Getting web authentication right

Joseph Bonneau

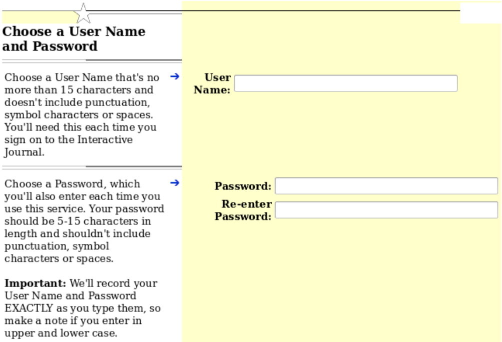
jcb82@cl.cam.ac.uk

Security Protocols Workshop
March 28, 2011

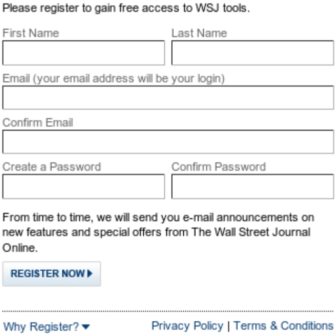
A parable of obsolescent technology

Credit: freeyellow.com

Web authentication has evolved very little...



Wall Street Journal, 1996



Wall Street Journal, 2010

Goals for this talk

- An outline for how secure web-based password authentication can be
  - As secure as possible
  - As simple as possible
    - No new software[1]
    - No change to user experience
- How secure is this?
- Why aren't implementations any where close?

[1] But a healthy dose of HTML 5 and other modern tricks

Goals for this talk

- An outline for how secure web-based password authentication can be
  - As secure as possible
  - As simple as possible
    - No new software[1]
    - No change to user experience
- How secure is this?
- Why aren't implementations any where close?

[1] But a healthy dose of HTML 5 and other modern tricks

Goals for this talk

- An outline for how secure web-based password authentication can be
  - As secure as possible
  - As simple as possible
    - No new software[1]
    - No change to user experience
- How secure is this?
- Why aren't implementations any where close?

[1] But a healthy dose of HTML 5 and other modern tricks

How password authentication goes wrong

- Keyloggers
- Phishing
- Persistent login cookies

...

- Password recovery questions
- Password re-use
- Password database compromise

...

- Cookie stealing
- Password guessing

Registration (TLS)

Transmitted:

$$y = \mathbf{H}^{Y}_{\ell_2}(u||s), \qquad x = \mathbf{H}^{X}_{\ell_1}(u||p||s)$$

Stored:

$$y = \mathbf{H}^{Y}_{\ell_2}(u||s), \qquad z = \mathbf{H}^{Z}(u||x)$$

- s: site identifier
- u: username
- p: password
- x: "authenticator"

Login (TLS)

Transmitted:

$$u, \quad x = \mathbf{H}^{X}_{\ell_1}(u||p||s)$$

Verified to exist in-database:

$$\mathbf{H}^{Z}(u||x)$$

Returned:

$$K_u, \quad a = \mathbf{AE}_{K_s}(K_u, u, x, t, d)$$

- s: site identifier
- u: username
- p: password
- x: “authenticator”
- K_S: Server master key
- a: session cookie
- K_u: session key
- t: expiration date
- d: additional data

Site interaction (Plain HTTP)

Transmitted as a cookie:

$$a = \mathbf{AE}_{K_s}(K_u, u, x, t, d)$$

Appended to requests:

$$\mathbf{AE}_{K_u}(\text{data})$$

- s: site identifier
- u: username
- p: password
- x: "authenticator"

- K_S: Server master key
- a: session cookie
- K_u: session key
- t: expiration date
- d: additional data

Site interaction (Plain HTTP)

Transmitted as a cookie: `HTTP-only`

$$a = \mathbf{AE}_{K_s}(K_u, u, x, t, d)$$

Appended to requests:

$$\mathbf{AE}_{K_u}(\text{data})$$

- s: site identifier
- u: username
- p: password
- x: "authenticator"
- K_S: Server master key
- a: session cookie
- K_u: session key
- t: expiration date
- d: additional data

Site interaction (Plain HTTP)

Transmitted as a cookie: `HTTP-only`

$$a = \mathbf{AE}_{K_s}(K_u, u, x, t, d)$$

Appended to requests: `JavaScript & HTML5 localStorage`

$$\mathbf{AE}_{K_u}(\text{data})$$

- s: site identifier
- u: username
- p: password
- x: "authenticator"
- K_S: Server master key
- a: session cookie
- K_u: session key
- t: expiration date
- d: additional data

