Authorization-Enhanced Mail System

Igor Zboran
izboran@gmail.com
March 2, 2021

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 the current mail system. 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 mail system. The resource server is accessed generally by HTTP 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 mail system utilization and elevates the email ecosystem to the ubiquitous Content Services platform.

Introduction

The main components of the mail 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 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 mail 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 the mail server. From a user standpoint, it is easy to read and send emails via web-browser on any device, from anywhere in the world. Centralized access to the mailboxes, increases the security of web-based mail 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; bulky files are not transferred as an attachment but are shared via links. An “attachment sharing” is not natural for current mail systems where each message with attachments is expected to be time-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 Mail System is resistant to this security exploit as there are no direct user involvement in access granting.

Proposed Solution

Given that the mail 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 mail system. At the core of the 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.

Main Concept

The Authorization-Enhanced Mail System is designed to follow the Identity and Access Management (IAM) best practices while keeping compatibility with the current mail system. We propose to incorporate the UMA framework between the mail system with standardized SMTP/POP3/IMAP interface and the proprietary RESTful web-based email application as it is illustrated in Figure 1.

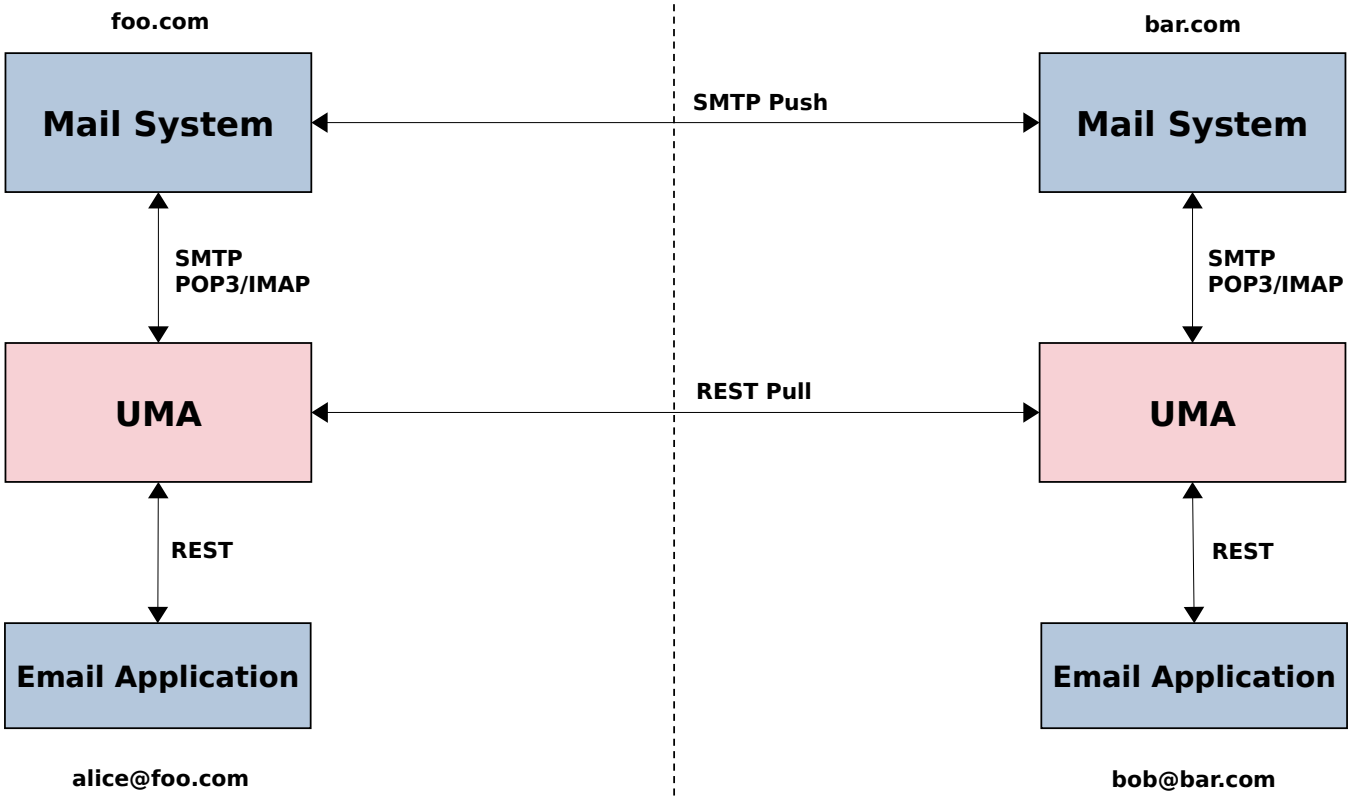

Figure 1. Main concept

Key points

1. An email consists of resources (message and attachments) stored in an UMA Mailbox – an email-specific UMA Resource Server.
2. The email resources owned by the sender stored in the sender's UMA Mailbox are temporarily shared with the recipient. Following a successful sharing process, links to the email resources are sent to the recipient through the authorization email – a type of system email.
3. The recipient's Mail Retrieval Agent that acts on behalf of the recipient retrieves the authorization email with the links to the email resources, authenticates against the sender's UMA Authorization Server, gets authorized access and downloads the email resources from the sender's UMA Mailbox. The downloaded data are stored in the recipient's UMA Mailbox.

