Correlated Authorization—Draft

Igor Zboran
izboran@gmail.com

***Abstract*—*Correlated authorization* is a dual-authority authorization protocol built on top of User-Managed Access (UMA) [1, 2] and OAuth 2.0 Token Exchange [3] protocols that allows users (resource owners) to delegate access to other users (requesting parties) across security domain boundaries. The requesting party is responsible for creating the request, while the resource owner approves this request either when it is online or by creating a relationship-driven policy. The resource owner and the requesting party may belong to different security domains administered by the respective authorities.**

This concept uses a permission ticket issued by the resource owner's authorization server as a correlation handle that binds the requesting party's claims to the authorization process. An email address is used as the unique requesting party identifier for cross-domain access control. The intrinsic challenge-response authentication protocol elevates trust between the resource owner's authorization server and the requesting party's authorization server, which also acts as the identity provider.

I. Introduction

With the growing popularity of protocols based on the OAuth 2.0 [4] specification, there is a need for an interoperable standard that specifies how to convey information about the user from an identity provider to an authorization server, especially across security domain boundaries. The problem is that such a system is difficult to design because OAuth2 [4], OIDC [5] and UMA are single-authority protocols. This draft profiles and combines the OAuth2 and UMA protocols into a dual-authority protocol, which not only meets the needs of interoperability, but also elevates trust between mutually unknown parties.

II. Motivation

Correlated authorization is an attempt to revive UMA WG's original idea—UMA wide ecosystem [6], when the resource owner and requesting party might "know each other" in the real world, but the resource owner's authorization server has no pre-established trust with the requesting party or any of their identity/claims providers—in other words, when the resource owner's authorization server and requesting party's identity provider don't know each other.

III. UMA Wide Ecosystem Concept

This high-level view illustrated in Figure 1 gives you an idea of relationships between UMA wide ecosystem entities.

UMA uses a special jargon. For the sake of brevity of this paper, the following list of acronyms will be used:

- IdP - Identity Provider
- AS - Authorization Server
- RS - Resource Server
- RO - Resource Owner
- RqP - Requesting Party
- RPT - Requesting Party Token

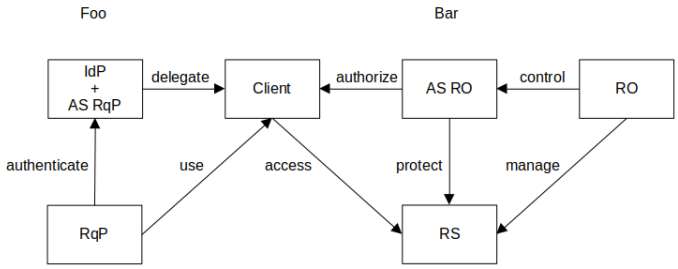

Fig. 1. Relationships between UMA wide ecosystem entities

The UMA wide ecosystem concept uses relationship-driven policies to drive automated dual-authority authorization assessment and token issuance. The relationship-driven policies incorporate user-to-user (U2U) relationships and user-to-resource (U2R) relationships.

IV. Challenge-Response Authentication Concept

The unilateral entity authentication protocol [7] illustrated in Figure 2 elevates trust between the resource owner's authoritative domain and requesting party's authoritative domain.

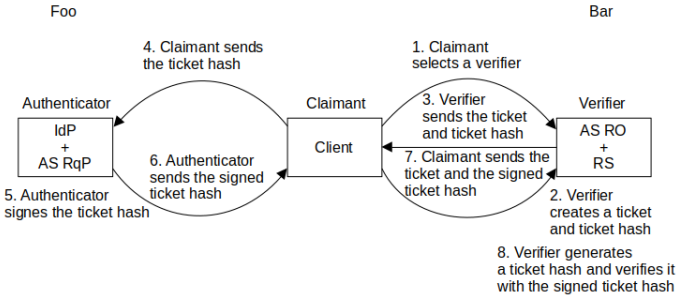

Fig. 2. Unilateral entity authentication protocol

The ticket represents a random challenge and the signed ticket hash represents the response. The hash of the ticket has to be there in order not to reveal the ticket to the authenticator.

