On The Structure and Authorization Management of RESTful Web Services

Bojan Suzic, Bernd Prünster and Dominik Ziegler

Graz University of Technology
Graz, Austria

Know-Center GmbH
Graz, Austria

The 33rd ACM Symposium on Applied Computing – SOAP Track
Pau, France - April 13th 2018

Overview

- Introduction and Motivation
- Approach and Methodology
- Properties and Access Control in APIs
- Design and Application of OAuth 2
- Summary and Conclusion

Introduction

Broad use of cloud services to process and share data among entities

Resources and operations typically exposed using Web APIs

ProgrammableWeb[1] currently lists more than 19 000 APIs

How popular platforms integrate security mechanisms in public interfaces?

Focus on *authorization*[2], a process of:

- … specifying access privileges to users or processes
- .. with the purpose to enforce access control over resources
- … ensuring high degree of security and privacy

[1] https://www.programmableweb.com
[2] Jøsang. A Consistent Definition of Authorization (2017)

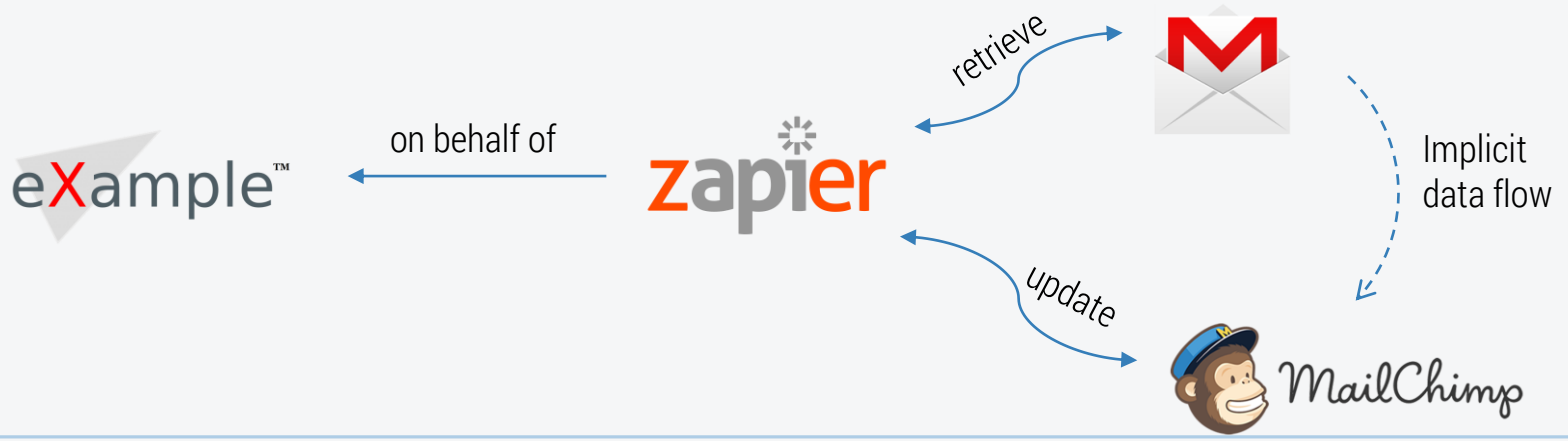
Motivational Scenario

Activity:

- *eXample Inc.* uses *Zapier* to automate its tasks using different services
- *Zapier* connects data sources from *Gmail* and *MailChimp* on behalf of a customer
- Web APIs (REST) typically applied to expose and share resources

Workflow:

- Periodically retrieve and extract email senders from recent emails at Gmail
- Add them as subscribers to a list at MailChimp



