Authorization-Enhanced Email System

Igor Zboran
izboran@gmail.com
September 6, 2020

Abstract

Electronic mail (email) is the most pervasive form of business information exchange. Email is often used not only as an interpersonal communication tool, but also as the default choice to send files. In this paper the User-Managed Access (UMA) authorization framework is proposed to address data storage, access control and data transfer limitations of current email systems. Outgoing mail is typically transferred from the source system to the destination system as a single text-encoded file using Simple Mail Transfer Protocol (SMTP). SMTP is a push protocol only. The UMA framework introduces a resource server and an authorization server into the email system. The resource server is accessed generally by HTTP protocol that was designed as a pull protocol. The two-way push-pull data transfer in combination with a data storage system controlled by the standardized authorization framework significantly leverages email security, enhances email system utilization and elevates the email ecosystem to the ubiquitous Content Services Platform.

Introduction

The main components of the email system have been designed between 1971 and 1992 by many inventors. In the course of time, email has become the most commonly used application of the Internet. Nowadays the email is the only truly decentralized communication system of the Internet and the email infrastructure forms the backbone of the worldwide digital identity.

Problem

Despite the importance of email infrastructure, the whole ecosystem still relies on over 40 year-old architecture and protocol design. There are spam and attachment issues from the very beginning. The email system, while conceptually sound as a communication means, is structurally obsolete and functionally deficient.

Current Situation

With the rising popularity of free email providers, such as Gmail or Outlook.com, web-browsers are increasingly being used to access email server. From a user standpoint, it is easy to read and send email via web-browser on any device, from anywhere in the world. Centralized access to the mailboxes, increases the security of web-based email systems.

Current Flaws

Even though the main email service providers claim email accounts to be safe, the fact remains that major security and functional flaws are not fixed. There is still an attachments delivery dichotomy; the bulky files are not transferred as an attachment but are shared via links. An “attachment sharing” is not natural for postal systems where each message with attachments is expected to be consistent. Shared links pose a consent phishing attack threat where attacker tricks users into granting a malicious application access to sensitive resources. This is known as an OAuth 2.0 authorization exploit. The Authorization-Enhanced Email System is resistant to this security exploit as there are no direct user involvement in access granting.

Proposed Solution

Given that email system is lagging behind modern communication and collaboration tools, we propose an OAuth-based access control management and consequently a new data exchange channel for the email ecosystem.

Motivation

Email still the most popular communication tool is lacking an important part of today's modern systems – an authorization framework. Understanding this lead us to implement the UMA authorization framework into the email ecosystem.

Main Concept

We propose to incorporate the UMA framework between the email system with standardized SMTP/POP3/IMAP interface and the proprietary RESTful web-based email (Webmail) application, illustrated in Figure 1.

