Multi-factor authentication for mail clients

or

“How do we get rid of passwords?“

Ben Bucksch

About Ben Bucksch

- Thunderbird
  - 25 years core contributor
  - Thunderbird project leadership (was member of Council)
  - Speaking for myself, not for the project
- Consultant
- Created multiple companies

- IMAP, POP3, SMTP authentication infrastructure and logic in Thunderbird
- Autoconfiguration standard, invented ISP DB
- 4 different OAuth2 for mail client implementations:
  - Mail client for largest German mail service provider
  - Owl: Office365 client for Thunderbird
  - Reviewer of OAuth2 impl in Thunderbird
  - Parula

Requirements

Needs, from end user POV:

- Setup: only:
  - E-Mail address
  - Password or MFA
- After setup: Continuous mailcheck
  without interrupt: no breakdown, no re-login etc
- No evi1 h4x0rs

Autoconfig

- Works since 15 years, used by 10+ mail clients and lots of domains
- Well-known URLs; XML
- Configs for ~90% mail accounts automatic
- IETF Draft **adopted**, hopefully soon **RFC**
- ISPDB: https://ispdb.net (to be launched)
- https://autoconfigure.email (to be launched)

Autoconfig

https://autoconfig.*example.com*/mail/config-v1.1.xml?emailaddress=*fred@example.com*

Spec: https://www.bucksch.org/1/projects/thunderbird/autoconfiguration/

```
<clientConfig version="1.1">
<emailProvider id="office365.com">
 <domain>onmicrosoft.com</domain>
 <domain>mail.protection.outlook.com</domain><!-- MX e.g. example.mail.protection.outlook.com -->
 <displayName>Microsoft 365</displayName>
 <incomingServer type="imap">
  <hostname>outlook.office365.com</hostname>
  <port>993</port>
  <socketType>SSL</socketType>
  <authentication>OAuth2</authentication>
  <username>%EMAILADDRESS%</username>
 </incomingServer>
 <incomingServer type="exchange">...</incomingServer>
 <outgoingServer type="smtp">...</outgoingServer>
</emailProvider>
<calendar>...</calendar><contacts>...</contacts><fileShare>...</fileShare>
<oAuth2>
 <issuer>login.microsoftonline.com</issuer>
 <authURL>https://login.microsoftonline.com/common/oauth2/v2.0/authorize</authURL>
 <tokenURL>https://login.microsoftonline.com/common/oauth2/v2.0/token</tokenURL>
 <scope>IMAP.AccessAsUser.All POP.AccessAsUser.All offline_access</scope>
</oAuth2>
</clientConfig>
```

PACC

- Autoconfig, but as IETF customs
- Uses DNS SRV
- JSON
- Assumes some „best practices“, like

  IMAP username = email address
- → Requires mail provider to adapt config to it
- → Cannot reflect many existing configs
- Internet Draft, to be discussed

Authentication: MFA

- Passwords must die
- OAuth2 has multiple problems for mail clients
- Passkey

OAuth2: Main problem points

- Configuration
  - URLs, Scope etc.
- Client registration
- Expiry
- OAuth2 loose: Framework, not protocol
- Goal: Fixing them

Configuration

- OAuth2 doesn‘t spec how to get the config
- → All URLs hardcoded
- → Only Google, Microsoft, Yahoo, Apple, ...
- → Small ISPs, Self-hosting, Privacy - impossible
- Solutions:
  - OpenID Connect with .well-known/
  - RFC .well-known/
  - Autoconfig

RFC 6749 3.1 „The means through which the client obtains the location of the authorization endpoint are beyond the scope of this specification."