V. Sequence Diagrams

The following sequence diagrams describe the mechanism of the *correlated authorization* protocol, which relies on the token exchange extension of OAuth2, where an access token is used to obtain a claims token from the Security Token Service (STS) endpoint.

UMA Profile

The sequence diagram illustrated in Figure 3 represents a profile of the UMA protocol and is in full compliance with the UMA 2.0 specification. [1]

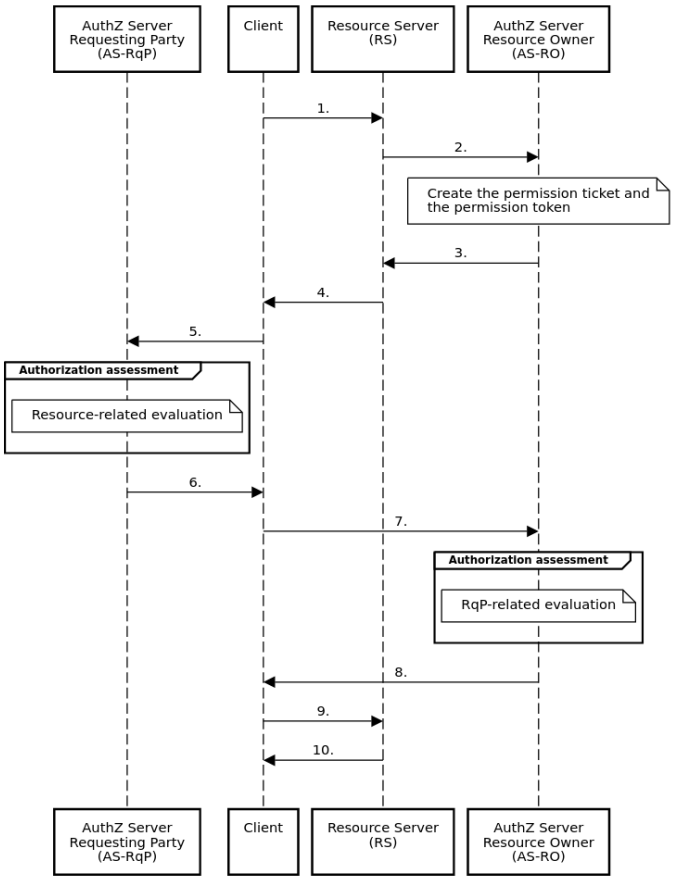

Fig. 3. *Correlated authorization* sequence diagram

Prerequisites:

- The AS-RqP supports the OAuth 2.0 Token Exchange [3] extension of OAuth2.
- The AS-RqP publishes its metadata on a URL /.well-known/oauth-authorization-server (alternatively on /.well-known/openid-configuration).
- The AS-RqP also acts as RqP's Identity Provider.
- The client is registered at the AS-RqP as a public or confidential client and acts as a Relying Party in a RqP's Identity Provider in order to obtain an access token with user claims.
- The client should be registered at the AS-RO as a public or confidential client; in case of immediate access, the client does not have to be registered at the AS-RO.
- The RO has set up the RS and registers his 'RS API' resource at the AS-RO according to the UMA Federated Authorization [2] specification.

Steps:

1. The RqP directs the client to access the 'RS API' resource with no access token.
2. The RS requests a permission ticket.

   The AS generates the permission ticket itself (ticket is a random NONCE) and the permission token[2], which is bound to the permission ticket through a permission ticket hash. The permission token contains these claims: {issuer, ts, rs_uri, resource_name_hash, permission_ticket_hash} where
   - issuer is the URI that identifies who issues the permission token
   - ts is the timestamp of when the permission ticket was created
   - rs_uri is the URI that identifies the resource server
   - resource_name_hash = Base64URL-Encode(SHA256(resource_name))
   - permission_ticket_hash = Base64URL-Encode(SHA256(permission_ticket))

3. The AS returns the permission ticket and the permission token.
4. Without an access token, the RS will return HTTP code 401 (Unauthorized) with the permission ticket and the permission token.
5. The client requests a claims token by presenting the access token with user claims, permission token and resource name (token exchange request).

