS) to retrieve the models

```
{ "@context": {
  "dasp-email": "http://www.daspsec.org/on/dasp-email#",
  "owl": "http://www.w3.org/2002/07/owl#",
  "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
  "xsd": "http://www.w3.org/2001/XMLSchema#",
  "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
  "dasp-general": "http://www.daspsec.org/on/dasp-general#"
}, "@graph": [
  { "@id": "http://www.daspsec.org/on/dasp-general",
    "@type": "owl:Ontology", "owl:imports": [
      { "@id": "http://www.daspsec.org/on/dasp-general" },
      { "@id": "http://www.daspsec.org/on/dasp-email" } ] },
  { "@id": "dasp-general:AccessRequest",
    "@type": [ "dasp-general:Request", "owl:NamedIndividual" ],
    "dasp-general:acceptsOperation": { "@id": "dasp-general:RemovePII" },
    "dasp-general:acceptsSubtype": { "@id": "dasp-email:MsgHeader" },
    "dasp-general:requestsAccess": { "@id": "dasp-email:Email" } } ] }
```

Discussion

- Scalability and overhead
 - Per-resource or endpoint models and policies – scalable management
 - Low overhead for smaller models (transport or policy execution)
 - Separated vocabularies allow supporting different domains iteratively
- Interoperability challenges
 - Not everything can be covered by a model
 - It is hard to provide a model for complex concepts or environments
 - Integrating it in practical scenarios – even harder
 - However: conceptualized as a bottom-up approach
 - Providers may extend existing or define own models
 - Reducing many-to-many to (partial) many-to-one mappings

Summary

Questions and challenges

- Cloud-based integration adds complexity by executing processes and connecting resources hosted at various organizations
- Challenging management of security and privacy
 - Which resources and services are consumed by an organization
 - How to control resources and policies at different, distributed systems?
 - Automated management of security?
- Centralized systems such as iPaaS are gaining significant power:
 - Can access client's resources on many providers using broad access scopes
 - Multi-tenant environment: extrapolating that to all customers using iPaaS's services

Summary and Future Work

Summary

- Our proposal aimed to advance interoperability of security management
- Abstracting and modeling services, resources and policies
- Following lightweight, bottom-up approach
- Initial prototype was applied to extend management of existing OAuth 2 flows

Further work

- Extend vocabularies with additional domains, refine and improve existing
- Investigate integration with other protocols (UMA, XACML)
- Separate approach?
- Mapping of concepts and policies (automated alignment)

Any questions?



Thanks for your attention!