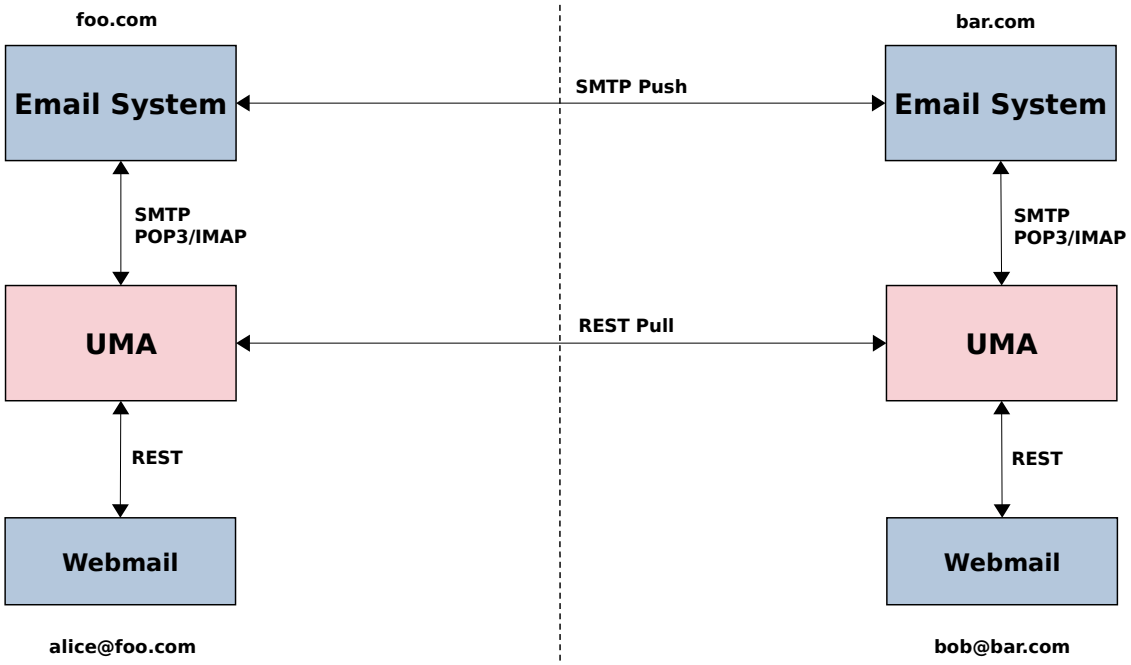

Figure 1. Main concept

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

- AS Authorization Server
- RS Resource Server
- RO Resource Owner
- RqP Requesting Party
- PAT Protection API (access) Token
- RPT Requesting Party Token
- PCT Persisted Claims Token

We introduce some new acronyms:

- AEES Authorization-Enhanced Email System
- MFA Mail Fetch Agent
- OAT OAuth (access) Token

The UMA framework plays its role during the data exchange process between mailboxes. The Webmail application gives the sender a user-centric approach to manage and protect his/her ready-to-send resources while the MFA that acts on behalf of the recipient is used to access and download sender's resources as it is illustrated in Figure 2 and Figure 3.