   {grant_type = token-exchange,
   resource = "RS API",
   scope = permission_token resource_name,
   subject_token = access_token_with_user_claims,
   subject_token_type = urn:ietf:params:oauth:token-type:access_token,
   requested_token_type = urn:ietf:params:oauth:token-type:jwt}

   The AS-RqP performs an authorization assessment
   - 1. verify permission_token
   - 2. compare resource_name_hash vs. Base64URL-Encode(SHA256(resource_name))
   - 3. evaluate issuer, ts, rs_uri, resource_name

   The AS-RqP generates the claim token, which contains these claims: {user_claims, permission_ticket_hash} where
   - user_claims are extracted from access_token_with_user_claims
   - permission_ticket_hash is extracted from permission_token

6. After an authorization assessment, it is positive, the AS-RqP returns the claims token.
7. At the AS-RO the client requests an RPT by presenting the claims token and the permission ticket.

   {grant_type = uma-ticket,
   pushed_claims = claims_token}

   The AS-RO performs an authorization assessment
   1. verify permission_ticket
   2. extract user_claims from claims_token
   3. select email_address claim

1. Unlike the UMA specification, the *correlated authorization* protocol allows the use of the UMA grant with or without client authentication or identification. Whether or not to allow unauthenticated or unidentified clients are policy decisions that are at the discretion of the authorization server.

2. The permission token is not mentioned in the UMA specification. A detailed description of the permission token format is out of scope of this paper.

4. bootstrap discovery of AS-RqP config url from email address via WebFinger;
   if this doesn't work, build well-known url using domain part of email_address
5. verify claims_token signature
6. evaluate resource = "RS API"
7. extract permission_ticket_hash scope from claims_token
8. compare permission_ticket_hash vs. Base64URL-Encode(SHA256(permission_ticket))
9. evaluate user_claims

8. After an authorization assessment, it is positive, the AS-RO returns RPT.
9. With the valid RPT the client tries to access the 'RS API'.
10. The RS validates the RPT, it is valid, the RS allow access the protected 'RS API' resource.

VI. Authority Boundaries, Interactions and Scenarios

The *correlated authorization* protocol allows us to indirectly (through the client) link identity providers with authorization services governed by different authorities that are not required to share information or collaborate.

The following scenarios demonstrate a system of trust between two authorities that allows the conveyance of identity information from identity providers to authorization services across security domain boundaries.

A. Identity Federation Scenario

The scenario illustrated in Figure 4 allows you to link a single authorization service to multiple identity providers. The client falls under the governance of the resource owner's respective authority.

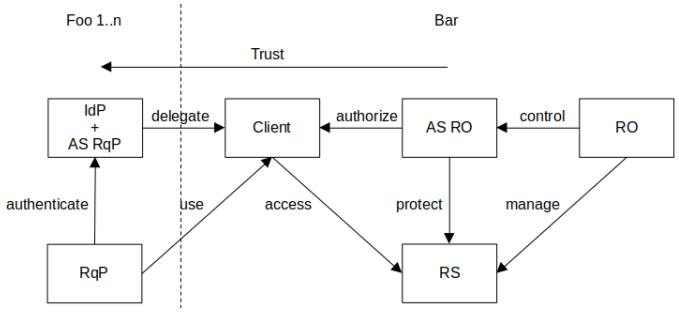

Fig. 4. Identity federation scenario

B. Federated Authorization Scenario

The federated authorization scenario illustrated in Figure 5 allows you to link a single identity provider to multiple authorization services. The client falls under the governance of the requesting party's respective authority.