The UMA framework plays its role during the data exchange process between mailboxes. The Email Application – with an UMA email extension – gives the sender a user-centric approach to manage and protect his/her ready-to-send email resources while the Mail Retrieval Agent is used to access and download temporarily shared sender's email resources as it is schematically illustrated in Figure 2.

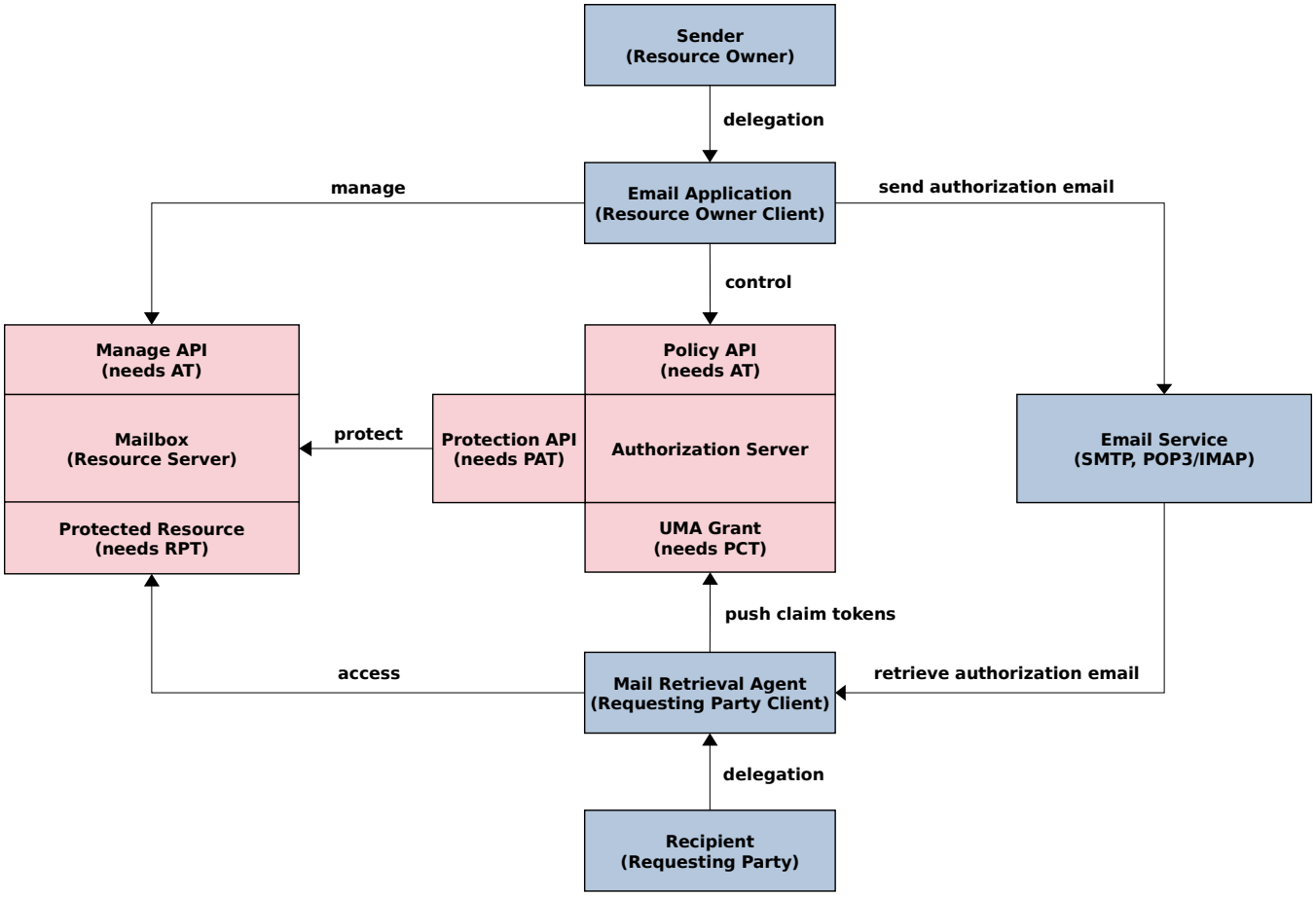

Figure 2. AEMS – schematic flow during the data transit from sender to recipient

During the data transit from sender to recipient, AEMS uses the existing email infrastructure to ensure authenticity of the authorization email, and the UMA protocol to control the recipient's access to the sender's email resources.

