User authentication on the web

Joseph Bonneau

jcb82@cl.cam.ac.uk

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

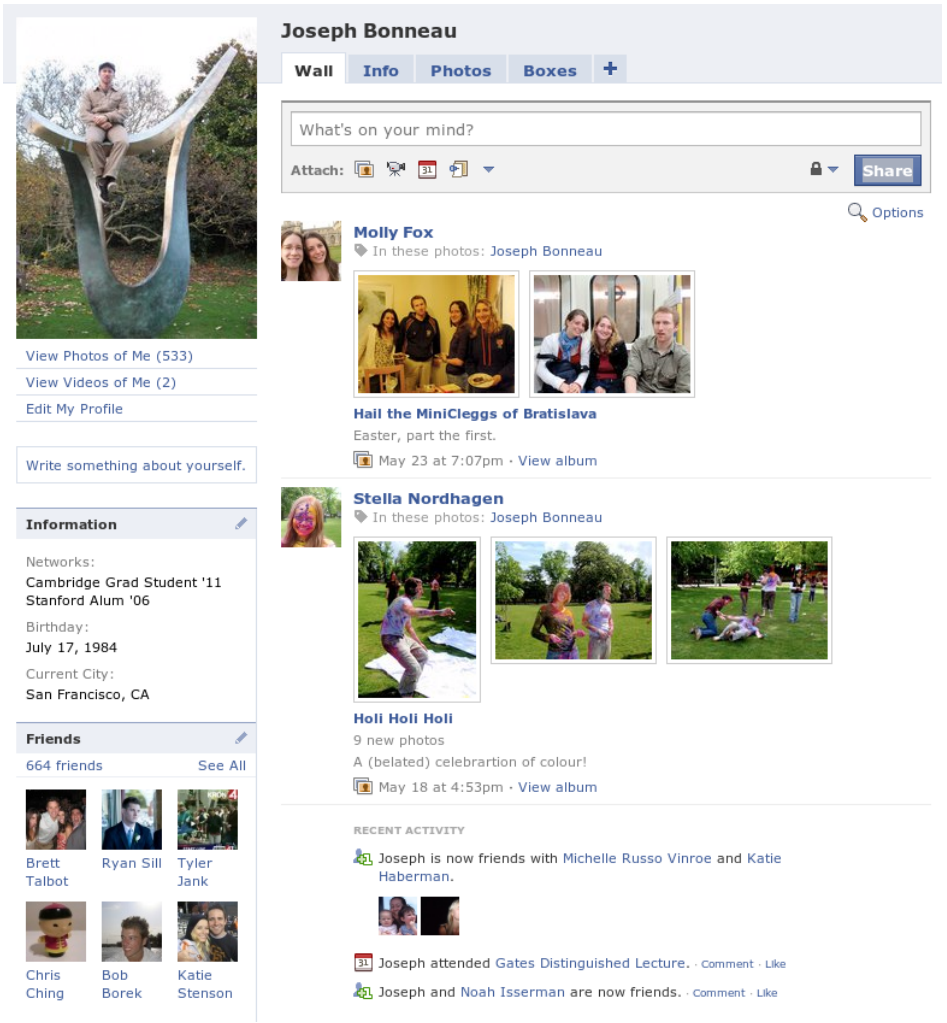
SOCIALNETS workshop

November 18, 2010

Looming authentication challenges

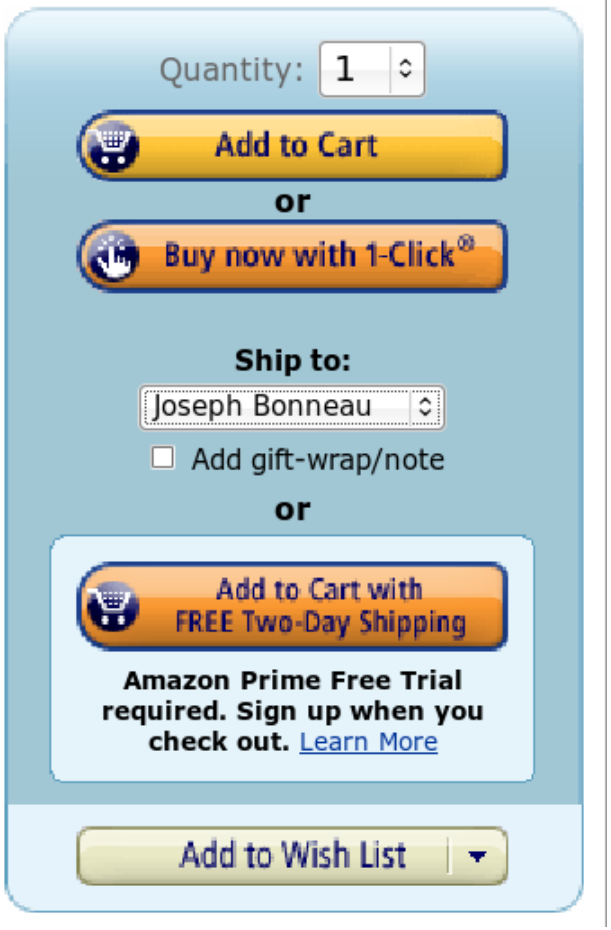
1. The old world
2. The emerging world

WEIS 2010: Large study of password deployments



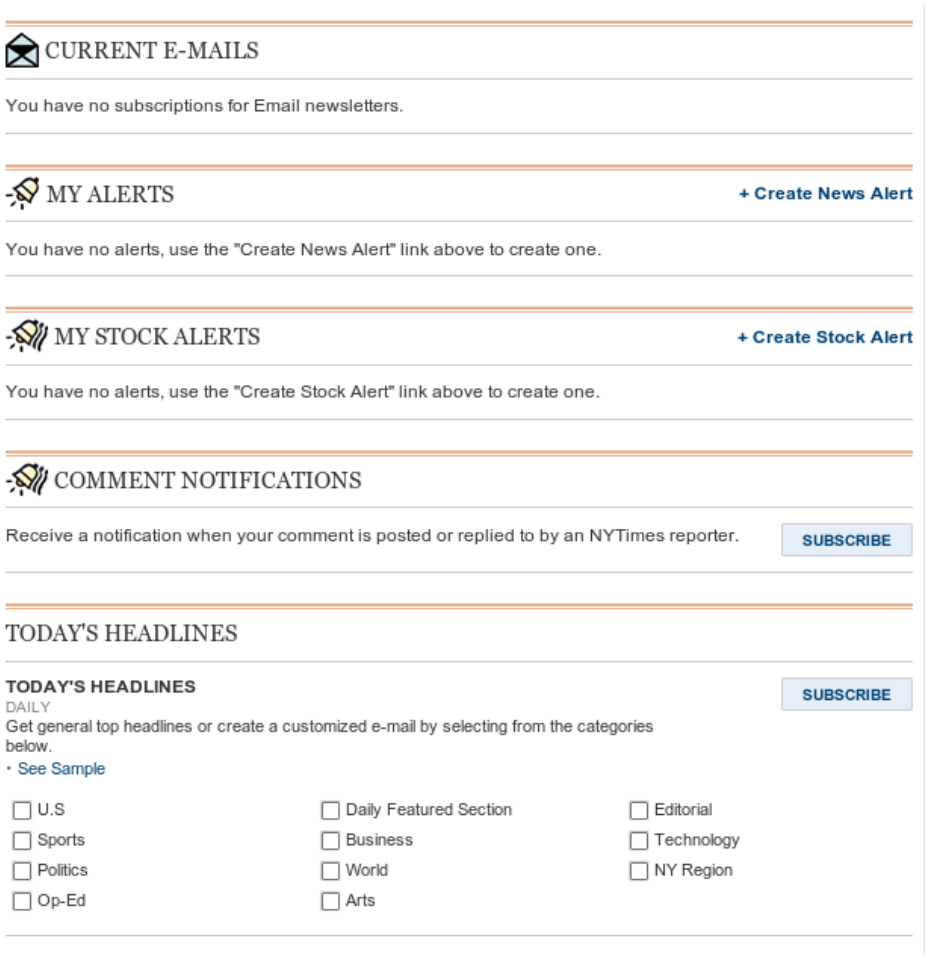
"Identity" websites

WEIS 2010: Large study of password deployments