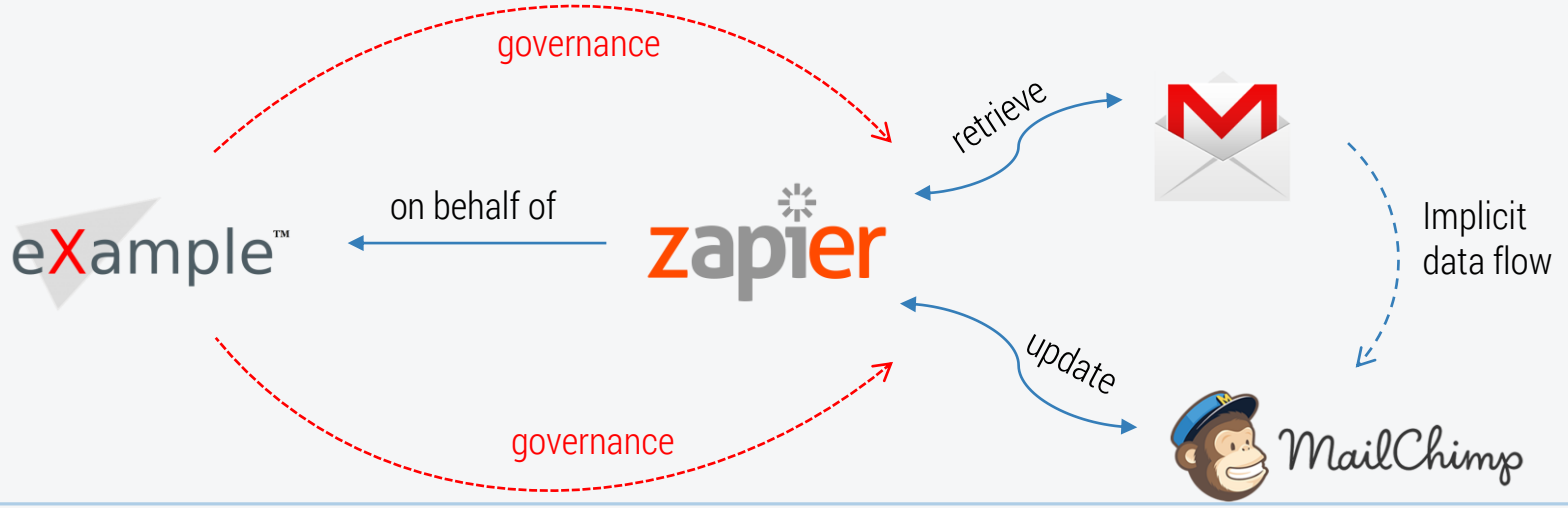
Motivational Scenario

Activity:

- *eXample Inc.* uses *Zapier* to automate its tasks using different services
- *Zapier* connects data sources from *Gmail* and *MailChimp* on behalf of a customer
- Web APIs (REST) typically applied to expose and share resources

Workflow:

- Periodically retrieve and extract email senders from recent emails at Gmail
- Add them as subscribers to a list at MailChimp



Approach and Methodology

Gather, process and evaluate structured API descriptions

Identify exposed authorization mechanisms and evaluate their use

OpenAPI [3] (Swagger) a dominantly applied approach to describe APIs

APIs.guru – largest directory of OpenAPI service descriptions

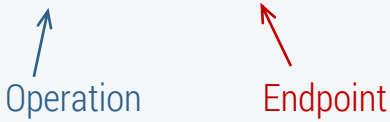
Crawled and processed n=523 API descriptions (September 2017)

[3] OpenAPIs terminology (formerly Swagger) – https://www.openapis.org

Properties of APIs

Service providers define *endpoints* (paths) and expose *operations* (methods) [4]

Example: **GET /messages/123**



Aggregate summary:

Measure	Number
APIs	523
Endpoints	11,664
Operations	14,991

[4] OpenAPIs terminology – https://www.openapis.org

Properties of APIs

API building blocks:

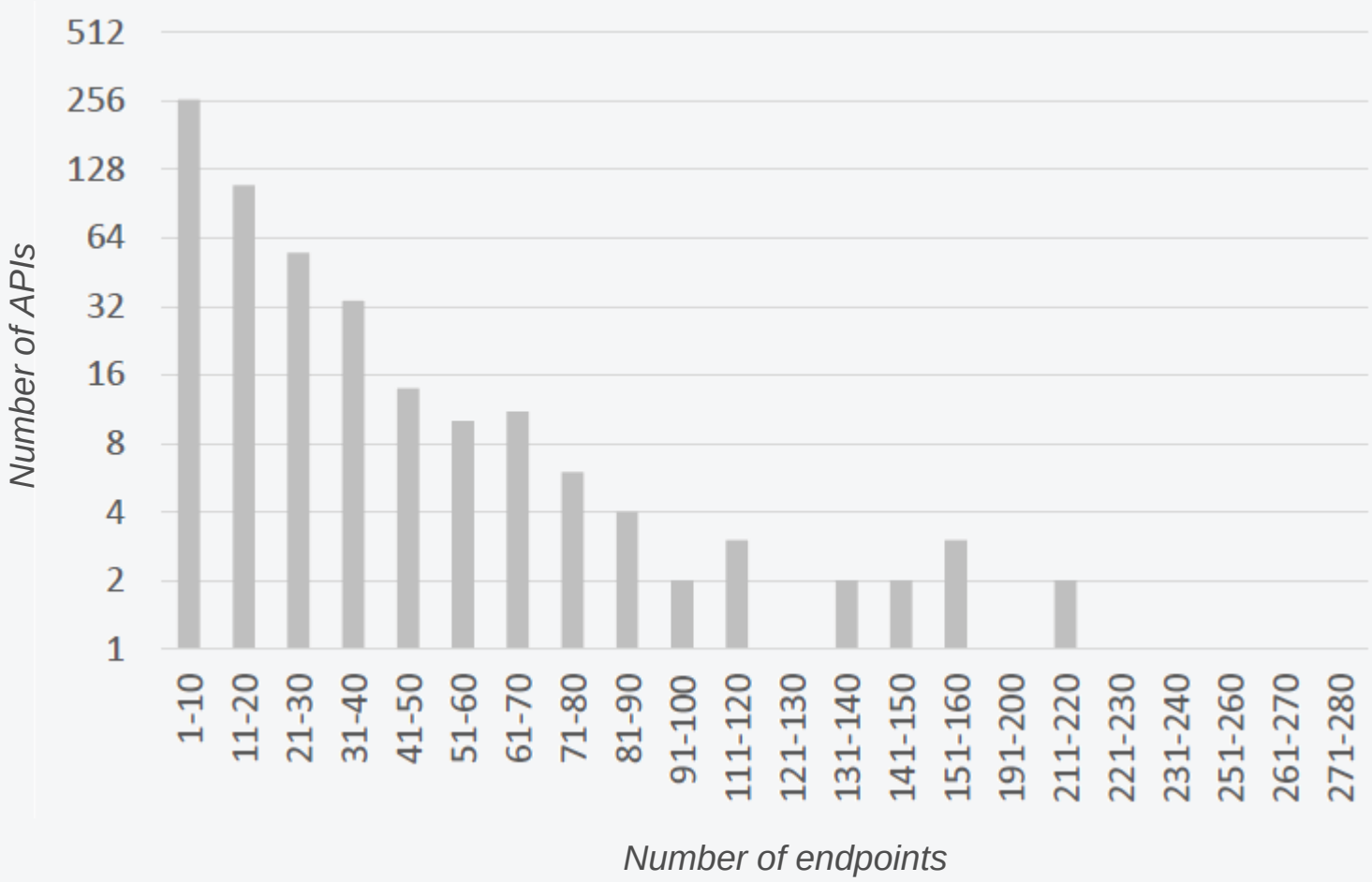
Measure	Average	Median
Number of endpoints	22.30	11
Number of operations	28.66	14
Number of methods	1.25	1

Distribution of HTTP methods applied over endpoints:

Method	Num.	%	Avg	Med
HEAD	16	0.11	0.15	0
DELETE	1,213	8.09	5.13	0
POST	**5,373**	**35.84**	**35.60**	**25**
GET	**6,726**	**44.87**	**52.25**	**50**
OPTIONS	2	0.01	0	0
PUT	1,245	8.30	4.87	0
PATCH	416	2.77	2.0	0

Properties of APIs

Distribution of endpoints across APIs:



Properties of APIs

Distribution of endpoints:

- 259 (49.5%) of APIs with <10 endpoints
- 368 (70.3%) of APIs with <20 endpoints

Application of HTTP methods:

- Low degree of method diversification
- GET and POST dominantly used
- *Unsafe methods*[5] comprise more than half of operations

[5] RFC7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content

Access Control in APIs

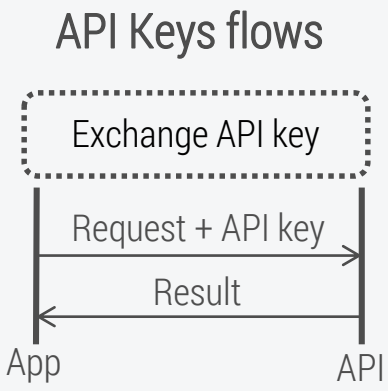
365 (69.79%) protected APIs vs 158 (30.21%) unprotected APIs