Figure 3 provides the sequence diagram for the AEMS/UMA Grant when the Mail Retrieval Agent client pushes a recipient's email address claim.

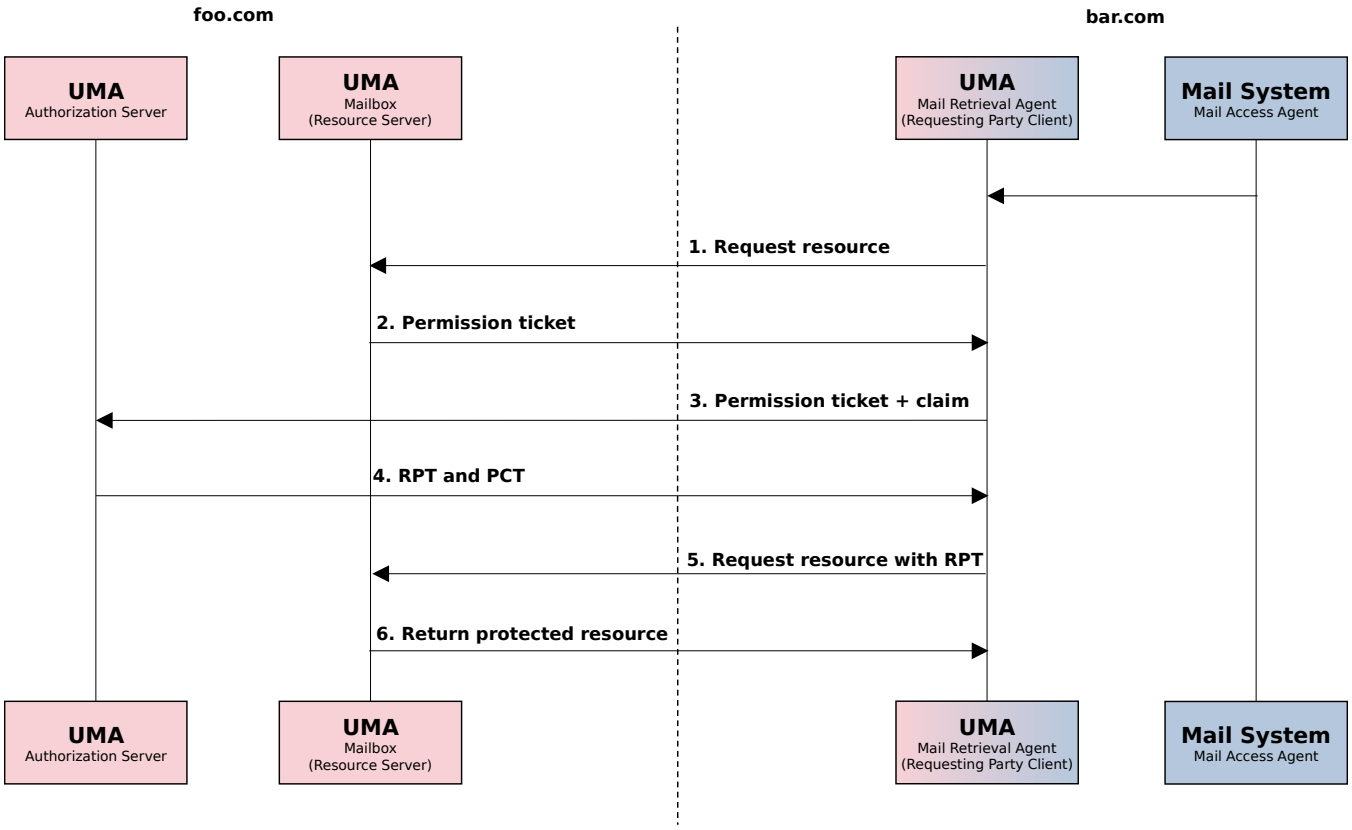

Figure 3. AEMS/UMA Grant sequence diagram (with pushed claim)

The following is a description of steps to get access to a protected resource:

1. The Requesting Party Client (aka Mail Retrieval Agent) is trying to access an UMA protected resource on the UMA Mailbox.
2. Without an access token, the Mailbox returns a permission ticket.
3. The client presents the permission ticket and claim along with client credentials at the UMA Authorization Server token endpoint (the claim value contains the recipient's email address).
4. The Authorization Server returns the access token (RPT) and PCT on successful evaluation of the policies.
5. The client again tries to obtain the resource.
6. The Mailbox returns the protected resource after validating the access token.