"E-Commerce" websites

WEIS 2010: Large study of password deployments

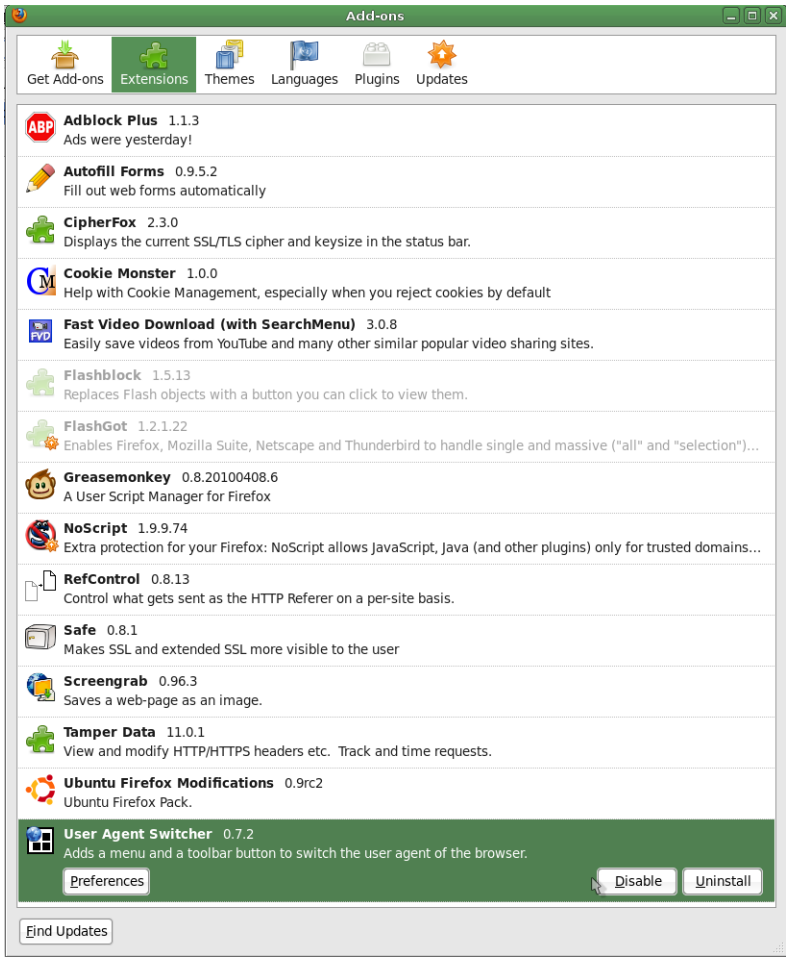


"Content" websites

WEIS 2010: Large study of password deployments

Mozilla Firefox v 3.5.8 with:

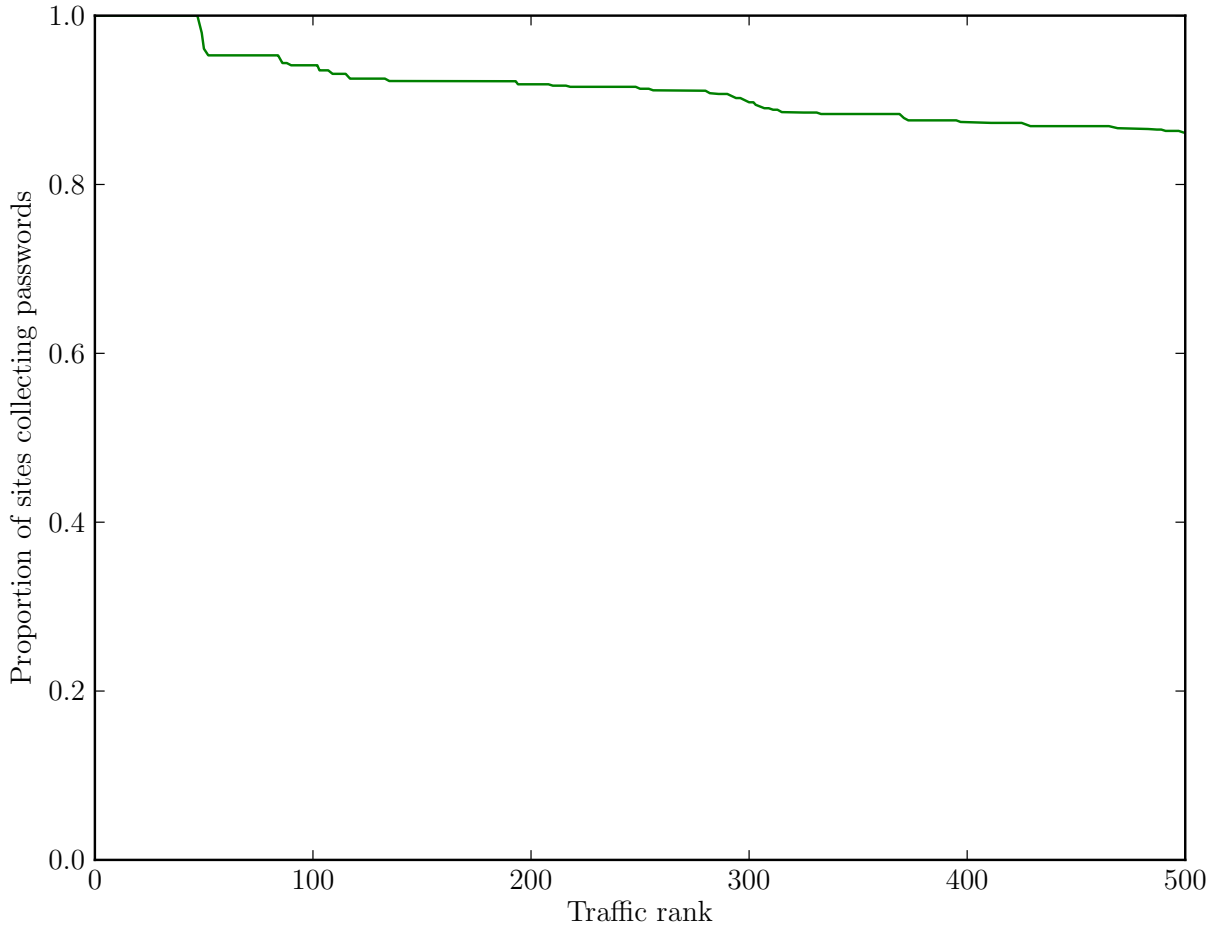
- **Autofill Forms 0.9.5.2**
- **CipherFox 2.3.0**
- **Cookie Monster 0.98.0**
- **DOM Inspector 2.0.4**
- **Greasemonkey 0.8.20100211.5**
- **Screengrab 0.96.2**
- **Tamper Data 11.0.1**