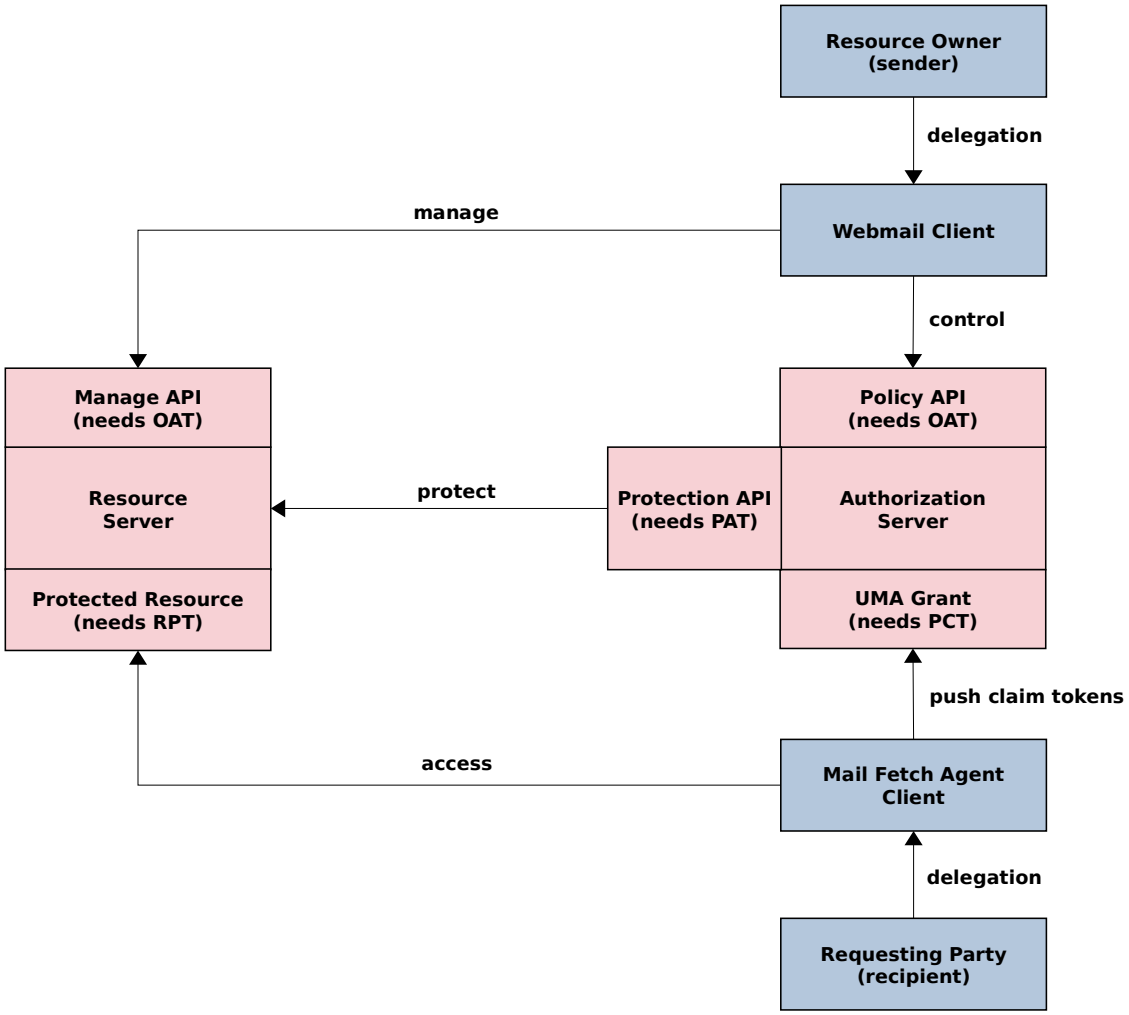

Figure 2. AEES/UMA abstract flow