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

Here are a few more acronyms and short forms:

- AEMS (pronounced “aims”) Authorization-Enhanced Mail System
- MBX Mailbox
- MRA Mail Retrieval Agent
- MAA Mail Access Agent
- AT Access Token
- Email App Email Application

Trust Model

AEMS uses a decentralized one-way trust relationship model through two separate communication channels:

- To ensure authenticity of the registration and authorization email (see “Scenarios and Flows” section) AEMS relies on existing email authentication SPF, DKIM and DMARC techniques.
- The UMA 1.0 authorization framework – built around the OAuth-like protocol standard – was originally designed for Business-to-Consumer (B2C) scenarios. In UMA the roles of the AS, RS, RO and RqP client are co-located, they are all under the realm of a single trust domain. Fortunately during the development of UMA 2.0, the working group also considered a wide ecosystem where you can access a previously unknown UMA-protected RS. AEMS combines a decentralized email ecosystem with the UMA wide ecosystem to satisfy both the B2C and Business-to-Business (B2B) scenarios. There is no direct contract between authorization servers themselves and the UMA roles remain co-located.

Figure 4 illustrates the one-way trust relationship model established via the email infrastructure:

- Mail System Trust – email authentication.
- UMA Trust – an UMA wide ecosystem dynamically established trust (a kind of trust relationship)

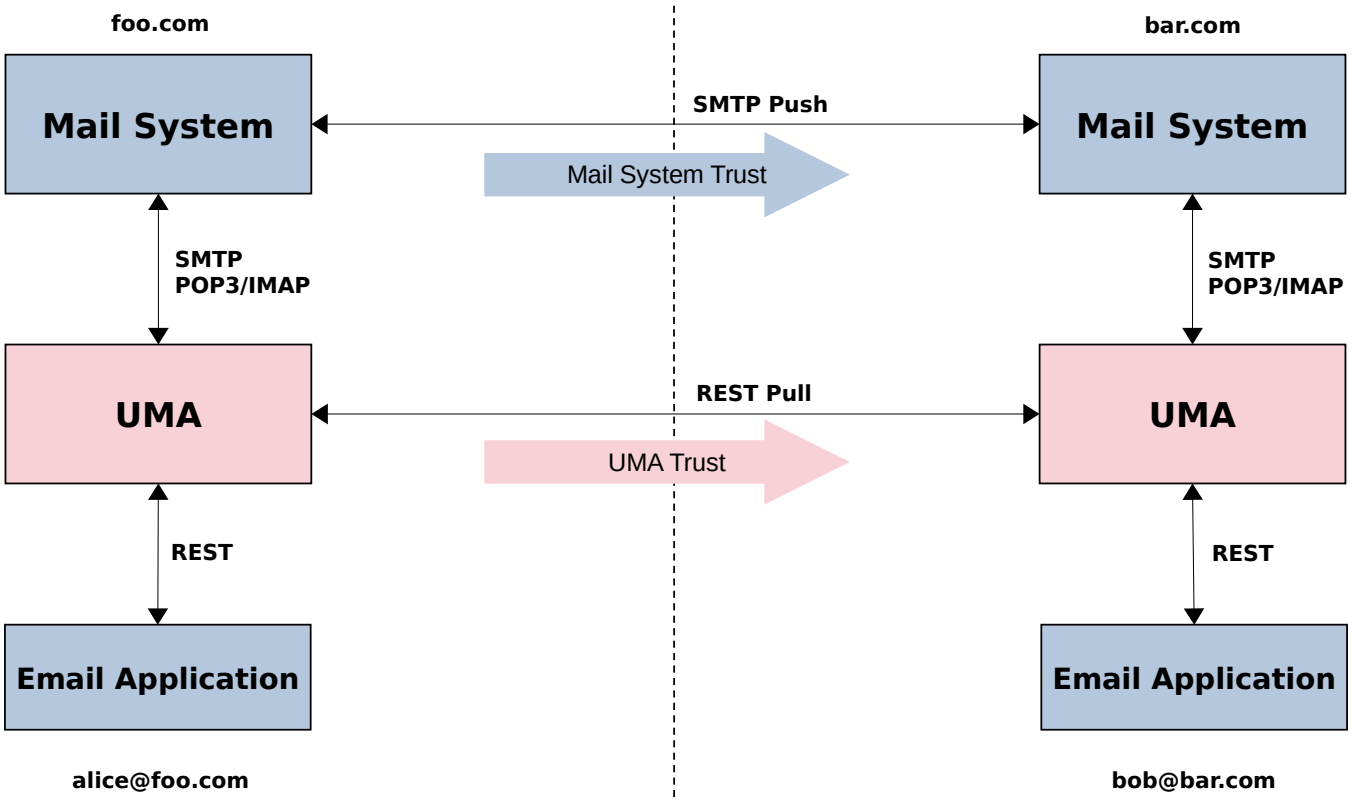

Figure 4. alice@foo.com to bob@bar.com trust model

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 trust relationship 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 relationship

Before the communication itself, a trust relationship governed by a contract must be established between the bar.com RqP client (aka MRA) and the foo.com AS. To set up a relationship, a registration message in an email (aka registration 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 UMA MBX and is typically bundled with the authorization email message sent from the Email Application.