WEIS 2010: Large study of password deployments

feature	scoring
enrolment	
Password selection advice given	+1 pt
Minimum password length required	+1 pt
Dictionary words prohibited	+1 pt
Numbers or symbols required	+1 pt
User list protected from probing	+1 pt
Cleartext password sent in email after enrolment	−1 pt
login	
Password hashed in-browser before POST	+1 pt
Limits placed on password guessing	+1 pt
User list protected from probing	+1 pt
Federated identity login accepted	+1 pt
password update	
Password re-entry required to authorise update	+1 pt
Notification email sent after password reset	+1 pt
password recovery	
Password update required after recovery	+1 pt
Cleartext password sent in email upon request	−1 pt
User list protected from probing	+1 pt
encryption	
Full TLS for all password submission	+2 pts
POST only TLS for password submission	+1 pt

The realities of web authentication

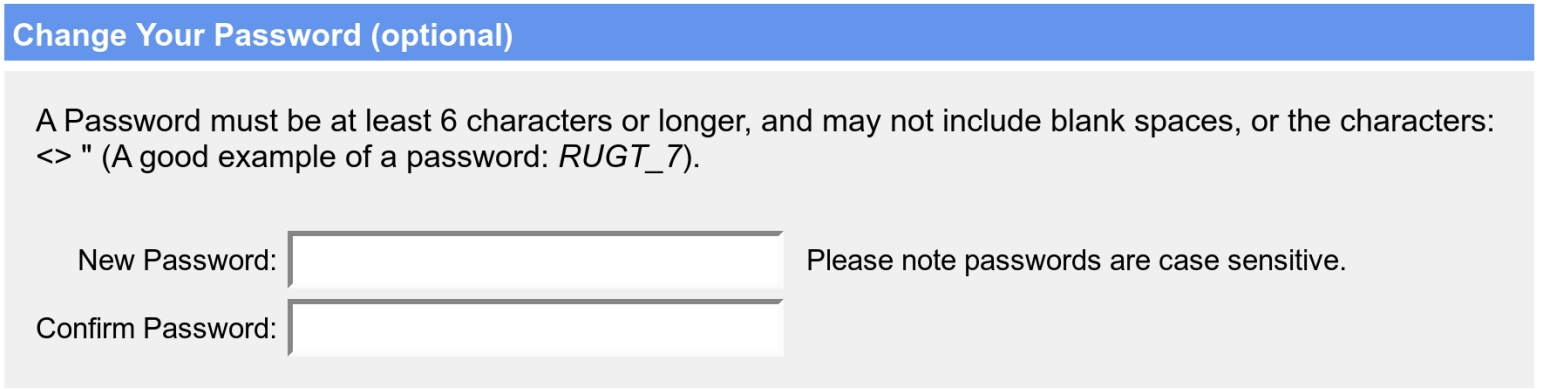


Frequency of password collection

The realities of web authentication

- $\sim$ all websites collect email address as username
- $\sim$ all websites use email for password reset
- $\sim$ all websites use persistent login cookies by default

Many schoolbook errors are quite common



29-50% of sites store passwords in the clear

Many schoolbook errors are quite common

Part of the RockYou website

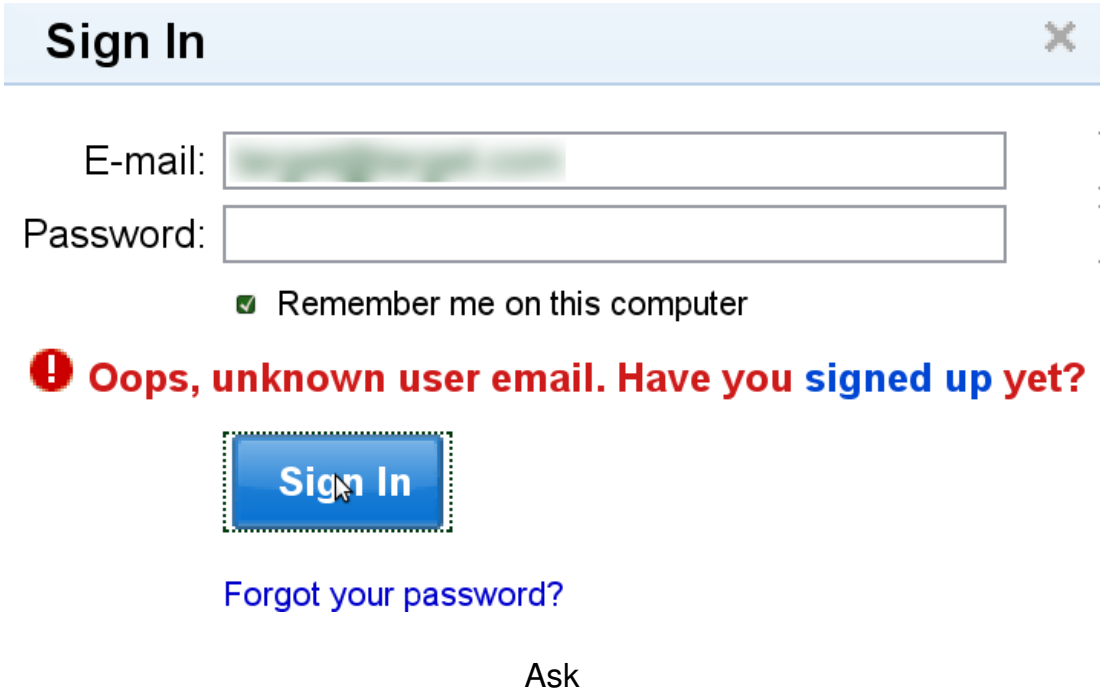
RockYou SQL injection hack
January 2010

Many schoolbook errors are quite common

countermeasure	I	E	C	Tot.
CAPTCHA	11	2	1	**14**
timeout	2	1	2	**5**
reset	1	3	1	**5**
none	37	43	46	**126**

