HTTP Mutual authentication and Web security

Yutaka OIWA
SAAG, IETF 80 Prague
Web security

- Its importance... no need to say
  - Transaction security (credit card, PayPal etc.)
  - User data privacy

- Most online consumer and business commerce transactions rely on Web
HTTP (Web) is very tricky!

Web/HTTP auth is tricky… why?
- It has completely different design from other protocols with authentications
- It has very different nature of security implications, even using the same technology

Let me compare this with other conventional protocols, such as IMAP/TLS
Authentication in usual protocols (mail client scenario)

- A pre-configured, single server to be connected (per an account)

- A single (or a few) authenticated connection are established, and used for several requests sequentially

```
A single connection

Channel setup ───► Authentication Security setup ───► Data Processing
```

Web client scenario

- Target host dynamically determined from URL host-part
  - It is often provided externally
    - from URL link in an email
    - from external Web site for federation
      (e.g. PayPal checkout, OpenID, OAuth etc.)

- No preconfigured authentication
  - When the server demands, the browser/webpage requires user credentials
Web and authentication

HTTP is a (kind of) packed-based protocol
- Each single request is independent from others
- Even requests on the same channel are independent from each other
  - Requests on “single session” can be sent on several different TCP/TLS connections
  - Requests on “different sessions” can be sent on the same connection
Web and authentication

HTTP is a (kind of) packed-based protocol

Access web mail A
... auth required

Send a credential
... access granted

Want to write a mail
... “new mail” form sent to client

Post a mail form
... server process the req.

Connection 1

Connection 2
Web and authentication

...even an interleaving is possible

- Access web mail A
  - auth required
- Send a credential A
  - access granted
- Want to write a mail
  - "new mail"
- Post a mail form
  - server process the req.
- Read a news B
  - auth req.
- Send a credential B
  - access granted
Web and authentication

Interleaving example: Gmail (Google)

Sessions of two different accounts
(one for gmail.com domain, one for my own domain)
runs concurrently on the same browser
Functionality layering

- Conventional protocols on TCP/TLS (e.g. IMAP)
- Web/https (per 1 host)

Diagram:
- Requests flow through layers:
  - Authentication
  - Security
  - Transport
- Transport = application session

Diagram:
- Authentication
- Application session
- requests flow through layers:
  - Authentication
  - Security
  - transport
  - Application session
Web authentication methods

HTTP auth (RFC 2617) has two problems

1: not very strong
- Basic = plaintext on wire
- Digest = just a salted MD5…dictionary attack possible

2: not used
- Bad UI design
  - Who wants to see that popup dialog?
- Lack of required flexibility to implement web apps
  - Log out, session timeout
  - Support for guest users
Web authentication methods

Cookies are widely (ab)used

- password checking implemented in application level using HTML and Web forms
- Application level session ID is issued as a “cookie”
- All authorizations and authentication status controls (such as logout, timeout etc.) are also implemented in application level

Cookie: SID=UxVwgVTWxGVZDeGe13PeOBK...
Web authentication methods

Cookie-based authentication

Problems

- Plaintext passwords always available to Web server
- Very weak against Phishing attacks
- Often misimplemented to cause security issues
Phishing

A social attack on the Web
- attacker leads victims to a wrong site with a similar looking to the genuine site
- Steal a username and a password

Why happens (only) on Web?
- Key: how the server authentication really works
TLS server authentication

- The client has an “intended host”
- The server sends a certificate to the client
  - With hostname in CN or altDomainName field
- The client checks whether the two hostname matches

This works for the mail client scenario, as “intended host” is fixed. However…
TLS server authentication

In the Web browser scenario:

- The “intended host” is a part of the given URL
  - Dynamically determined by the browser
  - “The given URL” sometimes comes externally

So, what happens?

- If the URL https://www.yahoooo.co.jp/ is given, the browser will accept a connection with someone other than Yahoo! Japan™
Current Phishing countermeasures

- Phishing site blacklists
  - Hard to maintain
  - Impossible to be perfect

- EV SSL certificates
  - Sacrifices Web openness in trade with financial security requirements
  - Users still need to check the EV status bar display
Problem statement

We have to fix the Web authentication by technology which is

- Enough secure!
  - Addresses many current issues on the authentication

And, implementable, deployable and usable!
  - Web people reject all ideas which decrease services’ flexibility and users’ experiences
  - Not just scalability or security
Use cases to be targeted

Usual “Web applications”
- Small ones such as Wiki, Trac etc.
- Large ones such as Google, Yahoo, etc.
- Application-specific auth scope designs
- Needs flexibility, depending on pages

“Intranet-type” sites
- Users always authenticated to used
- All pages authenticated with same credential
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
  - Phishers can’t make authentication to succeed
HTTP Mutual authentication proposal

Some design decisions

- Use HTTP-level authentications
  - Works well with HTTP architecture and existing Web application designs
- Use with TLS encryption/server authentication
  - Already working quite well, minus Phishing
- Strong protection of user identity
  - No information leakage on eavesdropping
  - Mutual authentication to detect Phishing attacks
HTTP Mutual authentication proposal

Technologies introduced:

- PAKE-based authentication
  - Based on ISO 11770-4 KAM3
  - Enables strong authentication, relying only on passwords
  - Both EC and DL supported … if people want

- Channel binding with both HTTP and TLS
  - To prevent any forwarding-type Phishing trials
  - TLS required to prevent transport-level MITM attacks

- Auth architecture extensions to regain usability
Sample implementations

- Modified Firefox 3.6
- Apache extension modules
- Reference implementations on Ruby

Existing on our project Web page

... but currently down 😞, due to the earthquake and related blackouts

Will be on-line as soon as possible
Other possibilities: SASL

SASL (RFC 4422)

- Provides unified methods for user authentication on many applications
  - Single user database and library can be used for many applications
  - Single library can support several authentication methods from PLAIN to CRAM, NTLM or Kerberos

How about use for Web?
Other possibilities: SASL

SASL (RFC 4422)

- May be used well for “intranet”-type application
  - But not easy for usual user Web applications
- How about use for Web as a general?
  - My answer is “not simple”
- Auth-method flexibility leads to “security downgrade attack”
  - In mail clients, there must be a “no plaintext authentication” preference checkbox (& it’s enough)
  - But how we do it for Web browser?
Other possibilities

- TLS user authentications
  - Passwords: TLS-EAP, TLS-PSK, TLS-SRP etc.
  - Client certificates

At first glance, it seems to be a good idea for doing auth. in transport level, but...

- Please remind the protocol architecture once
Other possibilities

HTTPS user authentication for Web?

Layering problem exists
- Impossible/hard to match authentication sessions (required by application design) with transport sessions

UI issues
- Even worse than current HTTP authentications
  - Authentication must be done before the URL is known to the server

Still works for intranet-type applications
- OK if only single, whole-the-server authentication done

We may need a way to use certificates wisely…
Thanks!

Some resources:

- Draft: draft-oiwa-http-mutualauth
- My homepage: https://staff.aist.go.jp/y.oiwa/index-en.html
  - online, have a link to the page below
- Project homepage: https://www.rcis.aist.go.jp/special/MutualAuth/
  - Implementations and more resources exists
  - Currently down, will be up after I go back to Tokyo 😊