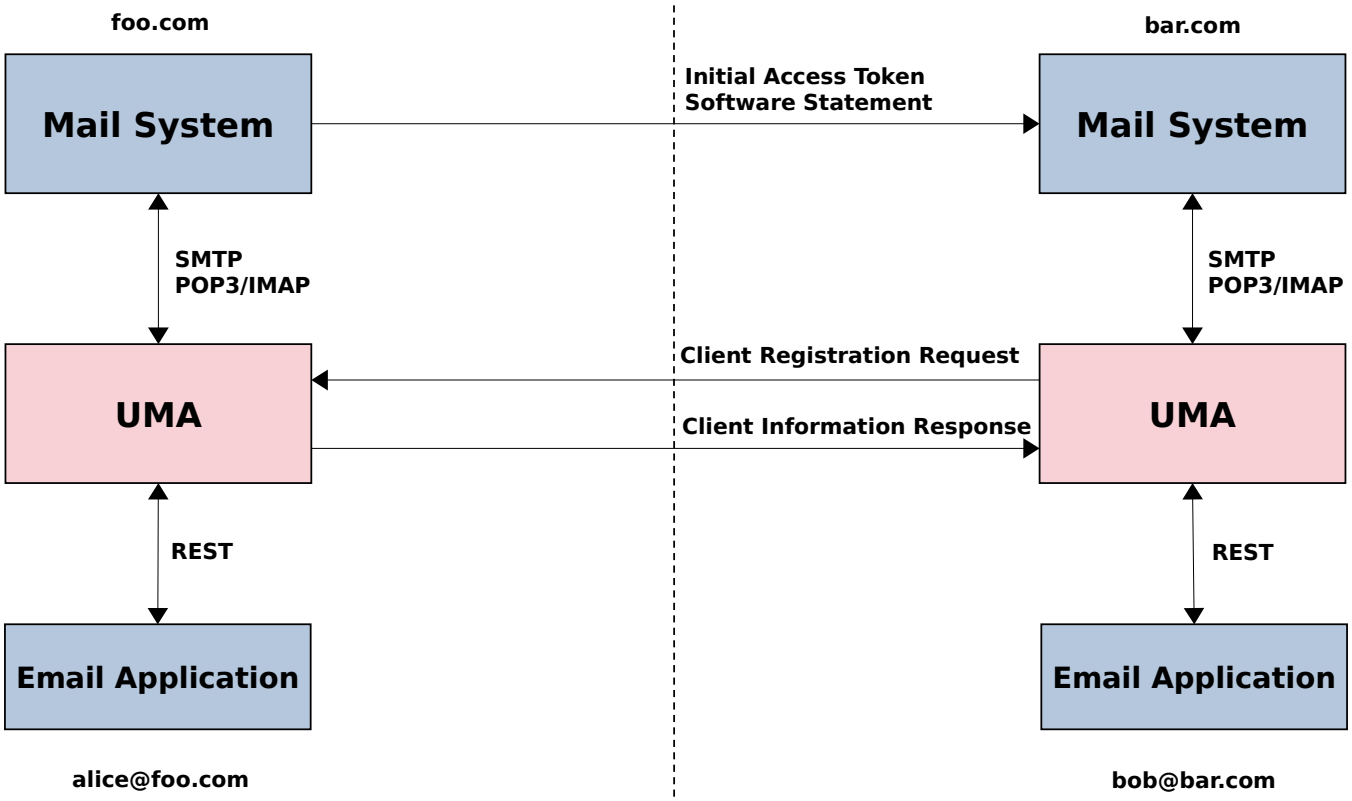

Figure 5. alice@foo.com to bob@bar.com dynamic client registration

After registration a trust relationship is set up between the bar.com RqP client and AS foo.com. The bar.com RqP client (aka MRA) now has access to email resources stored on the foo.com domain.

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

Let us assume the sender Alice – the resource owner (RO) – has prepared a draft email with an attachment. The draft message and the attachment are stored separately on the UMA protected MBX. The draft resources are not protected by UMA framework at this stage; data are considered inactive and are referred to as “data at rest”.

Before Alice presses the “Send” button, she fills in the “To” field with the recipient’s email address – bob@bar.com. This value will be used to set up an access to the email resources.

After Alice presses the “Send” button, the mutable draft resources become immutable email resources protected by UMA framework; at this stage data are considered active and are referred to as “data in transfer”. The email resources are registered at the AS resource registration endpoint. Next, a policy is created that gives Bob access to Alice’s email resources. Finally the authorization email with links to the email resources – in the AEMS authorization email format – is automatically created in the Email Application, and sent to the recipient Bob. The data flow is illustrated in Figure 6.