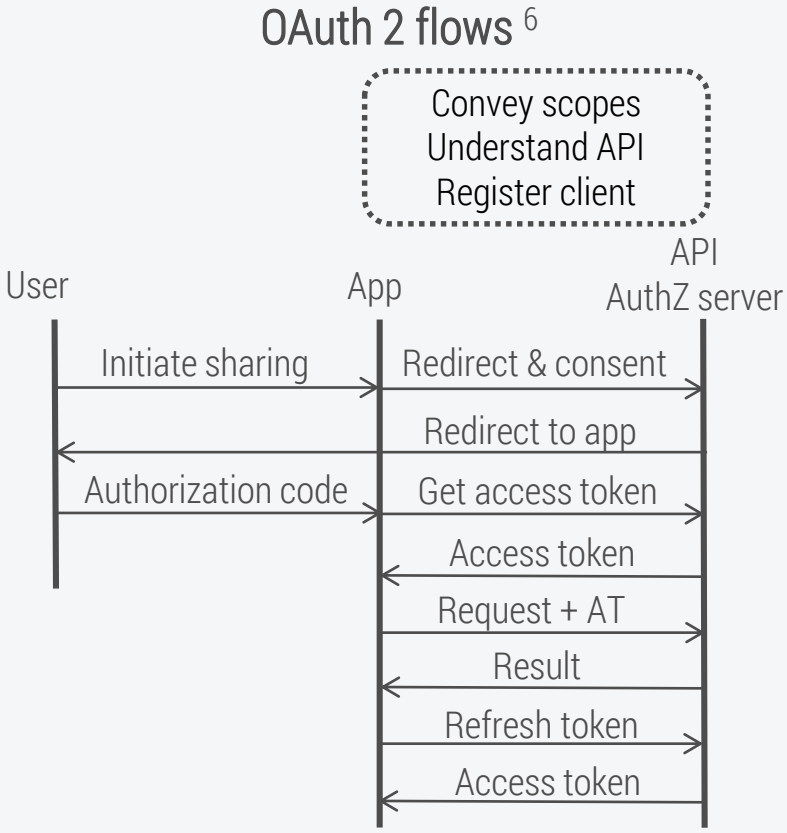
Declaration of security mechanisms:

Mechanism	APIs	%
HTTP	15	3.90
API Key	185	48.05
OAuth	177	45.97
Custom	8	2.08

20 APIs with two or more mechanisms

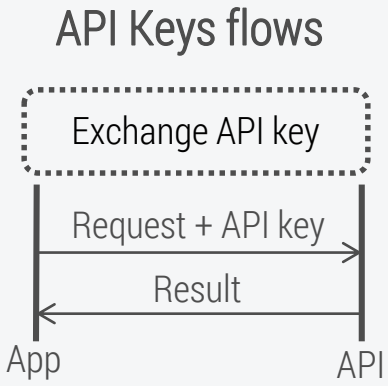
Access Control in APIs

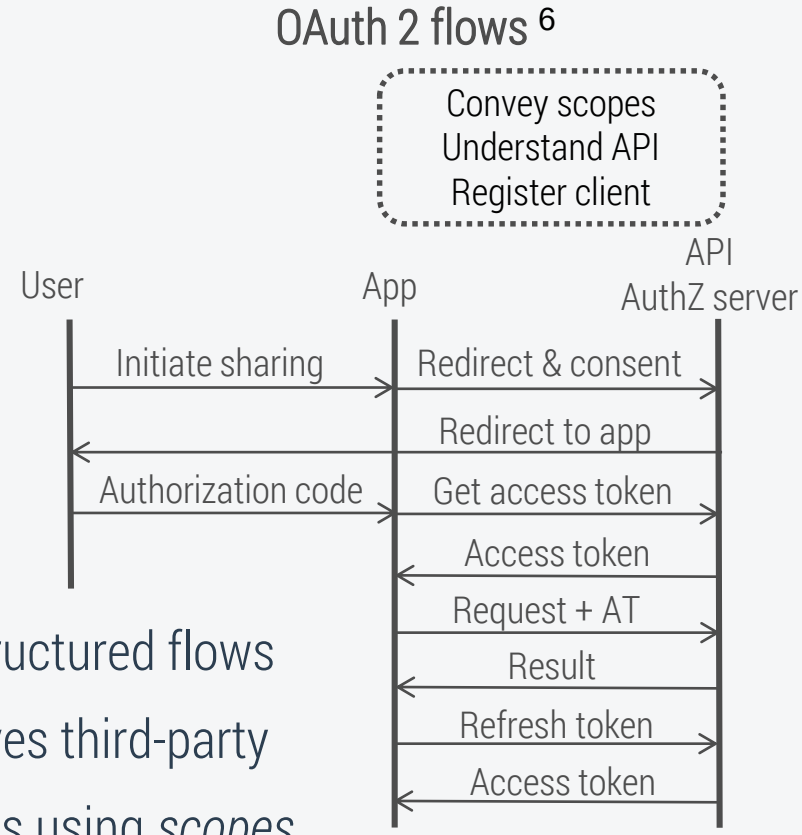




[6] RFC6749: The OAuth 2.0 Authorization Framework

Access Control in APIs





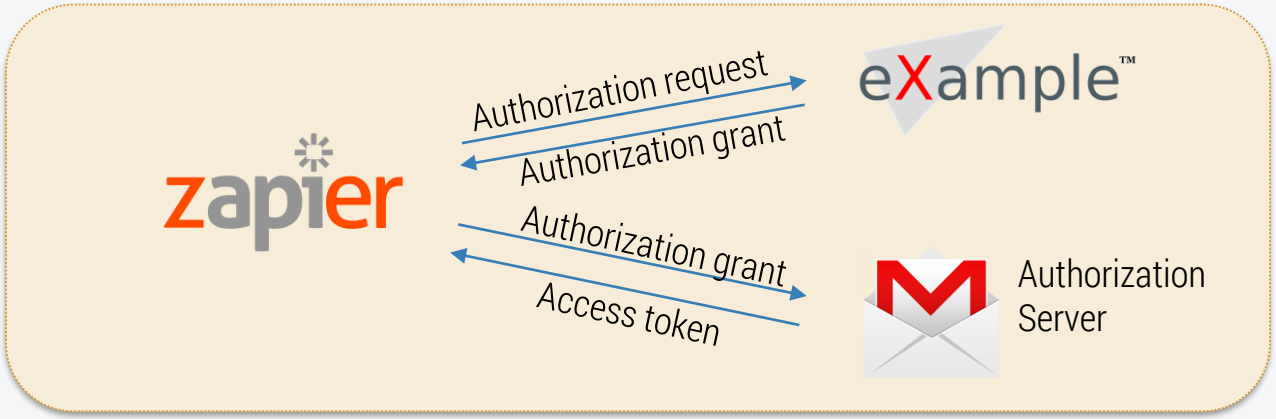
Key differences API keys vs OAuth 2

- AK manually issued, OA defines structured flows
- AK connects two parties, OA involves third-party
- OA introduces granular permissions using *scopes*
- OA uses mechanism to expiry and renew access tokens
- Authorization vs authentication, closed vs open, implicit vs explicit consent

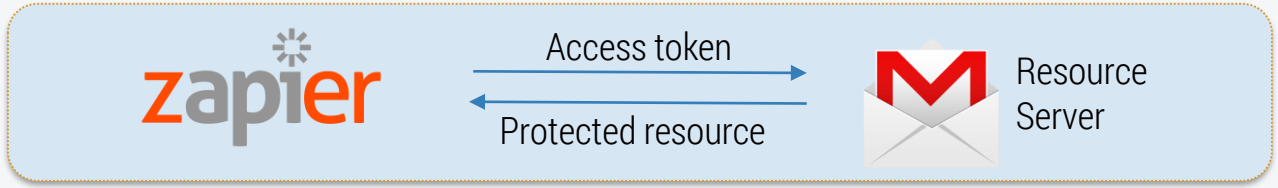
[6] RFC6749: The OAuth 2.0 Authorization Framework

Authorization Flows

Obtaining access token (initially)



Retrieving resource or performing operations (repetitive)



The same flow is applied in the case of MailChimp as well

Application of OAuth 2

Access scopes in OAuth 2

- Simple, opaque list of strings established by service provider
- Explained in human-readable documentation (for developers)
- Hard-wired, no dereferencing or semantics
- No guidelines on their definition and application
- Definition may change across the versions of an API

Example scope for Gmail[7]:

- gmail.readonly – provides access to 12 resources and 24 operations
  list emails, read all metadata and content, check user's history, inspect drafts, settings and labels

[7] https://developers.google.com/gmail/api/auth/scopes

Scopes in APIs

Declaration vs application of scopes in APIs

- 177 APIs declare scopes, 173 specified them
- 130 APIs applied scopes in descriptions, 89 use more than two scopes

Overview of specified scopes in APIs:

Scopes	APIs	%
One	80	46.24
Two	29	16.76
Three	13	7.51
Four	14	8.09
Five+	37	21.39

Design of scopes in APIs

Scope coverage:

- Describes the proportion of supported operations *op* of scope s in an API *α*
- Expressed as 0..1, with 1 implying that a scope supports all operations