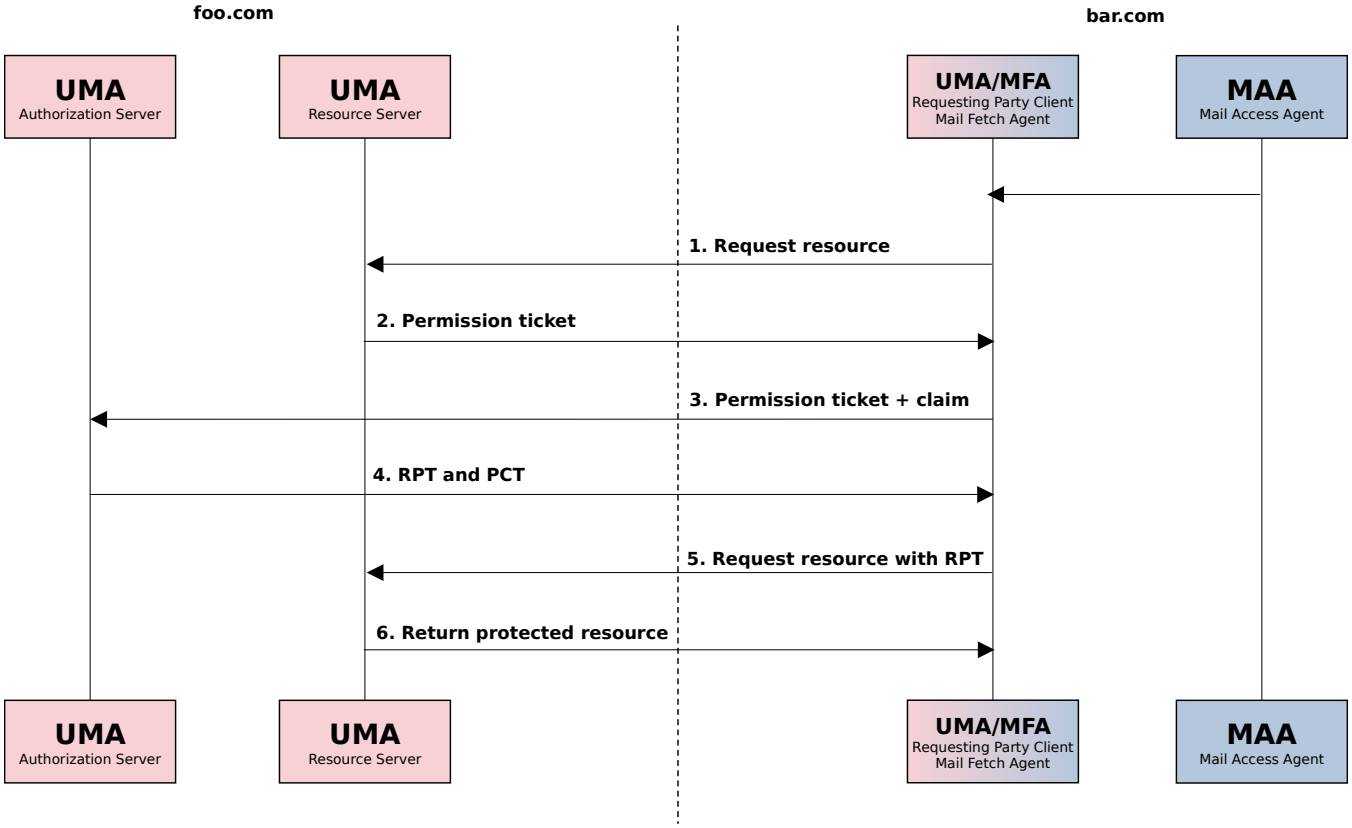

Figure 3. AEES/UMA Grant sequence diagram (with pushed claim)