Client registration

- C mail clients, P mail providers → C * P
- RFC 2971 (IMAP ID):
  “Servers **MUST NOT deny access** to or refuse service for a client based on … the ID command.“ – Question to audience: Why? (Open-Source...
- RFC 6749 Section 2.2 (OAuth2):
  - Client ID is **required**
  - Registration is at the **mercy** of the ISP
- Anti-competitive: ISPs make registration hard or practically impossible.
- Directly against the idea of Open-Source

Web browser

- Full interactive web browser
- → Mail without web impossible
- → Many non-UI clients (Alexa, server, console, car etc.) locked out
- Security: Mail in car → OAuth2 → Browser → Monthly security updates for 20 years? → Security holes in the car → Liability
- Complexity: Even Win/Android: Browser security updates → Cannot ship browser → Need system browser → Dependent on system APIs
- Need to watch URL changes to know when done and get auth code
- Solution suggestion/idea: Not HTML, but challenge/response:
  - Server asks question in plain text, client sends response in plain text.
  - Waiting on server, with message.

Unreliable

- Does not spec actual login
  - AuthCode could be sufficient for login to IMAP
  - OAuth2 (currently) doesn‘t solve any problem that the mail client has
- Lacks hard guarantees
- Users always blame the UI/client
  - → Mail client needs reliability

Expiry

- refresh_token is optional; expiry time is optional
- access_token -> refresh_token
  - → Client needs to cache and repeat any and all server calls.
    E.g. Copying 1000 emails → Coffee → Login expired in the middle
- refresh_token -> interactive login
  - → Client needs to jump from **library** code to interactive, in any function
  - → Makes library APIs and app code complicated, e.g. **all** library functions need a login callback
- How to do mail check reliably?
  - User waiting for specific mail, but mail check expired.
    - Cannot ring phone, just for login. User sleeping (at other time).
    - „I‘m sorry, Dave, I‘m afraid I cannot check mail anymore. I hope it wasn‘t anything important.“
  - Big difference between expiry 5 min, 12 h, 6 months
- Username+password don‘t have such problems

Error handling

- Need to watch URL changes to know when done or failed
- Error message only in English (US-ASCII) → No error message for end user
- Browser window closed. Stuck without knowing why: Crash? Error message in HTML? User changed mind?
- OAuth2 error codes are unhelpful: "access_denied"
  - Password wrong → Try again
  - Rate limited, Account suspended → Do not try again
  - User changed mind → Close dialog
- Mail client is completely blind. No idea how to continue.
- Solution: Specify detailed error codes, with error classes, and an error message for end user.

Password vs. OAuth2

- **That's** why email clients still use username and password. It's simple and predictable. I know the input and the output. It's clearly defined. I have no config issues. That's the whole reason.
- If we want 2FA for email clients, we need to nail it down, so that it's reliable and not dependent on the implementation at the service provider.
- → Mauth
- → Autoconfig

OAuth Profile for Open Public Clients

- Internet Draft. Adopted by IETF WG
  https://www.ietf.org/archive/id/draft-ietf-mailmaint-oauth-public-00.html
- Defines exact client flow
- Config .well-known, similar OpenIDConnect
- Requires Dynamic Client Registration RFC7591
- Defining OAuth2 scopes
- Expiry: „should not“ expire

MAuth

- Similar to Open Public Clients, alternative
- But hardcoded (= disabled) client ID „open“
- No expiry
- Detailed error codes
- Scopes defined for IMAP read, IMAP write, POP3, SMTP etc.

Passkey

- SASL standard
- Vendor lock-in: Need free implementations

SASL Passkey

- https://benbucksch.github.io/sasl-passkey/draft-bucksch-sasl-passkey.html
  https://github.com/mustang-im/mustang/wiki/Auth-Passkey
- Create Passkey on ISP website, and stored in OS passkey manager
- Mail app uses OS passkey APIs (1 C function, like web credentials.get())
- Flow
  1) Server sends challenge
  2) mail app passes challenge 1:1 to OS passkey APIs (1 C functions)
  3) OS does auth (fingerprint, face, device PIN)
  4) OS signs challenge with private key of passkey
  5) Mail app returns 1:1 to server
  6) Server validates response with public key of passkey
- Retain login with SASL Rememberme (JWT, refresh token, app password, ...)

Need software: APIs and libs

- Windows, macOS, iOS, Android
- But bound to vendor cloud: Lock in!
- Nothing on **Linux**
- Define **API** between app and passkey manager
- Implement DLL/lib to switch between managers
- Implement a passkey manager and device sync
  - Bitwarden, KeePass etc.
- **Need your help with that!**