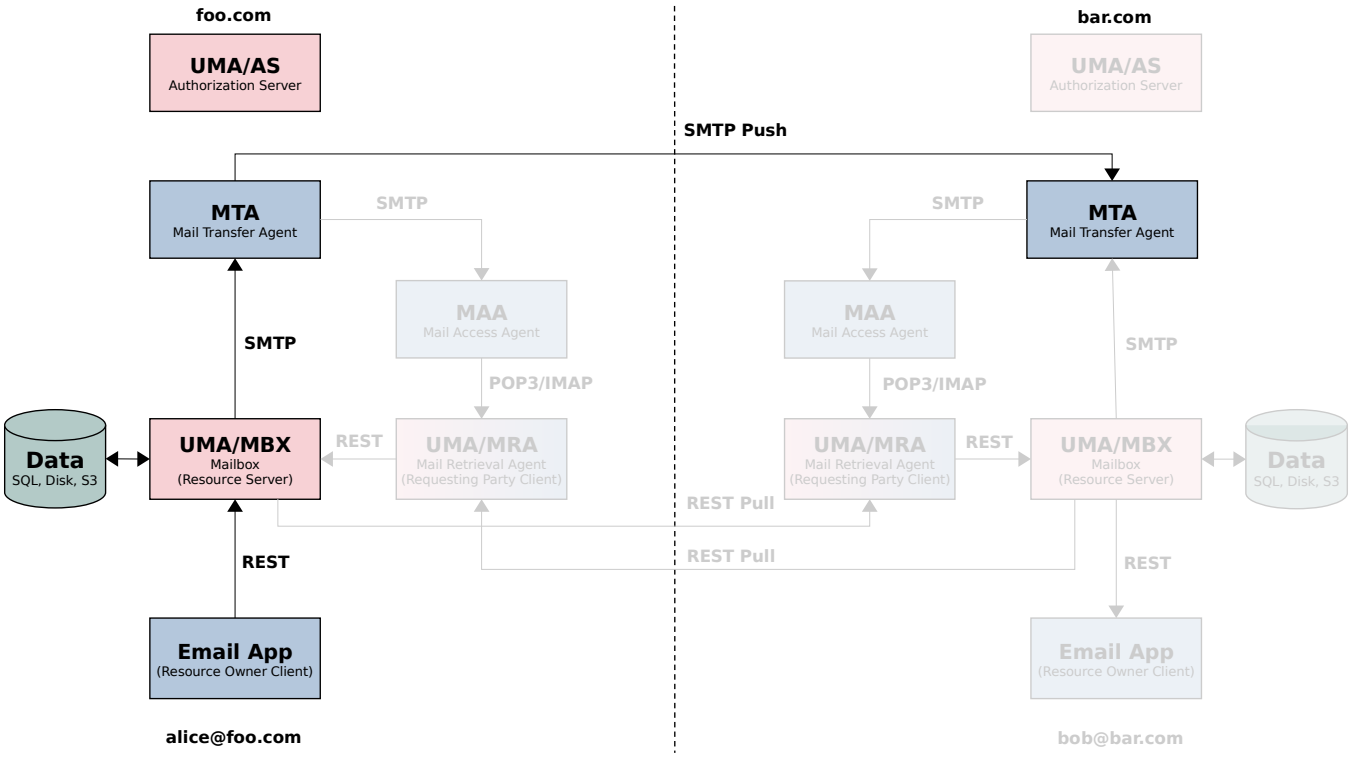

Figure 6. alice@foo.com to bob@bar.com – send resources links

Note: The specification of the AEMS authorization email format is outside the scope of this document.

Phase II – get authorization email with resources links, push claims, get authorization

An authorization email from Alice with links to her email resources stored on her MBX has just arrived at Bob's email provider. The mail access agent (MAA) notifies RqP client (aka MRA) – authenticated via IMAP/OAuth 2.0 protocol – of incoming authorization email. The RqP client retrieves the authorization email and checks its format. The data flow is illustrated in Figure 7. If the email is in the AEMS authorization email format, links to Alice's MBX are extracted and authorization process – using the AEMS/UMA Grant protocol – will proceed to get access to Alice's email resources. Bob's email address is used in the pushed claim value. The AEMS/UMA Grant sequence diagram is illustrated in Figure 3.