Notes:

● On the server side no user content is stored in the messages, only links to the UMA RS (message body link, attachment links).
● The actual content of messages and attachments is stored on the UMA RS.
● Webmail application assembles/disassembles email that is stored in the RS.
● The user communicates with the UMA RS via Webmail application (REST protocol).
● Email is transferred by parts (envelope, body, attachments).
● A user as a sender in a RO role shares temporary links to the actual message content (disassembled email) with recipients and sends dummy email message with links to the RS via email system (SMTP protocol) to recipients. This is done automatically, user always works with assembled email in Webmail application.
● On the recipient side the incoming email with temporary shared links is processed in Mail Fetch Agent (MFA) that acts in a RqP role on behalf recipient and actual message content is downloaded from sender's RS and are copied to recipient's RS. Recipient becomes an owner of copied content. The temporary shared links on the sender side are deleted.
● The original email is assembled from the recipient's RS data in the recipient's Webmail Application.

Trust Model

Figure 4 illustrates the decentralized three-way trust relationship model:

- Mail Trust – SMTP to SMTP trust (the most vulnerable).
- Mail to UMA Trust – a trust delegation from the email system to the UMA framework.
- UMA Trust – a trust between UMA components.

There is no contract between authorization servers and UMA roles remains co-located.

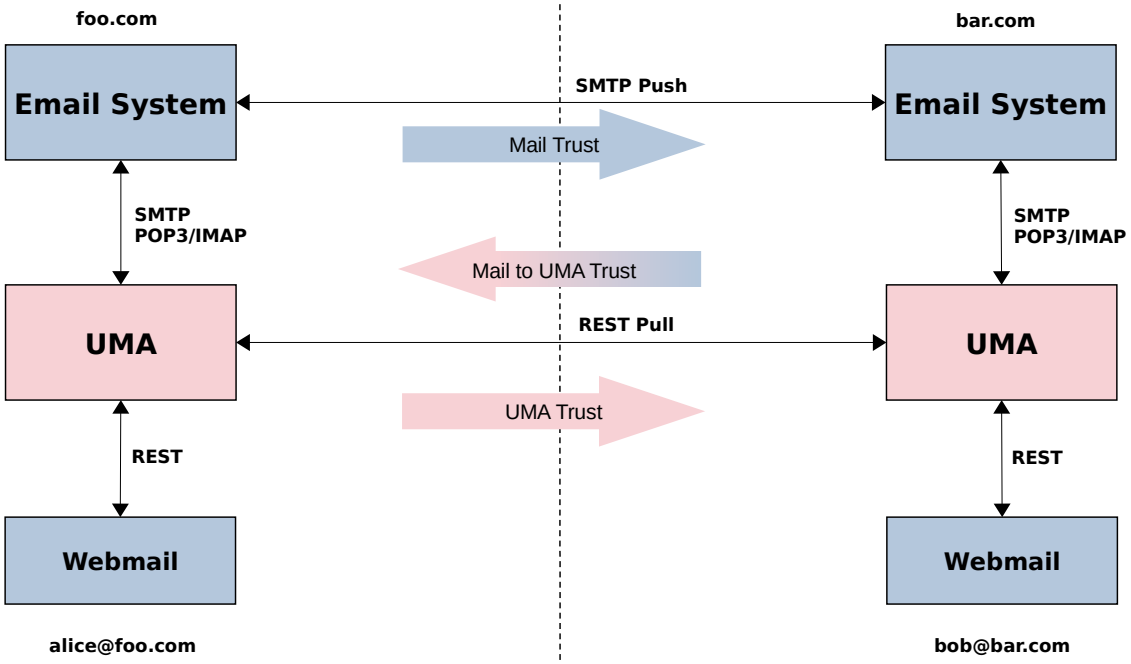

Figure 4. alice@foo.com to bob@bar.com trust model

Scenarios and Flows

The following scenario represents an email sent from alice.foo.com address to bob.bar.com address assuming that this is the first ever communication between the foo.com and bar.com security domains and no previous trusted relationships was established between them. Trusted communication between users in the two security domains can be divided into an one-time initial registration action and three main phases.