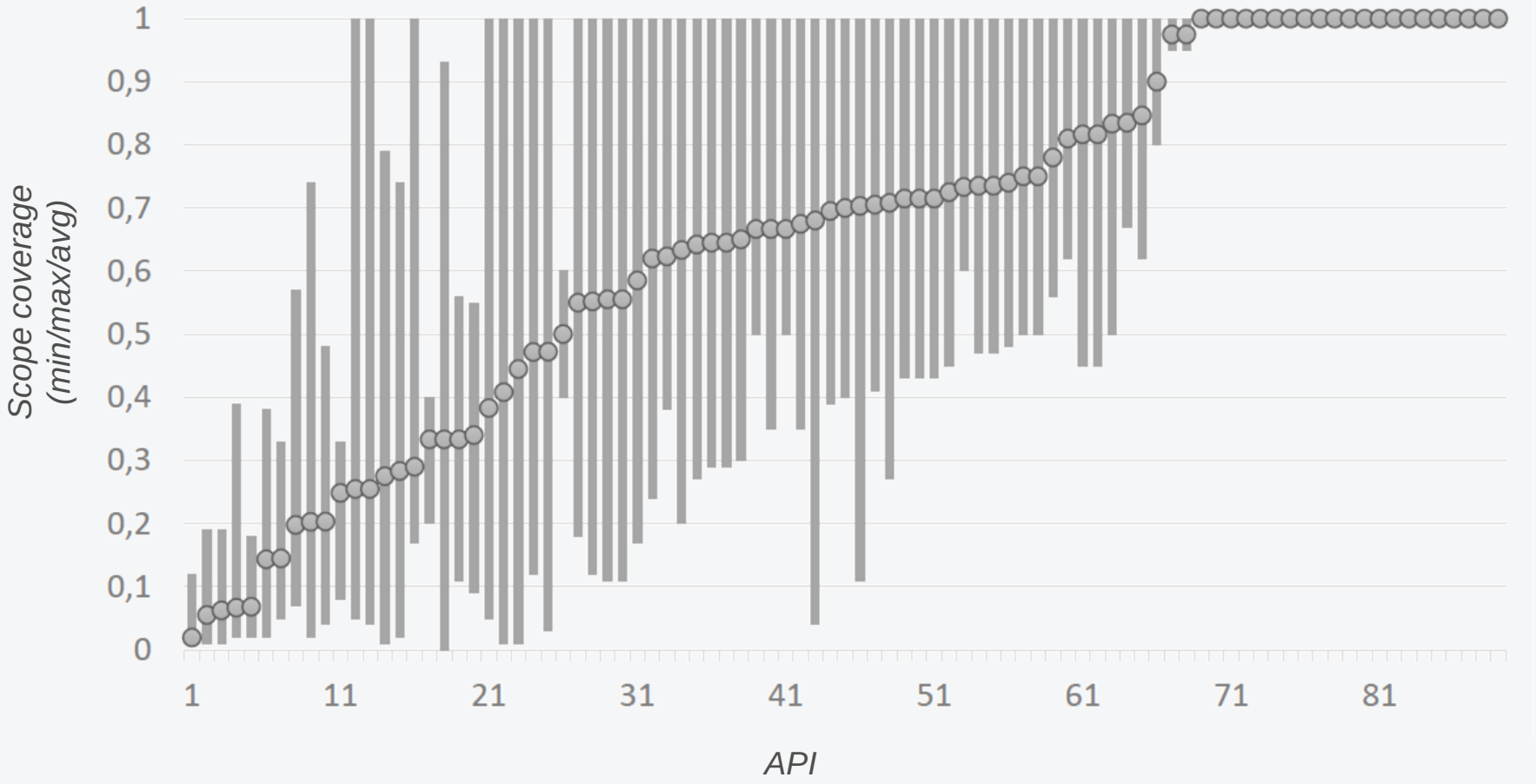
$$cov_s^{\alpha} = \frac{|op_s^{\alpha}|}{|op^{\alpha}|}$$

Scope similarity:

- Degree of operations shared between two scopes s_1 and s_2 in an API *α*
- Expressed as 0..1, with 1 implying a complete functional equivalence

$$sim_{s1,s2}^{\alpha} = \frac{|op_{s1}^{\alpha} \cap op_{s2}^{\alpha}|}{|op_{s1}^{\alpha} \cup op_{s2}^{\alpha}|}$$