Many websites allow unlimited brute-force guessing

Many schoolbook errors are quite common



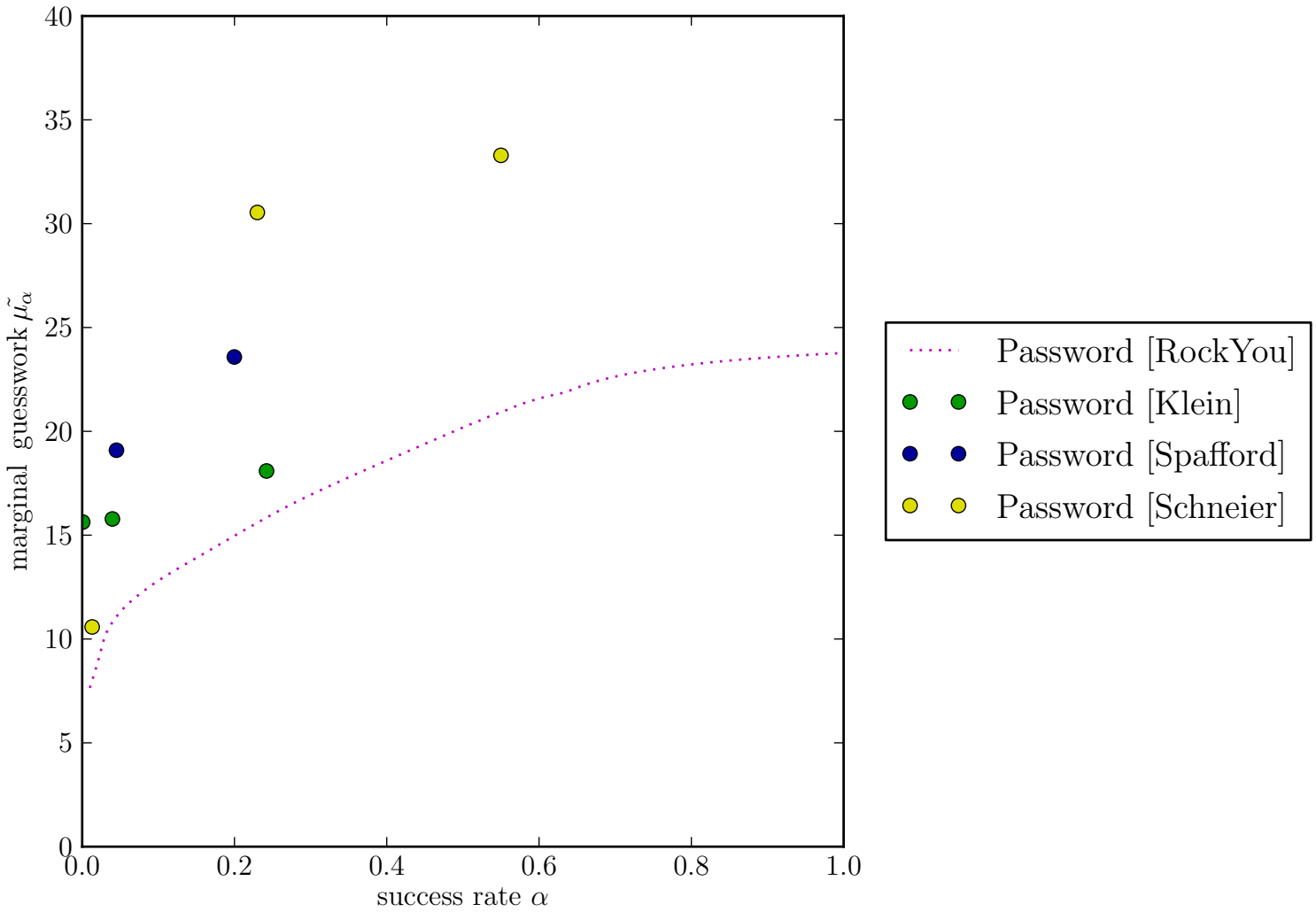
User probing is rarely prevented

Many schoolbook errors are quite common

interface	I	E	C	Tot.
enrolment	4	1	1	**6**
login	43	41	38	**132**
reset	11	7	2	**20**
all	1	1	0	**2**