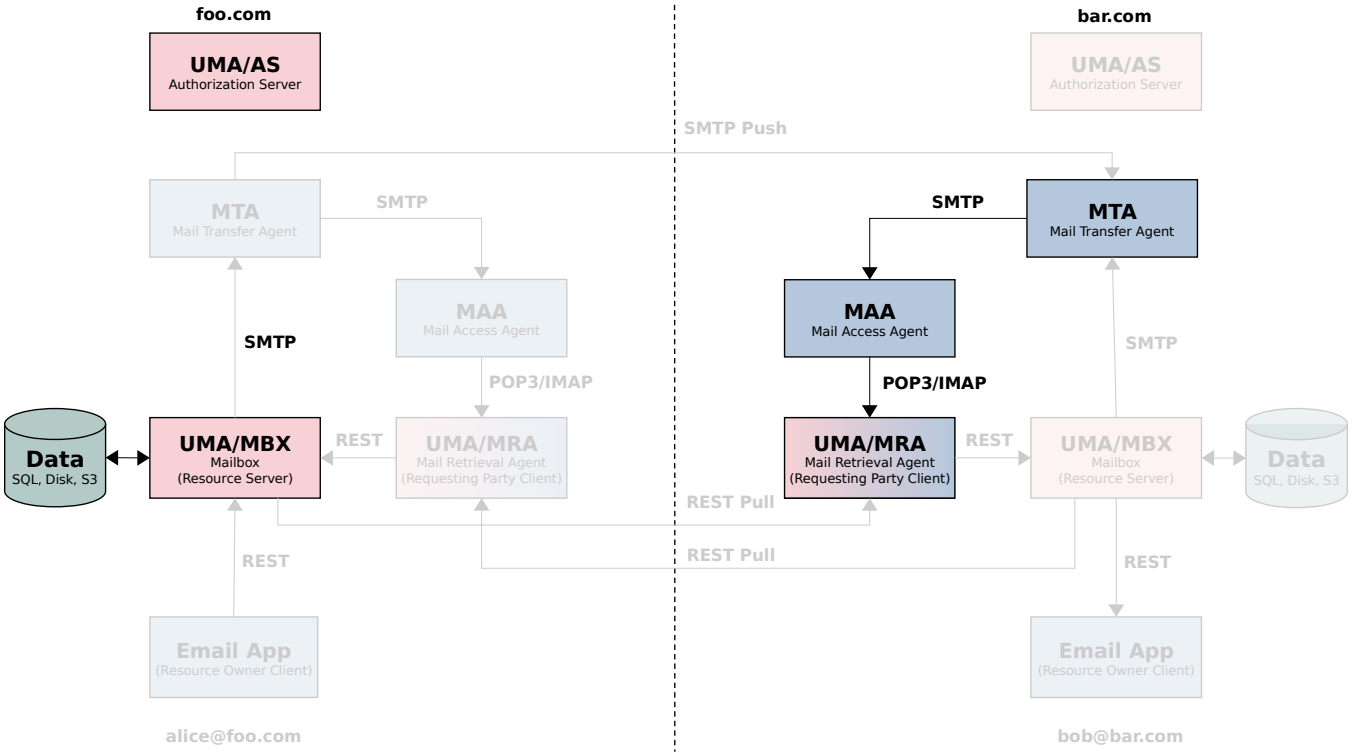

Figure 7. alice@foo.com to bob@bar.com – get authorization email

Phase III - get data, notify email application

After authorization using the AEMS/UMA Grant protocol flow, Alice's email resources are downloaded as it is illustrated in the last section of sequence diagram in Figure 3. The RqP client (aka MRA) must store the downloaded data on Bob's MBX as it is illustrated in Figure 8. Bob's Email Application should be notified of Alice's incoming email.