Site interaction (Plain HTTP)

Transmitted as a cookie: `HTTP-only`

$$a = \mathbf{AE}_{K_s}(K_u, u, x, t, d)$$

Optional cookie: `HTTP-only, SECURE`

$$a_{\text{secure}} = \mathbf{AE}_{K_s}(K_u, u, x, t_2 > t, d)$$

Appended to requests: `JavaScript & HTML5 localStorage`

$$\mathbf{AE}_{K_u}(\text{data})$$

- s: site identifier
- u: username
- p: password
- x: "authenticator"

- K_S: Server master key
- a: session cookie
- K_u: session key
- t: expiration date
- d: additional data

Server verification

- Verify & decrypt $a = \mathbf{AE}_{K_s}(K_u, u, x, t, d)$
- Verify & decrypt $\mathbf{AE}_{K_u}(\text{data})$
- Verify that $z = \mathbf{H}^Z(u||x)$ is stored (optional)
- Check timestamp $t \geq$ now
- Check ACL for u, d, data

- s: site identifier
- u: username
- p: password
- x: "authenticator"
- K_S: Server master key
- a: session cookie
- K_u: session key
- t: expiration date
- d: additional data

Server verification

- Verify & decrypt $a = \mathbf{AE}_{K_s}(K_u, u, x, t, d)$
- Verify & decrypt $\mathbf{AE}_{K_u}(\text{data})$
- Verify that $z = \mathbf{H}^Z(u||x)$ is stored (optional)
- Check timestamp $t \geq$ now
- Check ACL for u, d, data

- s: site identifier
- u: username
- p: password
- x: "authenticator"
- K_S: Server master key
- a: session cookie
- K_u: session key
- t: expiration date
- d: additional data

Performance analysis

- Login
  - Server
    - 1 hash
    - 1 DB lookup
    - 1 AE + 1 RNG
  - Browser
    - 1 iterated hash ($\leq$ 0.1 s, PC; $\sim$ 1 s, mobile)
- Interaction
  - Server
    - 2 AE
    - 1 DB lookup (optional)
  - Browser
    - 2 AE ($\leq$ 10 ms, PC; $\leq$ 0.1 s, mobile)

Security analysis-many attacks prevented

- rainbow tables
- online password guessing
- cookie modification

. . .

- session key theft (XSS)
- session cookie theft (sidejacking)
- read-only DB access
- user probing

. . .

- XSS + sidejacking
- DB access + cookie theft
- malware in browser
- password theft
- phishing
- persistent log-in

Security analysis-many attacks prevented

- rainbow tables
- online password guessing
- cookie modification

  . . .
- session key theft (XSS)
- session cookie theft (sidejacking)
- read-only DB access
- user probing

  . . .
- XSS + sidejacking
- DB access + cookie theft
- malware in browser
- password theft
- phishing
- persistent log-in

Security analysis-many attacks prevented

- rainbow tables
- online password guessing
- cookie modification

  . . .
- session key theft (XSS)
- session cookie theft (sidejacking)
- read-only DB access
- user probing

  . . .
- XSS + sidejacking
- DB access + cookie theft
- malware in browser
- password theft
- phishing
- persistent log-in

Some sobering facts

- Over 90% of the top 500 websites collect passwords
- 29-50% store them in the clear
- 84% do not prevent brute force attacks at all
- 40% implement TLS correctly (20% incorrectly, 40% not at all)
- hashing in browser, HTTP-only cookies extremely rare...

Even the frameworks get it wrong!

Default parameters in common web frameworks and CMSes

Language	Framework	Plugin	ver.	Algorithm	Iteration count	Salt (bits)	Output (bits)	notes
.NET	ASP.NET		4	SHA-1	1	none	160	also supports cleartext storage
PHP	built-in		5.3	MD5	1,000	72	132	MD5 crypt()
PHP	CakePHP		7	SHA-1	1	none	160	
PHP	Drupal		7	SHA-512	16,384	48	256	
PHP	Joomla!		1.5	MD5	1	48	128	
PHP	WordPress		3.1	Blowfish	256	48	132	uses PHPPass
Python	Django		1.2	SHA-1	1	20	160	also supports unsalted MD5
Python	generic WSGI	repoze.who	2.0	SHA-1	1	none	160	Recommended for Pylons
Ruby	Rails	restful_auth		SHA-1	10	160	160	salt has only 80 bits of entropy

Is it worthwhile to fix password authentication?

`jcb82@cl.cam.ac.uk`