User probing is rarely prevented

Many schoolbook errors are quite common

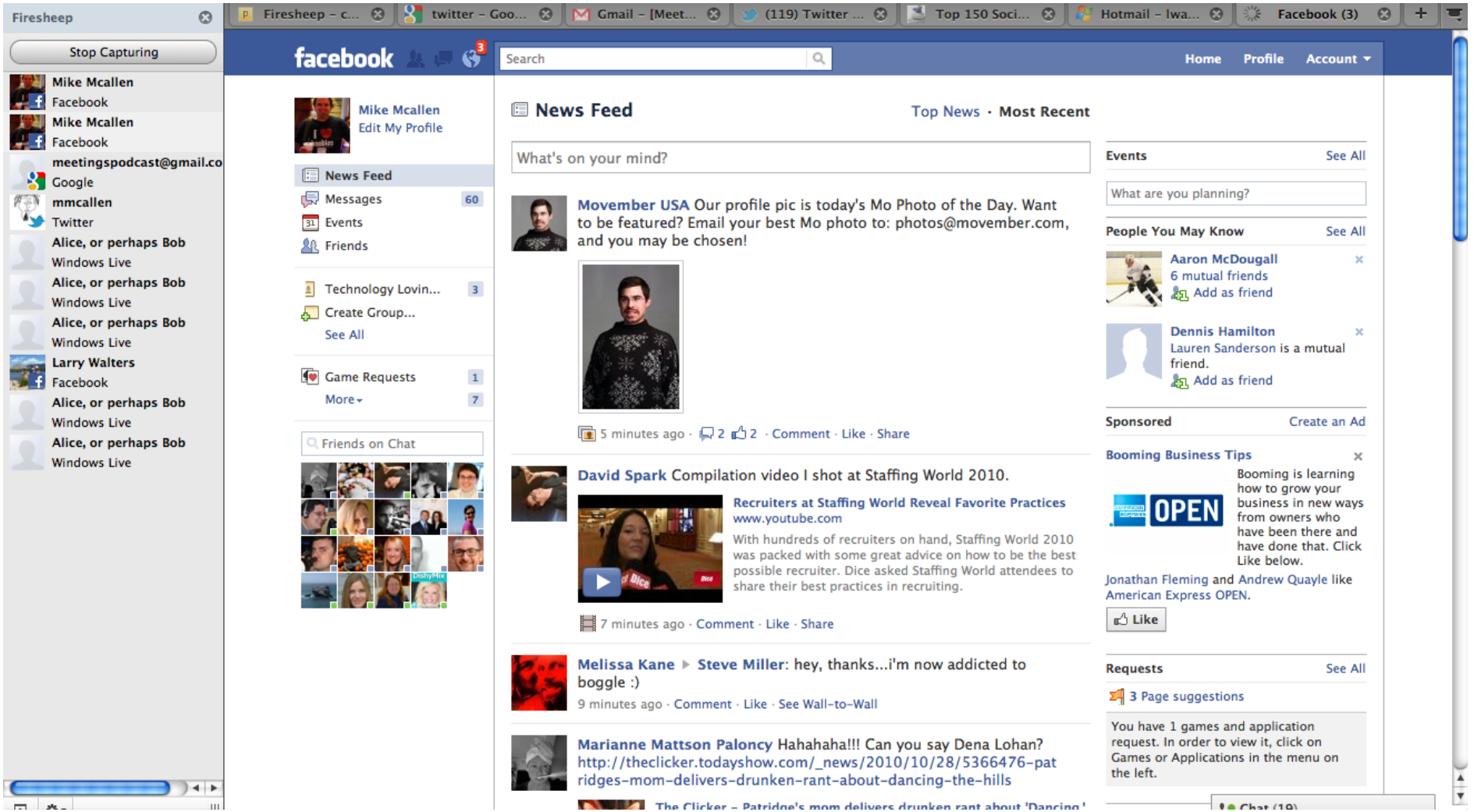


Many schoolbook errors are quite common

TLS Deployment	I	E	C	Tot.
Full	10	39	10	**59**
Full/POST	3	1	1	**5**
Inconsistent	14	6	5	**25**
None	23	4	34	**61**

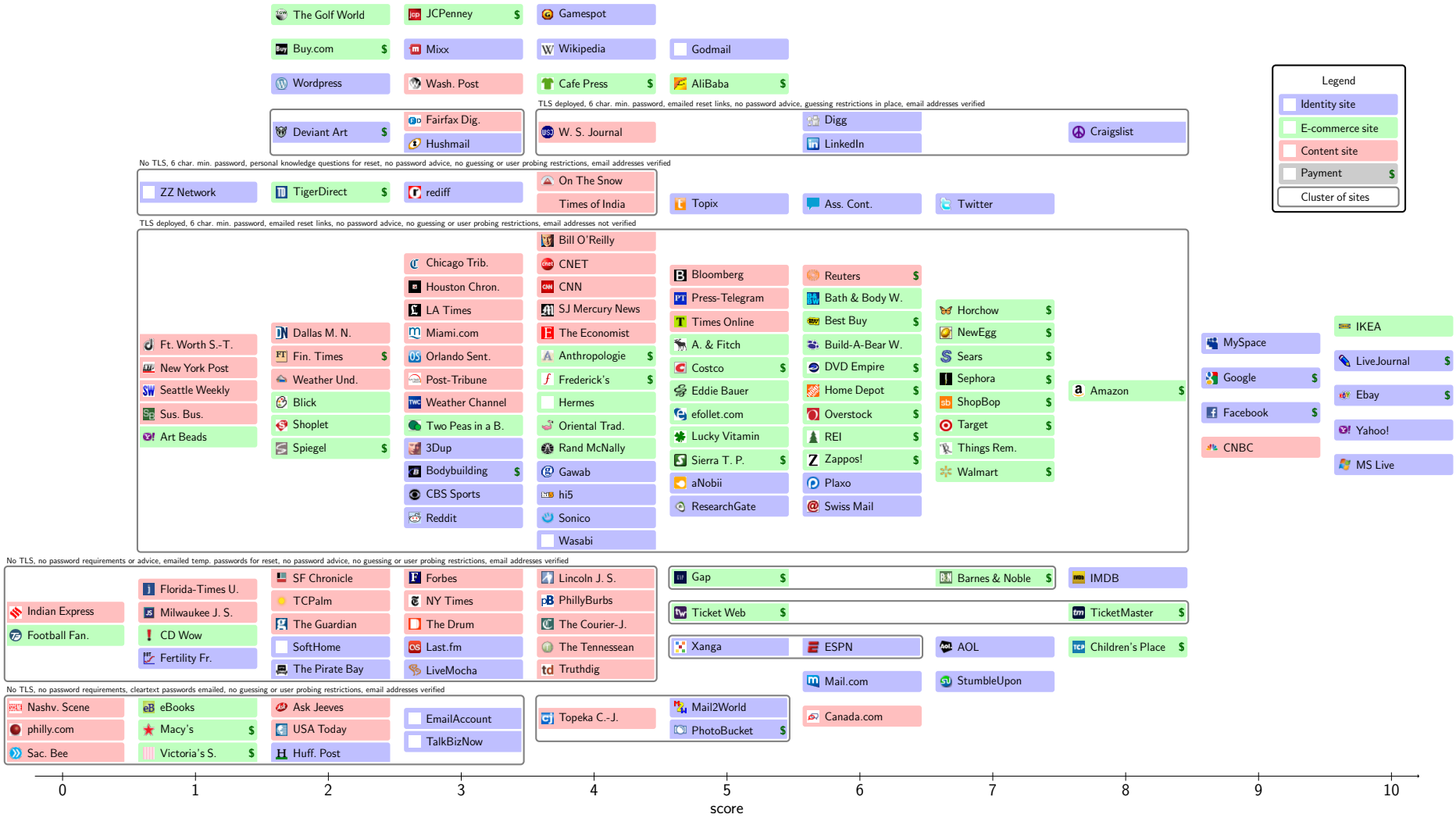
TLS deployment remains uneven, poorly done

