HTTP Mutual authentication protocol proposal

Yutaka OIWA

RCIS, AIST

Problem

- Current HTTP auth is weak
  - In security:
    - Basic: plain-text authentication
    - Digest: off-line attack, not well implemented
    - TLS Client cert:
      too complex for most users, side
  - In functionality:
    - No log-off
    - Modal dialog for authentication
    - Authentication “enforced”
      - No good support for guest users

Problem

- In reality, form-based auth is widely-used
  - Having many problems
    - Plain-text only
    - Very weak against phising attacks
    - Prone to implementation bugs

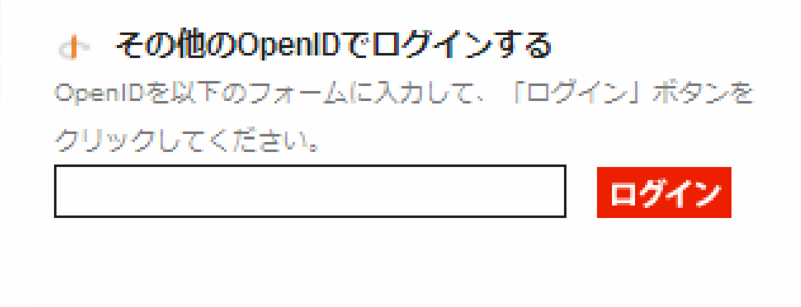
- To solve, a “better” HTTP auth is required.

Problem

- In reality, form-based auth is widely-used
  - Having many problems
    - Plain-text only
    - Very weak against phishing attacks
    - Prone to implementation bugs

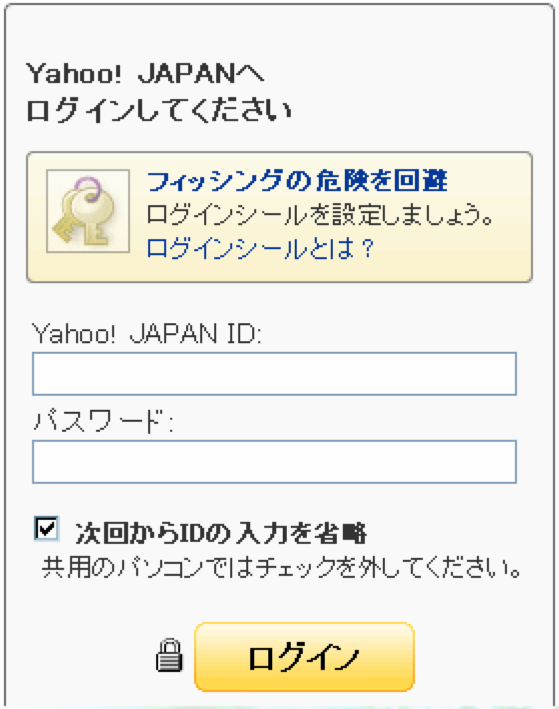
Problem with federated Auth/authz

- Users have to input passwords in a redirect page
  - How we can make sure it is not a phishing page?



その他のOpenIDでログインする

OpenIDを以下のフォームに入力して、「ログイン」ボタンをクリックしてください。



Yahoo! JAPANへ
ログインしてください

フィッシングの危険を回避
ログインシールを設定しましょう。
ログインシールとは？

Yahoo! JAPAN ID:

パスワード:

☑ 次回からIDの入力を省略
共用のパソコンではチェックを外してください。

Can you carefully check identity of this form every time without mistake?

Goal

- A better authentication which will enable
  - Password-based authentication
  - Strong protection of password,
    even if it is either eavesdropped or phished
    - Note: hash is not enough strong against password-crack on recent computers
  - Prevent that *phishing site to make authentication succeed, or even pretend it succeeded*
  - Works well with recent web applications design
- *Mid-/Long-term solution: very secure, but requires both client/server implementation changes*

HTTP “Mutual” auth.

- New access authentication method for HTTP
  - Secure (⇔ HTTP Basic/Digest, HTML Form)
    - No offline password dictionary attack possible from received/eavesdropped traffic
  - Easy to use (⇔ TLS client certificates)
  - Provides *Mutual authentication*:
    clients can check server’s validity
    - Authentication will ONLY succeed with servers possessing valid authentication secrets
    - Rogue (phishing) servers can’t make authentication to succeed

Basic design

- Implemented on top of RFC2617
  - Standard WWW-auth/Auth-info headers used
- Password-based Mutual authentication
  - Using PAKE as underlying crypto primitive
- Authentication only
  - Can be used both with HTTP and HTTPS
  - Encryption/integrity provided by HTTPS
- No long-term storage required