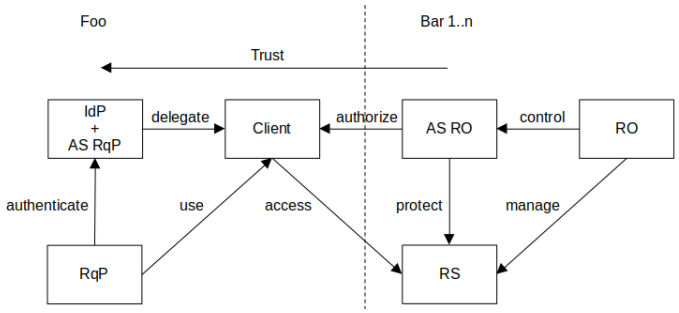

Fig. 5. Federated authorization scenario

C. Combined Federation Scenario

As the name suggests, the scenario illustrated in Figure 6 allows multiple authorization services to be linked to multiple identity providers. The client falls under the governance of a third-party authority.

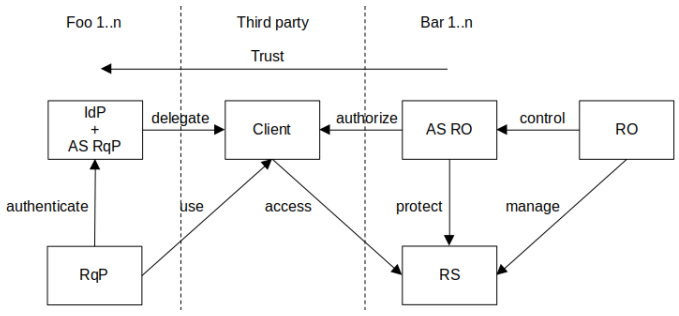

Fig. 6. Combined federation scenario

VII. Use Cases

Healthcare and enterprise cross-domain services e.g. email, file sharing, instant messaging, tele-conferencing. Also, Fintech and Telco services.

VIII. Future Work

1. Consider an authentication protocol, where RS/AS acts as an external authoritative attribute/claims provider.
2. Employ the DPoP to bind RPT to the client.
3. Describe how the resource owner can use the *correlated authorization* protocol.
4. Consider using the *correlated authorization* mechanism to transfer digital/virtual assets in the form of transactions.

Acknowledgment

This work has benefited from the valuable discussions with Eve Maler, founder of WG-UMA [8]; and Alec Laws, chair of WG-UMA [8]. Both gave feedback that improved this paper's content. Last but not least, the UMA Work Group archives [9, 10] serve as a source of comprehensive information on authorization-related topics—many thanks to all involved.

References

[1] E. Maler, M. Machulak, J. Richer, and T. Hardjono, "User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization," Internet Engineering Task Force (2019), https://datatracker.ietf.org/doc/draftmaler-oauth-umagrant-00.

[2] E. Maler, M. Machulak, J. Richer, and T. Hardjono, "Federated Authorization for User-Managed Access (UMA) 2.0," Internet Engineering Task Force (2019), https://datatracker.ietf.org/doc/draftmaler-oauth-

umagrant-00.
[3] M. Jones, A. Nadalin, B. Campbell, J. Bradley, C. Mortimore, “OAuth 2.0 Token Exchange,” RFC 8693 (2020), https://rfc-editor.org/rfc/rfc8693.txt.
[4] E. D. Hardt, “The OAuth 2.0 Authorization Framework,” IETF RFC 6749 (Informational), 2012, http://tools.ietf.org/html/rfc6749.
[5] OpenID specifications at “OpenID Foundation,” 2022, https://openid.net/developers/specs/.
[6] “UMA telecon 2016-03-31” https://kantarainitiative.org/confluence/display/uma/UMA+telecon+2016-03-31
[7] National Institute of Standards and Technology, “FIPS PUB 196: Entity Authentication Using Public Key Cryptography,” 1997. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/fips/196/archive/1997-02-18/documents/fips196.pdf.
[8] “User-Managed Access” Work Group at “Kantara Initiative” https://kantarainitiative.org/confluence/display/uma/Home.
[9] “The WG-UMA Archives” https://kantarainitiative.org/pipermail/wg-uma/.
[10] “Kantara Initiative User Managed Access WG” https://groups.google.com/g/kantara-initiative-uma-wg.