Many schoolbook errors are quite common

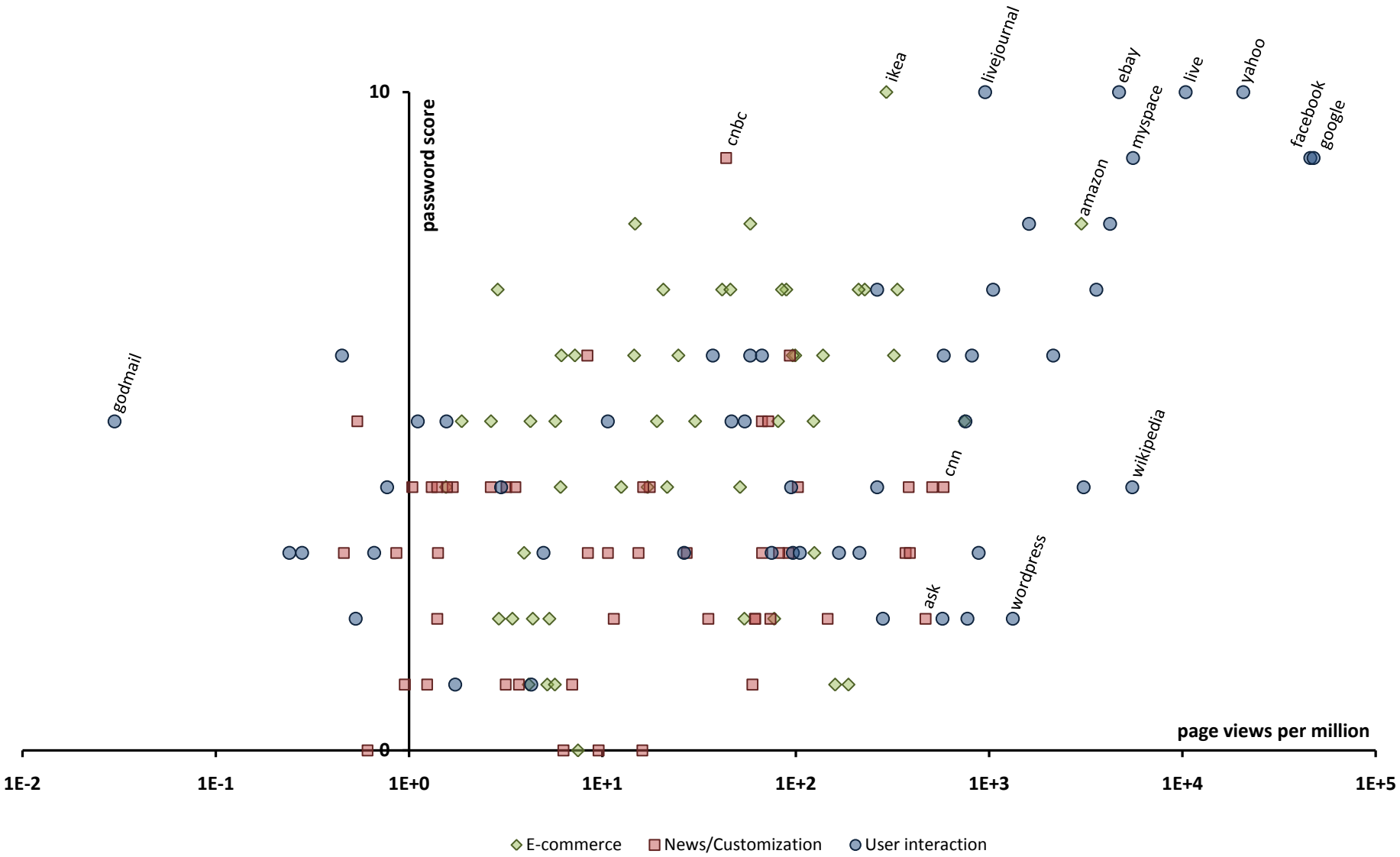


Firesheep

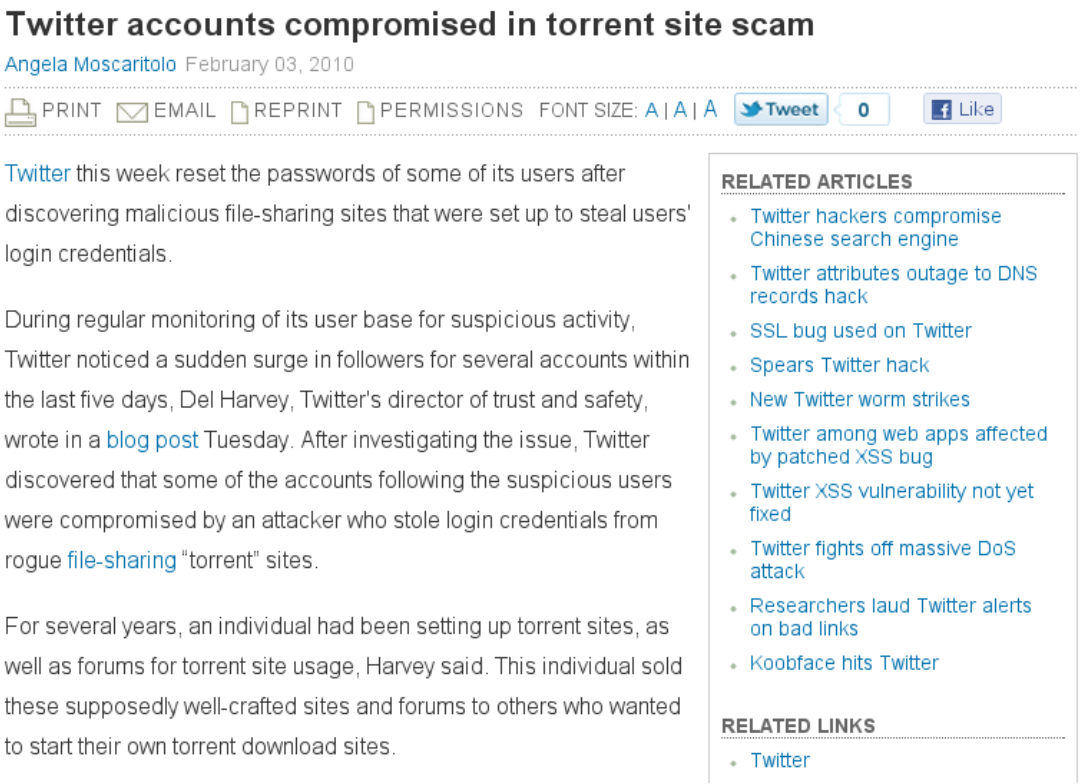
Security policies vary far more than requirements



More popular sites do better



Economic failures



Twitter accounts compromised in torrent site scam

Angela Moscaritolo February 03, 2010

PRINT EMAIL REPRINT PERMISSIONS FONT SIZE: A | A | A Tweet 0 Like

Twitter this week reset the passwords of some of its users after discovering malicious file-sharing sites that were set up to steal users' login credentials.

During regular monitoring of its user base for suspicious activity, Twitter noticed a sudden surge in followers for several accounts within the last five days, Del Harvey, Twitter's director of trust and safety, wrote in a blog post Tuesday. After investigating the issue, Twitter discovered that some of the accounts following the suspicious users were compromised by an attacker who stole login credentials from rogue file-sharing "torrent" sites.

For several years, an individual had been setting up torrent sites, as well as forums for torrent site usage, Harvey said. This individual sold these supposedly well-crafted sites and forums to others who wanted to start their own torrent download sites.