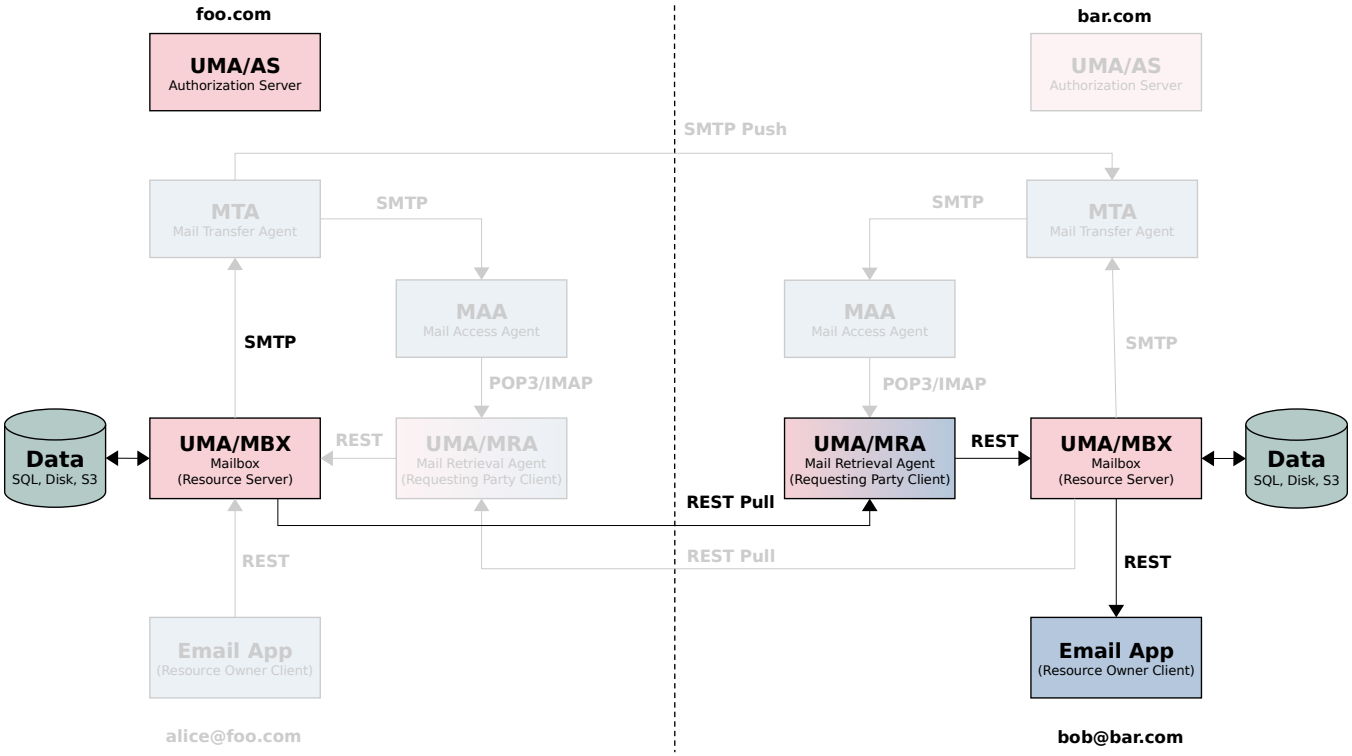

Figure 8. alice@foo.com to bob@bar.com – get data

Features and Comparison with Current Mail System

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

New Features

The proposed AEMS solution provides several new features that are lacking in the current mail system:

- Intrinsic privacy-preserving properties. Each user can have their own separate MBX as an email repository. The user can run his own MBX, even his own AS.
- Single MBX for both incoming and outgoing emails.
- Multiple MBXs per one email address.
- Built-in cross-domain autonomic (without conscious user intervention) access control using the standardized UMA framework.
- Autonomous (without interfering with the mail 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 Mail System

The use of AEMS has many advantages over the current mail system. The AEMS architecture increases the robustness and performance of the existing mail system. In the following we highlight the advantages of the proposed solution compared to the current mail system.

1. Security and Privacy

User correspondence takes place between UMA Mailboxes. The mailbox of the current email system becomes redundant and is only used for system (verification, authorization, ...) emails. This architecture guarantees more control over potential security and privacy issues such as leakage of intellectual property or loss of confidential content and makes the system compatible with enterprise security policies.

2. Usability

To separate official, business, personal and healthcare correspondence, AEMS provides the flexibility for storing emails according to various criteria with an appropriate AEMS provider. The ability of the UMA protocol to use multiple autonomous resource servers allows a user with a single email address to use simultaneously multiple UMA Mailboxes.

3. Platform

With the capability to store, locate, send and receive any content including documents, images, audios and videos, the proposed solution can be considered a promising platform for Content Services.

4. Integrations

AEMS provides a standardized Restful/GraphQL application interface to ease the integrations with external marketing, sales, Enterprise Content Management (ECM) or Customer Relationship Management (CRM) systems.

Conclusion

AEMS can play an important role in communication across various industries in the public and private sectors. Consolidation of repository, communication and identity represents a central source of information within any organization.

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. The proposed solution can be used as a Content Services platform to provide the e-records storage, exchange and retrieval system protected by the standardized authorization framework utilized by users through the email application.

A consolidated access control and a new data exchange mechanism leverages email security and enhances the mail system utilization. The question arises as to whether the standard implementation of UMA 2.0 will fit into the current mail system and how difficult it will be to build the UMA email extension.

Future Work

The UMA framework brings into the email ecosystem a new data storage and exchange technology that predestine the mail system to become more than a bare messaging tool.

The following are potential future R&D areas:

- In the authorization email, consider using an authorization code generated by AS instead of shared links to email resources.
- Explore the possibility of delivering the authorization code via SMS; use phone numbers instead of email addresses.
- 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 – managing the attachment content changes.
- 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 protocol to support a standardized email API.
- Design and use a proprietary GraphQL API to replace the poorly adopted JMAP protocol.
- Incorporate an electronic mailing/discussion list system into the proposed solution to extend the basic email functionality.
- Explore other ways of data origin authenticity (WebFinger, WebFist), replace SMTP with a web-based protocol.

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, Slovakia 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, Czech Republic 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.