Distribution of scope coverages



Design of scopes in APIs

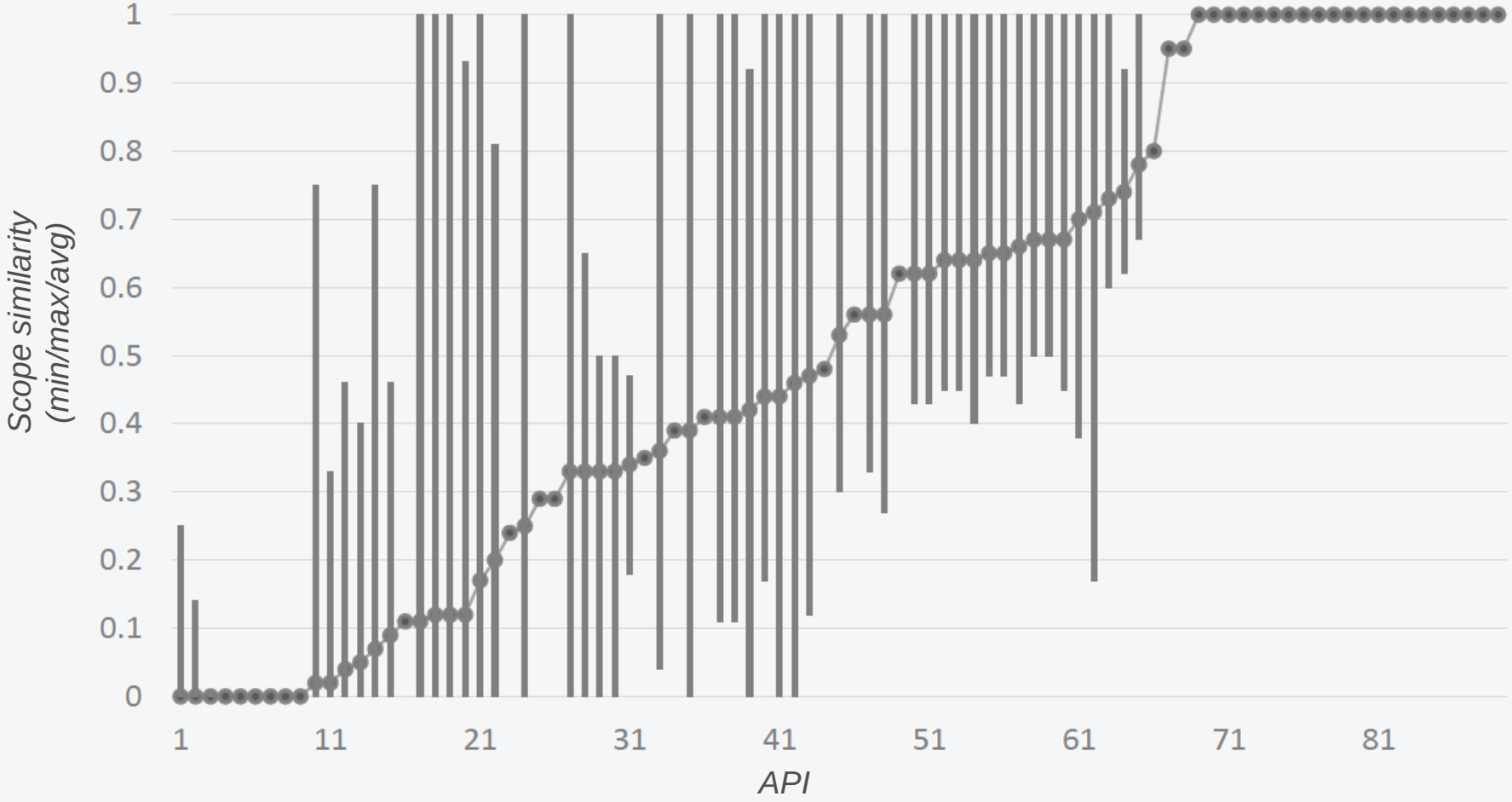
Both measures applied over reduced data set (n=89)

Scope coverage

- Nearly ¼ of services use scopes with complete coverage (*n*=21, *cov*=1)
- More than 70% of APIs with *avg_cov*>0.5
- 56% of services exhibit *min_cov*>0.33
- Only 11.2% (*n*=10) use scopes with *max_cov*<0.5
- Some providers introduce cross-API scopes

Relaxed, broader permissions observed in a typical API

Distribution of scope similarities



Design of scopes in APIs

Both measures applied over reduced data set (n=89)

Scope similarity

- Roughly ½ of services exhibit *sim_avg*>0.5
- 15.7% of APIs contain a scope with *max_sim*≤1/3
- 16.9% of APIs exhibit low average scope similarity (≤0.1)