Registration action

- get authorization, register, set up a relationship

Before the communication itself, a trust relationship governed by a contract must be established between the bar.com RqP client (aka MFA) and the foo.com AS. To set up a relationship a registration message in an email must be sent from foo.com domain to the bar.com domain. The sending of the registration message must be authorized by the foo.com AS. To make this process streamlined OAuth 2.0 Dynamic Client Registration Management Protocol (RFC 7592) is used to avoid manual registration workflow as it is illustrated in Figure 5. The Initial Access Token and the Software Statement are sent in the registration message. This registration message is generated by the foo.com RS and is typically bundled with the main email message sent by the user.

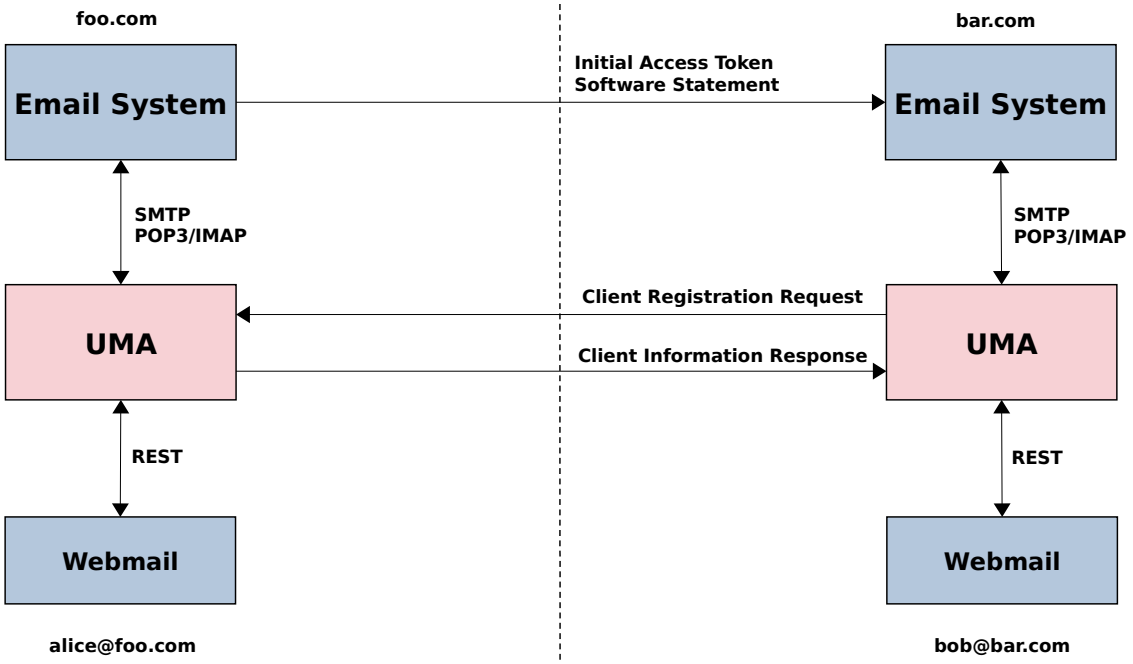

Figure 5. alice@foo.com to bob@bar.com dynamic client registration

After registration a trust relationship is set up between bar.com RqP client and the foo.com AS.

Phase I

- put resources under protection, create policy, send resources links to the recipient(s)