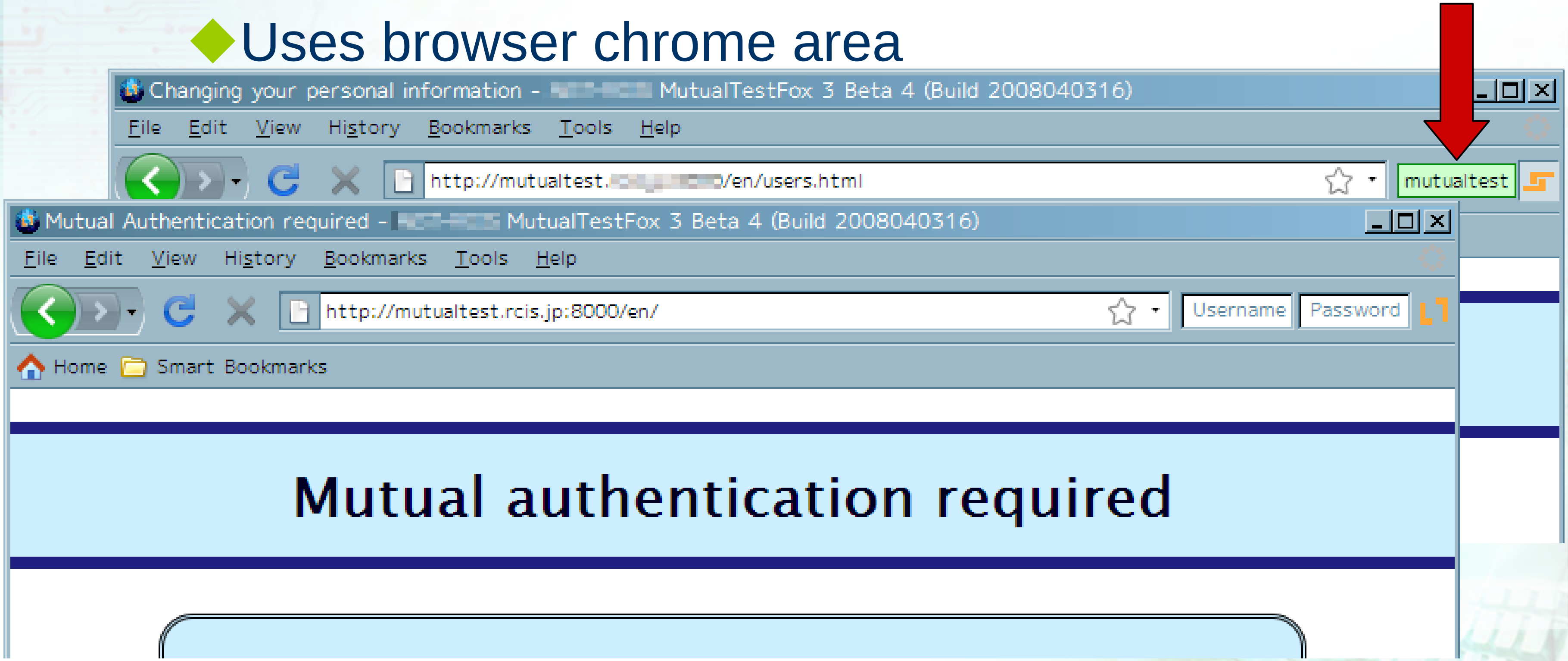
  (⇔ Client Certificate, pwd-mgr + auto-gen etc.)

More features

- Support for recent Web application design
  - To solve several current issues with HTTP auth: covers reasons to use Form-based auth.
- Optional authentication
  - Single URI can serve both auth/unauth contents
  - Support for sites like Slashdot, Google or Yahoo
- Timed/server-initiated logout
- log-on/log-off page redirection
  - More to come?

UI consideration

- Trusted display for mutual authentication result will be needed
  - We propose new UI for this auth scheme
    - Uses browser chrome area



Changing your personal information - [redacted] MutualTestFox 3 Beta 4 (Build 2008040316)

http://mutualtest.[redacted]/en/users.html

mutualtest

Mutual Authentication required - [redacted] MutualTestFox 3 Beta 4 (Build 2008040316)

http://mutualtest.rcis.jp:8000/en/

Username

Password

Mutual authentication required

Demo

- If we have a time and a working Internet connection…

Current status

- Spec draft: draft-oiwa-http-mutualauth-06
  - Submitted as an Internet-Draft
- Draft Implementations
  - Server-side: Apache, Ruby webrick
  - Client-side:
    - Mozilla-based implementation (Open-source)
    - Pure-Ruby reference implementation (to appear)
    - IE-based implementation (closed-source)
  - Available from project homepage: https://www.rcis.aist.go.jp/special/MutualAuth/
    - Trial website there!

Future work

- Standardization and Impl. integration
- Integration with application frameworks
  - E.g. Rails, PEAR etc.
- With higher-level auth/authz schemes:
  - OAuth (federated authz delegation):
    - should work well
  - OpenID, SAML or (other federated auth):
    - Will work as a primitive
    - For better user-feeling, integration may be needed
    - Experts needed ☺

Thank you

- More resources
  - Our project homepage: https://www.rcis.aist.go.jp/special/MutualAuth/
  - Draft:
    - Official: https://datatracker.ietf.org/drafts/draft-oiwa-http-mutualauth/
    - Some preliminary drafts (before submission) may be on our homepage