http-auth side meeting

Yutaka OIWA
Wednesday 30 March 2011
IETF 80 Prague
Agenda

- How we work with http authentication?
- A short introduction of my proposal
- Discussions
Web authentication

Many peoples might agree it is broken

How?

- People continues to use Form/plaintext auth.
- People does not use HTTP authentication although there is it
- We failed to solve any kind of existing problems
  - Phishing...
  - Hardness of using any cryptography...
Things what we have now

- HTTP authentication schemes
  - Basic
  - Digest… more or less died
  - NTLM, Negotiate, … limited usage
- TLS
  - Client authentication … very limited usage
- Form authentication
  - Very widely used
  - Causes LOTS of problems
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?

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

- HTTP etc. has provided no *usable* solutions
  - Recent Web application evolved to provide lots of security-related application features
  - Most of these hard to be implemented on HTTP/TLS authentication

→ people has difficulty/distaste of using HTTP authentication, prefers Form-based auth
True cause of problem

(Incomplete) list of modern features implemented by using application-level auth

• Complex timeout management of log-in status
• Forced/user-originated log-out
• Persistent log-in
• Site-wide single-sign-on
• Federated log-in
• Multiple authentication realms (user-name spaces)
Application-level auth... drawbacks

- No protection of passwords to the server
  - Form and server-provided HTML have full control of what is inputted
  - Plaintext always available (often sent) to the server (on TLS, though)
- No cryptographic protection against fraudulent servers
  - So-called “Phishing”, many variations
**Chicken and Egg problem**

✦ “Improving HTTP-auth is boring, if people does not use those instead of Form auth.”

✧ Or, “Why they do not use this incredibly-secure solution existing now?”
  - *it often does not meet application/business requirements*

✦ “If there is only HTTP-Basic useful, no one have good reasons to throw Form auth. away.”
So what we need?

- We need to cut the Gordian knots
  - We must provide enough-Secure mechanisms to address existing security problems
  - We must, at the same time, provide enough useful mechanisms so that people can move to the new things
Possible authentication means

- **Passwords**
  - Most simple, easy-to-understand credential
    → HTTP Mutual authentication proposal

- **Certificates, keys in smart cards**
- **Two-factor authentications (e.g. HW token)**
- **Federated Authentications**
- **Use existing backend (SASL, Kerberos etc.)**
What “I” want to talk about today

- Discussion on the “Problem space”
  - What we should solve from this year
  - What we are required to solve
  - What we can use now
- Discussion on the time scope
  - Possible future timeline/schedule?
- “Cloud/association” of people interested
  - We need friends to work with
A (relatively) short description of HTTP Mutual authentication

Yutaka OIWA
RCIS, AIST
IETF 80
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.)
To overcome “usability” problem

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 be needed?
    – I need a feedback for that, too
Draft organization

As of draft-08:

1.: Introduction
2.-9.: Core part
   • message syntax, state machines, session caching
   • Single-sign-on treated in 5.
10.: Authentication-Control header
   • Extensions to make it usable with Web apps.
   • “application” peoples comments needed
11.: Authentication Algorithms
   • All boring mathematics 😊
   • “security” people’s comments needed
12.-16.: all finish-ups
   • IANA, security consideration, references etc.
UI consideration

trusted display for mutual authentication result will be needed

- We propose new UI for this auth scheme
  - Uses browser chrome area
  - Not a part of the draft, however
Current status

- Spec draft: draft-oiwa-http-mutualauth-08
- 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!
Thank you

More resources

Our project homepage:
https://www.rcis.aist.go.jp/special/MutualAuth/

Draft:
- Some preliminary drafts (before submission) may be on our homepage