Let’s assume…

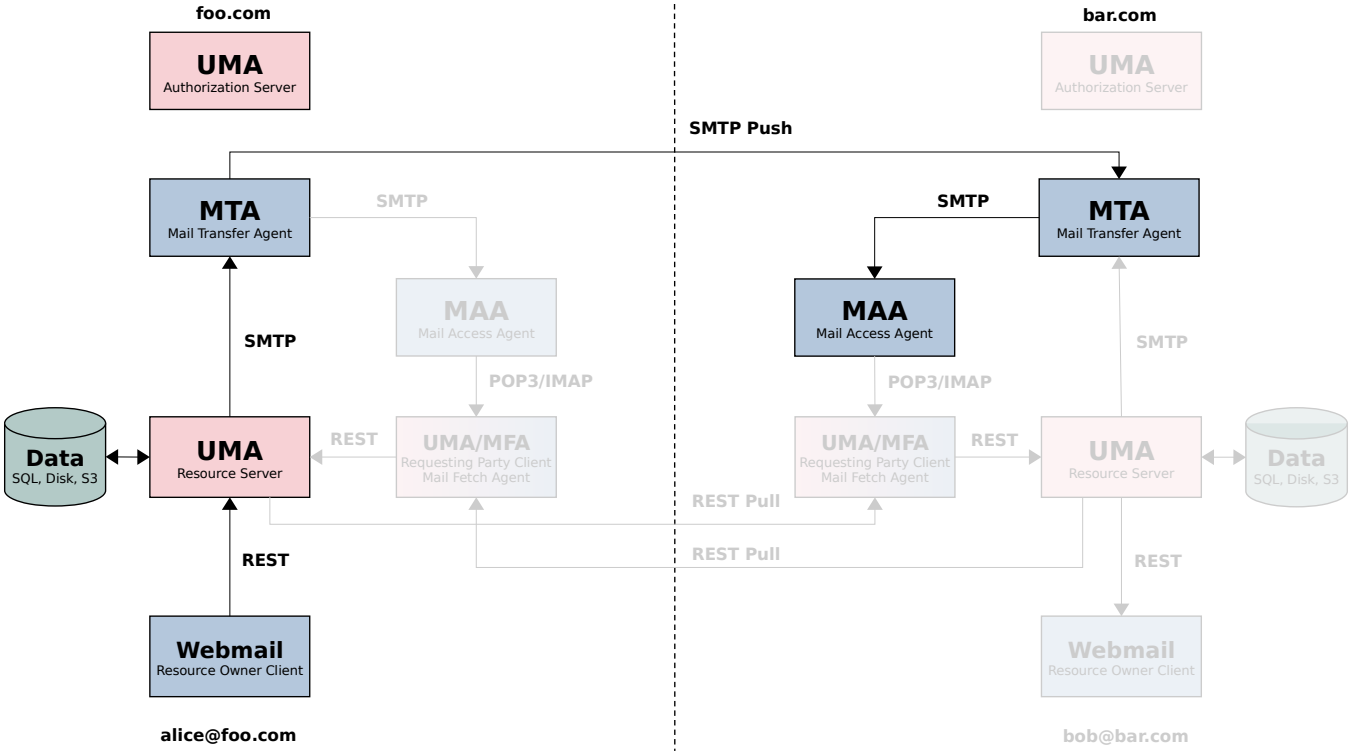

Figure 6. alice@foo.com to bob@bar.com send resources links

Phase II

- notify RqP client (MFA), get resources links, push claims, get authorization

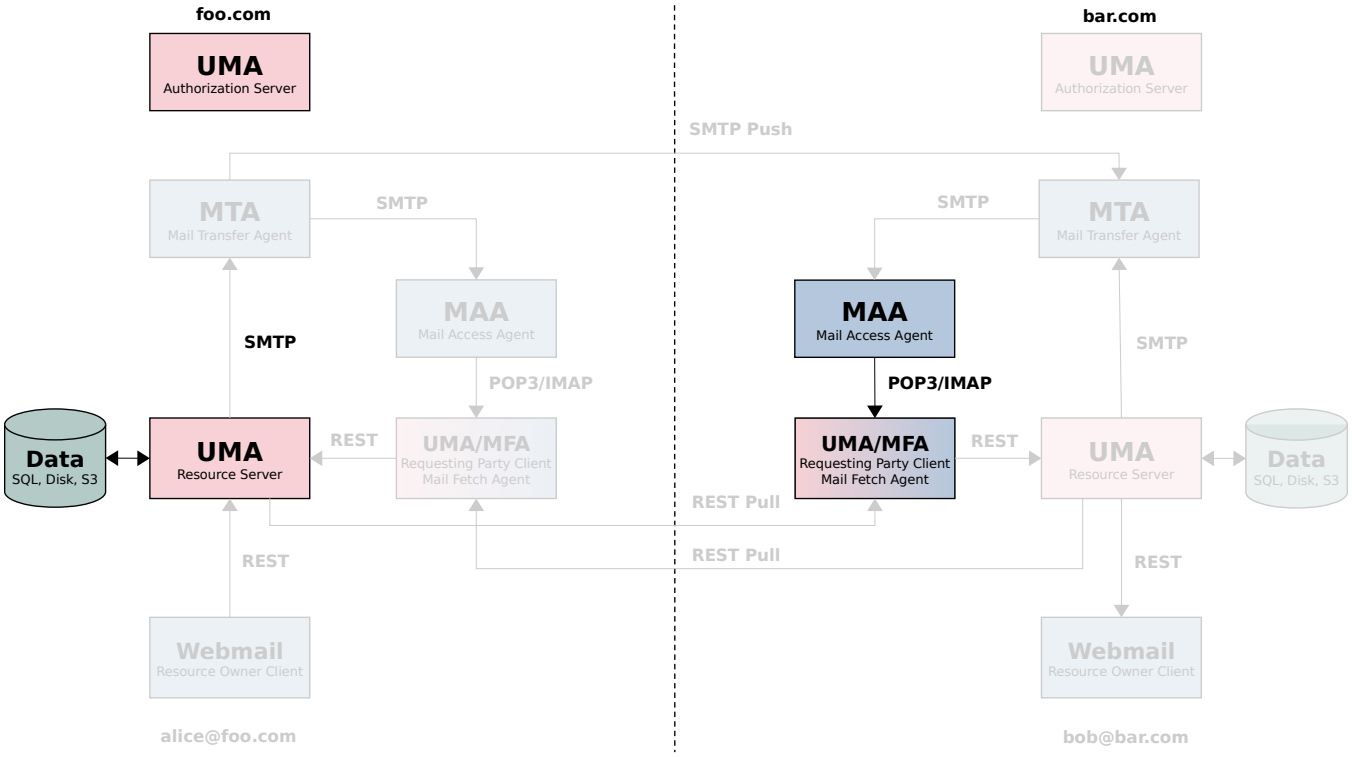

Figure 7. alice@foo.com to bob@bar.com get authorization

Phase III

- get data, notify the Webmail client

The ...