This measure confirms the existence of broader permissions as well

Design of scopes in APIs

Application of HTTP methods in scopes

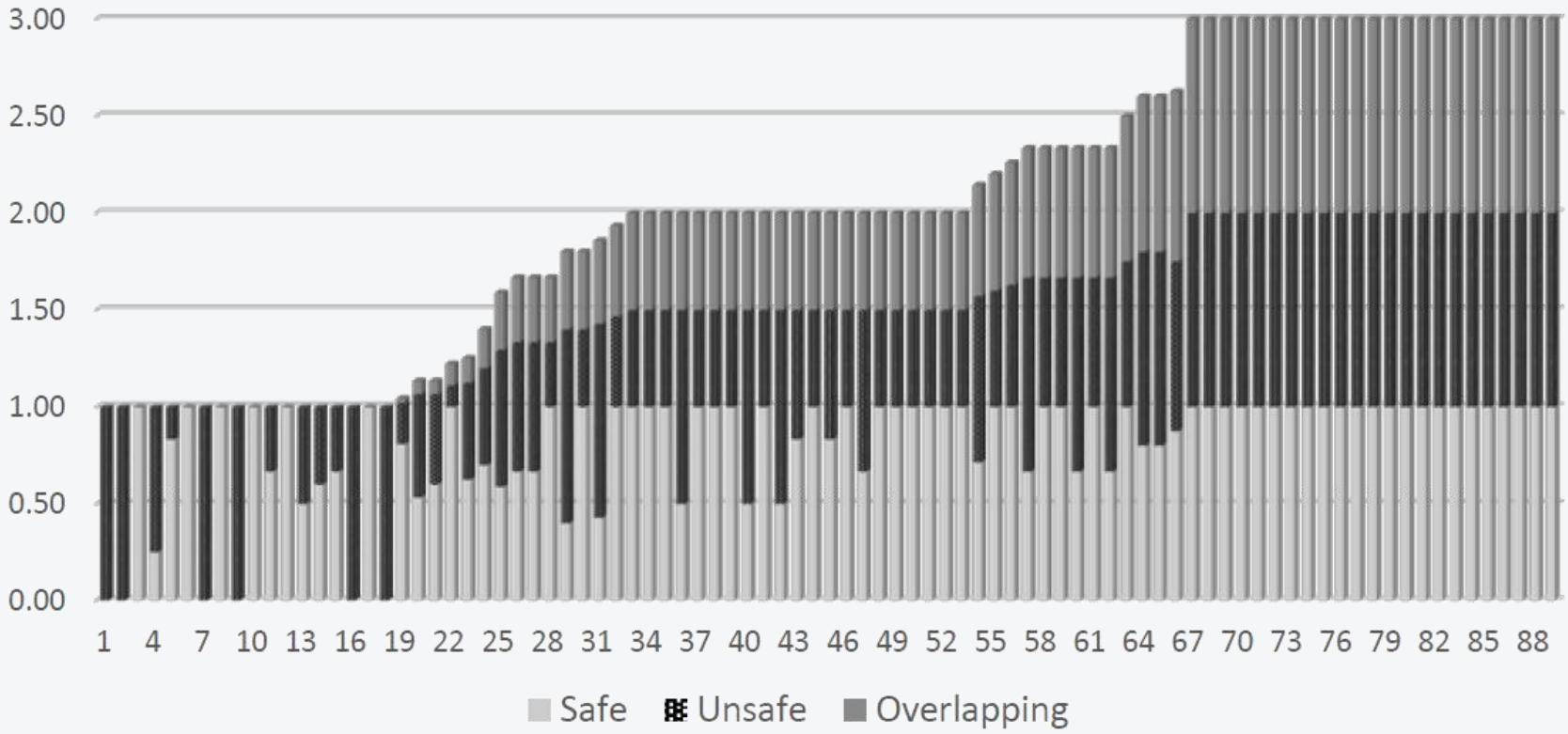
Considering distinction between *safe* and *unsafe* methods[5]

- Operation – an invocation of specific HTTP method over an endpoint (resource)
- Hence, each scope relates to one or more HTTP methods and endpoints
- *Overlapping scopes* – support both safe and unsafe methods

Deriving the degree of scopes that depend on safe and unsafe simultaneously

[5] RFC7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content

Application of HTTP methods in scopes



Application of HTTP methods in scopes

18 APIs (20.2%) specify scopes with no overlapping

23 APIs (25.8%) apply scopes with full overlapping

Other 48 APIs (54%) exhibit an partial overlap

APIs tend to employ scopes that combine safe and unsafe methods

- 57 APIs (64%) demonstrate overlapping of ≥0.5
- Use of unsafe methods correlates positively with overlapping (0.79, $p<0.01$)

API vendors tend to compartmentalize scopes into read-only, and read+modify

Summary

Other research points to *insecure implementations*[8] of OAuth 2

We consider the shortcomings in the permission model based on:

- Design issues
- Missing guidelines or good practices
- Implementation issues

[8] Wang et al. The Achilles heel of OAuth: a multi-platform study of OAuth-based authentication
Fernandes et al. Security analysis of emerging smart home applications.

Summary

Low diversity of permissions

- Most APIs (63%) declare no more than two scopes
- A typical API exhibits 28.66 (avg) or 14 (median) operations

Coarse-grained permissions leading to overprivileging

- Low number of scopes
- Broad coverage
- Significant overlap of operations in scopes
- Low distinction among single scopes
- Inconsistent relation with HTTP methods

Summary

Overall discrepancy in developing and applying mechanisms to protect resources

- Rich and deep research in access control models
- Low maturity/capability in case of cross-organizational data sharing

Design issues[9]

- Interoperability in vertical and horizontal dimensions
- Relating with API structures in machine readable way (dereferencing)
- Understanding and structuring the scopes across interacting entities
- Decentralizing definition and generation of authorization extent
- Static, context insensitive scopes
- Lack of instructional mechanism (supporting dynamic transformation)

[9] Suzic et al. Structuring the scope: enabling adaptive and multilateral authorization management.

Any questions?

Thanks for your attention!