RELATED ARTICLES

- Twitter hackers compromise Chinese search engine
- Twitter attributes outage to DNS records hack
- SSL bug used on Twitter
- Spears Twitter hack
- New Twitter worm strikes
- Twitter among web apps affected by patched XSS bug
- Twitter XSS vulnerability not yet fixed
- Twitter fights off massive DoS attack
- Researchers laud Twitter alerts on bad links
- Koobface hits Twitter

RELATED LINKS

- Twitter

- Bad websites can do real damage to good ones
- Password insecurity is a negative externality
- Password over-collection is a tragedy of the commons

Economic failures

- Bad websites can do real damage to good ones
- Password insecurity is a negative externality
- Password over-collection is a tragedy of the commons

Economic failures

- Bad websites can do real damage to good ones
- Password insecurity is a negative externality
- Password over-collection is a tragedy of the commons

Looming authentication challenges

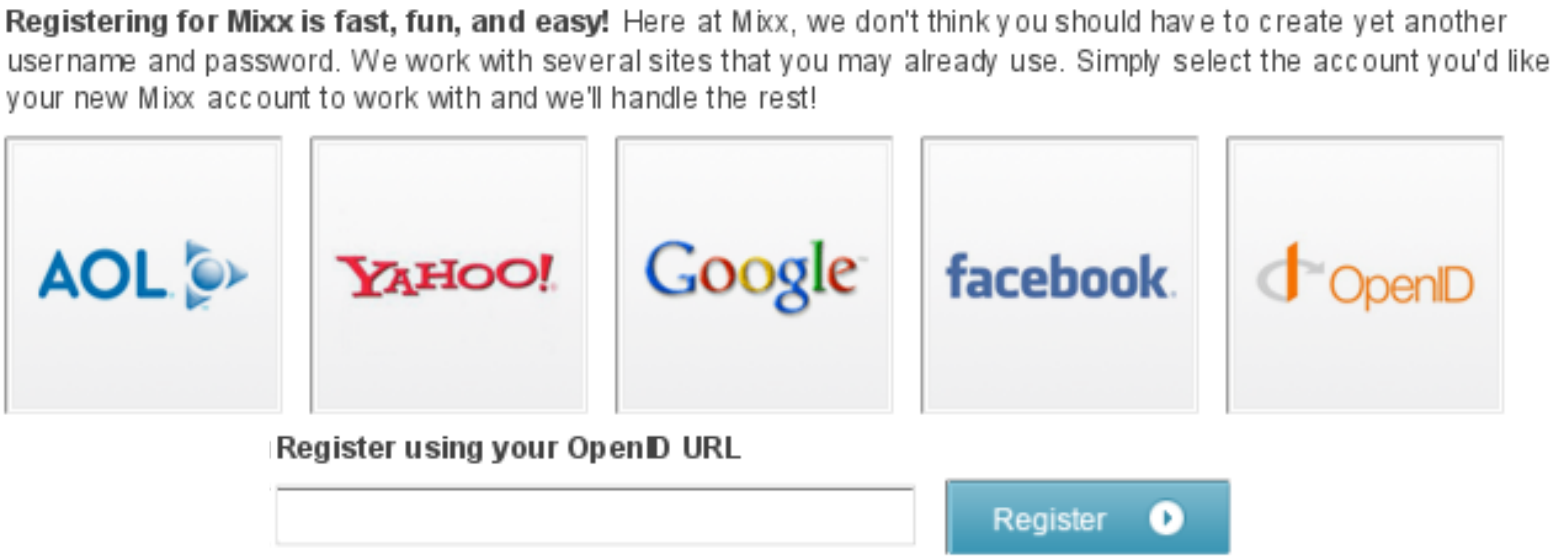
1. The old world
2. The emerging world

OpenID—Single sign-on

R	Relying party (www.example.com)
P	OpenID Provider (Facebook, Google, etc.)
$\mathbf{U}_E$	End user (a human)
$\mathbf{U}_A$	User agent (a browser)

$\mathbf{U}_E \longrightarrow \mathbf{R}$ I'm **U**@**P**!

OpenID—Single sign-on



OpenID—Single sign-on

R	Relying party (www.example.com)
P	OpenID Provider (Facebook, Google, etc.)
$\mathbf{U}_E$	End user (a human)
$\mathbf{U}_A$	User agent (a browser)

$$\mathbf{U}_E \longrightarrow \mathbf{R} \quad \text{I'm } \mathbf{U}@\mathbf{P}!$$

$$\mathbf{R} \longleftrightarrow \mathbf{P} \quad K_{\text{R-P}}, n \leftarrow \text{D-H key exchange}$$

OpenID—Single sign-on

R	Relying party (www.example.com)
P	OpenID Provider (Facebook, Google, etc.)
$\mathbf{U}_E$	End user (a human)
$\mathbf{U}_A$	User agent (a browser)

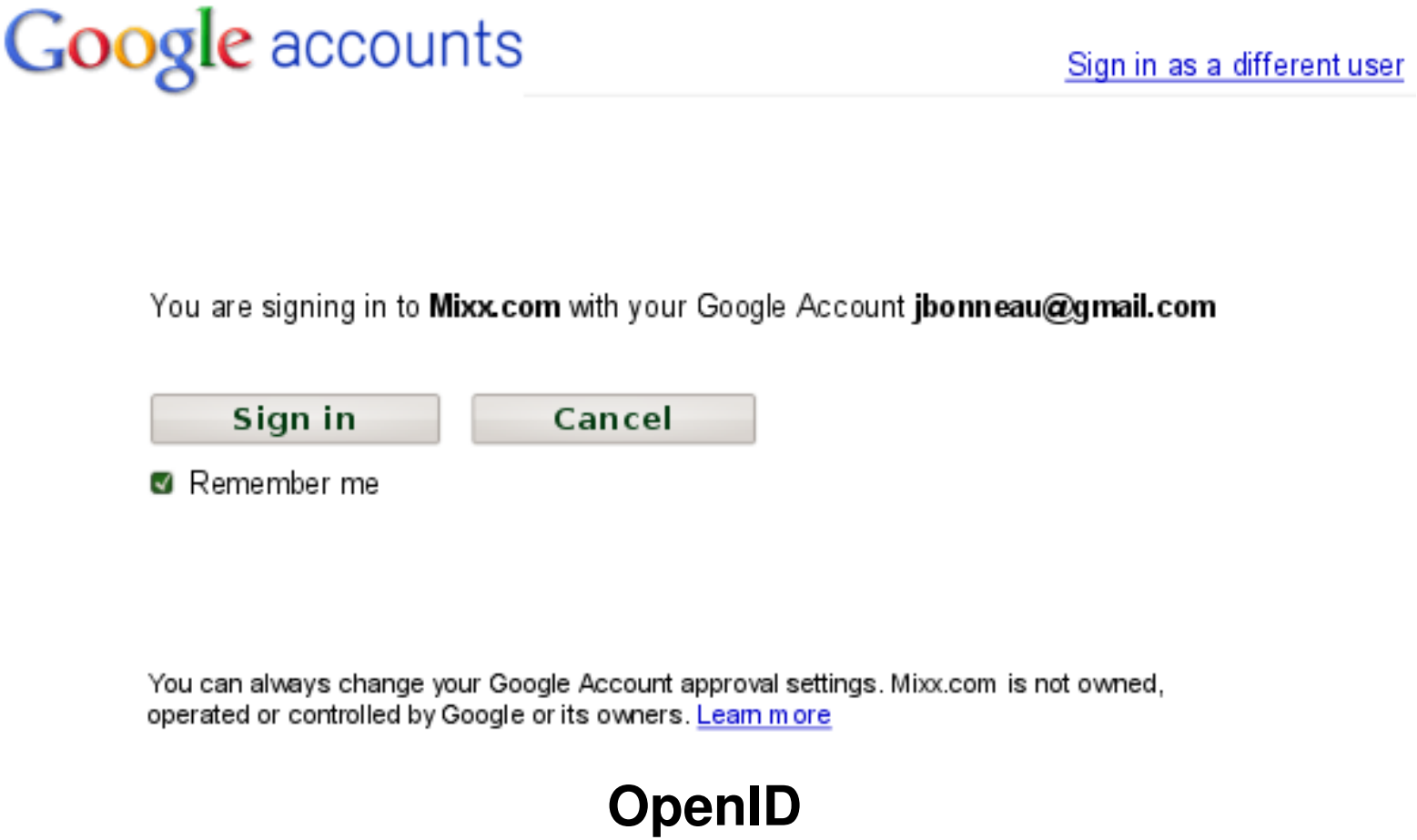
$\mathbf{U}_E$	$\longrightarrow$	**R**	I'm **U**@**P**!
R	$\longleftrightarrow$	**P**	$K_{\text{R-P}}, n \leftarrow$ D-H key exchange
$\mathbf{U}_E$	$\longleftarrow$	**R**	OK, go verify with **P** (`HTTP 302`)
$\mathbf{U}_E$	$\longrightarrow$	**P**	I want to talk to **R**, who you share n with