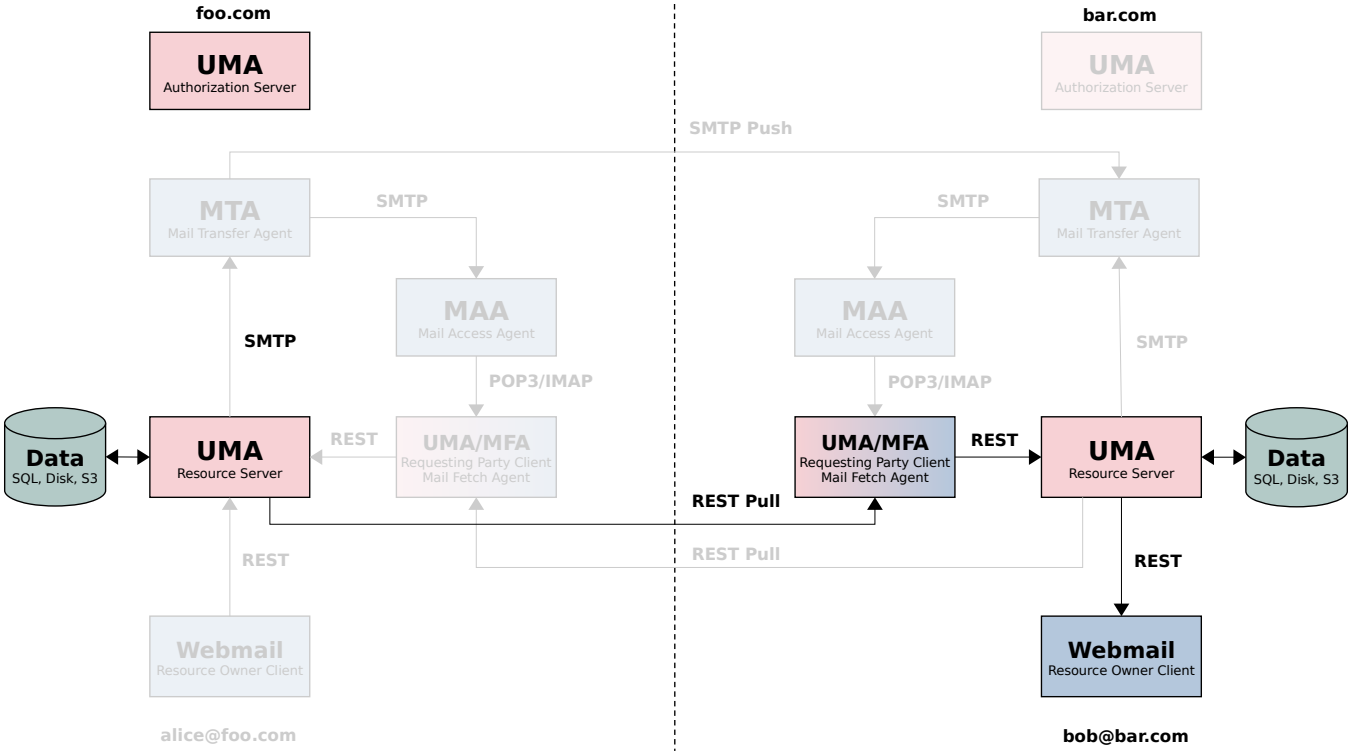

Figure 8. alice@foo.com to bob@bar.com get data

Features and Comparison with Current Email System

The novelty of the proposed solution approach can be assessed by comparison with the current email system.

New Features

The proposed AEES solution provides several new features that are lacking in current email systems:

- Intrinsic privacy-preserving properties. Each user can have their own separate RS as an email repository. The user can run its own RS, even its own AS.
- Built-in cross-domain autonomic (without conscious user intervention) access control using the standardized UMA framework.
- Autonomous (without interfering with the email system) data exchange channel.
- No attachments size limit. Attachments are transferred as separate files without size limit.
- Linked content using a clickable hyperlinks.
- Instant messages. Messages and attachments are transferred separately, there is no need to wait for incoming bulky message-with-attachments file. Attachments-stripped bare messages are transferred with a higher priority.

Comparison with Current Email System

From the users point of view, the use of the AEES has many advantages over the standard email system. In the following we highlight the advantages of the proposed solution compared to the current email system.

1. Security and Privacy:

The architecture of the AEES guaranties more control over potential security and privacy issues such as leakage of intellectual property or loss of confidential content and makes this system compatible with enterprise security policies.

2. Usability:

At the core of proposed solution is an attempt to improve the usability of email - not only as an interpersonal communication tool, but also as the default choice to send and store files. With the ability to store, locate, send and receive any content including documents, images, audios and videos the proposed solution can be used as a Content Services Platform.

3. Integrations:

The AEES provides a standardized Restful API interface to ease the integrations with external marketing, sales, Enterprise Content Management (ECM) or Customer Relationship Management (CRM) systems.

Conclusion

The AEES has been designed to follow the UMA framework best practices while keeping compatibility with current email systems.

Overall Summary

The Email System technology in combination with the UMA framework creates a composite architecture that meets the needs of the modern communication tool. This architecture increases robustness and performance of existing email ecosystem. The proposed solution can be used as a Content Services Platform to provide the storage repository protected by the standardized authorization framework utilized by users via the Webmail application.

A consolidated access control and a new data exchange mechanism leverages email security and enhances email system utilization. The question arrives whether a standard implementation of UMA can be integrated into the current email ecosystem.

Future Work

The combination of the UMA framework with email system outcomes in a new data exchange technology that predestine email system to become more than a bare messaging tool.

The following are potential future R&D areas:

- Explore the upcoming standardized UMA Relationship Manager (aka Wallet) Policy API vs. the proprietary Keycloak Policy API.
- Consider a Consent mechanism extension design.
- Explore linked content using a clickable hyperlinks – linking content across the business
- Design an extension for exchanging tagged messages and attachments – grouping content across the business
- Design an attachment versioning extension – the attachments with the same content are versioned
- Explore health information exchange between healthcare professionals and inspect use of email communication between patients and healthcare professionals.
- Employ regular email clients and applications using JMAP as a standardized email API.

A prototype implementation of the proposed solution, working as a proof of concept, would be interesting to build.

About the Author

Igor Zboran is a mechanical engineer by education with professional experience as a software engineer and solutions architect. He'd like to transform his knowledge into a useful system or service that people would love to use.

Igor received Ing. degree in Mechanical Engineering from the University of Žilina in 1988. After graduating, he worked in several small private companies as a software developer. From 2008 to 2009, he provided expert advice to Prague City Hall IT department as an external consultant. He invented a new decentralized Identity-Based Privacy (IBP) trusted model built around OAuth2 and OpenID Connect standards. Igor is a strong proponent of open source software and open standards.