Interactive Authentication of Non-Interactive HTTP Requests

Ben Schwartz, HTTPAPI and OAUTH @ 115
Popup Authentication
This is what login looks like on the web today...
... and this is how it looks for the rest of HTTP
Non-web HTTP login is stuck in 1996
What about OAuth?

OAuth (currently) enables clients to speak proprietary protocols (over HTTP) to specific origins that are known in advance.

This protocol is for clients that want to speak standardized protocols (over HTTP) to any compatible origin.
Your Example service has requested interactive authentication.

OPEN BROWSER

CHANGE EXAMPLE PROVIDER

Interactive authentication complete

OK
HTTP Exchange 1: The trigger

OPTIONS /home/bemasc/calendars HTTP/1.1
Host: cal.example.com

HTTP/1.1 401 Unauthorized
WWW-Authenticate: interactive location=/login
WWW-Authenticate: ...

Hey, do you support CalDAV?

Who are you? Open a browser.
HTTP Exchange 2: The login screen

GET /login HTTP/1.1
Host: cal.example.com
Accept: text/html,...
Accept-Language: en-US,...
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
...

HTTP/1.1 401 Unauthorized
Content-Type: text/html
...

Hi cal.example.com, this is the web browser.

The user has opened https://cal.example.com/login in a new tab.

Here are the login instructions.
HTTP Exchange 3: The success signal

GET /login HTTP/1.1
Host: cal.example.com
Accept: text/html,...
Accept-Language: en-US,...
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Cookie: login=6bb0e2c8-874e-44c8-b8e0-25e12f339b46
...

HTTP/1.1 200 OK
Content-Type: text/html
...

The user followed a link to https://cal.example.com/login, and I already have a cookie for this request.

OK, you can close now.
HTTP Exchange 4: The access

OPTIONS /home/bemasc/calendars HTTP/1.1
Host: cal.example.com
Cookie: login=6bb0e2c8-874e-44c8-b8e0-25e12f339b46

HTTP/1.1 200 OK
Allow: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, COPY, MOVE
Allow: PROPFIND, PROPPATCH, LOCK, UNLOCK, REPORT, ACL
DAV: 1, 2, access-control, calendar-access...
Specified procedure

1. New auth-scheme “interactive” with a “location=” parameter that provides the authentication path.
2. The client reacts by opening this path in a browser “popup”.
3. The client interacts, navigates, types passwords, accesses second factors, etc.
4. If the authentication path ever loads successfully, the client stores the request headers and closes the popup.
5. The client copies any stored Cookie or Authorization headers into its future requests for this origin.
Interesting corners of this spec

- Both Cookie and Authorization headers are supported.
  - “Authorization” is more natural, but only “Cookie” can be used without Javascript.

  - …but Cookie headers are just dropped
  - Should we define a way to send cookies to a proxy?

- The spec mandates a URL bar (to avoid phishing) and interstitial dialogs before the browser opens and after it closes (to avoid clickjacking).
  - Is there a better way?

- “interactive” can be used alongside “basic” or “digest” for compatibility
  - Browsers are required to ignore “WWW-Authenticate: interactive”

- No way to declare success without closing the browser…
Closing thoughts

- Brand new draft!
- Brings all the goodness of modern web login to the rest of HTTP
- Needs more HTTP and OAuth expert input
  - How should Set-Cookie parameters work?
  - Should we define a way to send cookies to HTTP proxies?
  - Is there a way to share more concepts with OAuth?
- Seeking adoption in HTTPAPI/HTTPBIS/OAUTH/???
- Mentioned in draft-schwartz-masque-access-descriptions as a good way to authenticate to proxies.