OpenID—Single sign-on

R	Relying party (www.example.com)
P	OpenID Provider (Facebook, Google, etc.)
$\mathbf{U}_E$	End user (a human)
$\mathbf{U}_A$	User agent (a browser)

$\mathbf{U}_E$	$\longrightarrow$	**R**	I'm **U**@**P**!
R	$\longleftrightarrow$	**P**	$K_{\text{R-P}}, n \leftarrow$ D-H key exchange
$\mathbf{U}_E$	$\longleftarrow$	**R**	OK, go verify with **P** (`HTTP 302`)
$\mathbf{U}_E$	$\longrightarrow$	**P**	I want to talk to **R**, who you share n with
$\mathbf{U}_E$	$\longleftarrow$	**P**	Sure you want to talk to **R**?

OpenID—Single sign-on



OpenID

OpenID—Single sign-on

R	Relying party (www.example.com)
P	OpenID Provider (Facebook, Google, etc.)
$\mathbf{U}_E$	End user (a human)
$\mathbf{U}_A$	User agent (a browser)

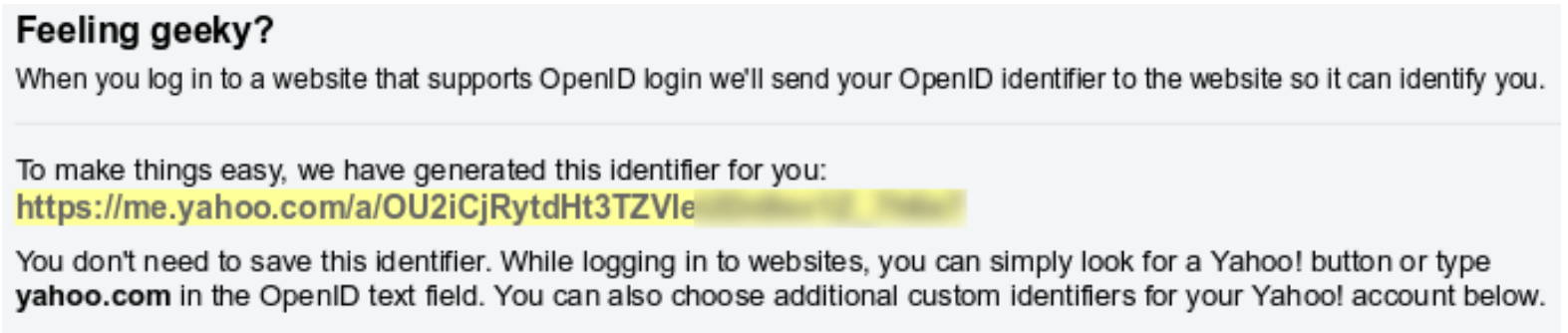
$\mathbf{U}_E$	$\longrightarrow$	**R**	I'm **U**@**P**!
R	$\longleftrightarrow$	**P**	$K_{\text{R-P}}, n \leftarrow$ D-H key exchange
$\mathbf{U}_E$	$\longleftarrow$	**R**	OK, go verify with **P** (`HTTP 302`)
$\mathbf{U}_E$	$\longrightarrow$	**P**	I want to talk to **R**, who you share n with
$\mathbf{U}_E$	$\longleftarrow$	**P**	Sure you want to talk to **R**?
$\mathbf{U}_E$	$\longrightarrow$	**P**	Yes, here's my password: p

OpenID—Single sign-on

R	Relying party (www.example.com)
P	OpenID Provider (Facebook, Google, etc.)
$\mathbf{U}_E$	End user (a human)
$\mathbf{U}_A$	User agent (a browser)

$\mathbf{U}_E$	$\longrightarrow$	**R**	I'm **U**@**P**!
R	$\longleftrightarrow$	**P**	$K_{\text{R-P}}, n \leftarrow$ D-H key exchange
$\mathbf{U}_E$	$\longleftarrow$	**R**	OK, go verify with **P** (`HTTP 302`)
$\mathbf{U}_E$	$\longrightarrow$	**P**	I want to talk to **R**, who you share n with
$\mathbf{U}_E$	$\longleftarrow$	**P**	Sure you want to talk to **R**?
$\mathbf{U}_E$	$\longrightarrow$	**P**	Yes, here's my password: p
$\mathbf{U}_E$	$\longleftarrow$	**P**	Okay, use $\mathbf{MAC}_{K_{\text{R-P}}}(\mathbf{U}, \mathbf{P})$ (`HTTP 302`)
$\mathbf{U}_E$	$\longrightarrow$	**R**	$\mathbf{MAC}_{K_{\text{R-P}}}(\mathbf{U}, \mathbf{P})$! See, I'm **U**@**P**

OpenID—Single sign-on



Yahoo!

OpenID—Single sign-on

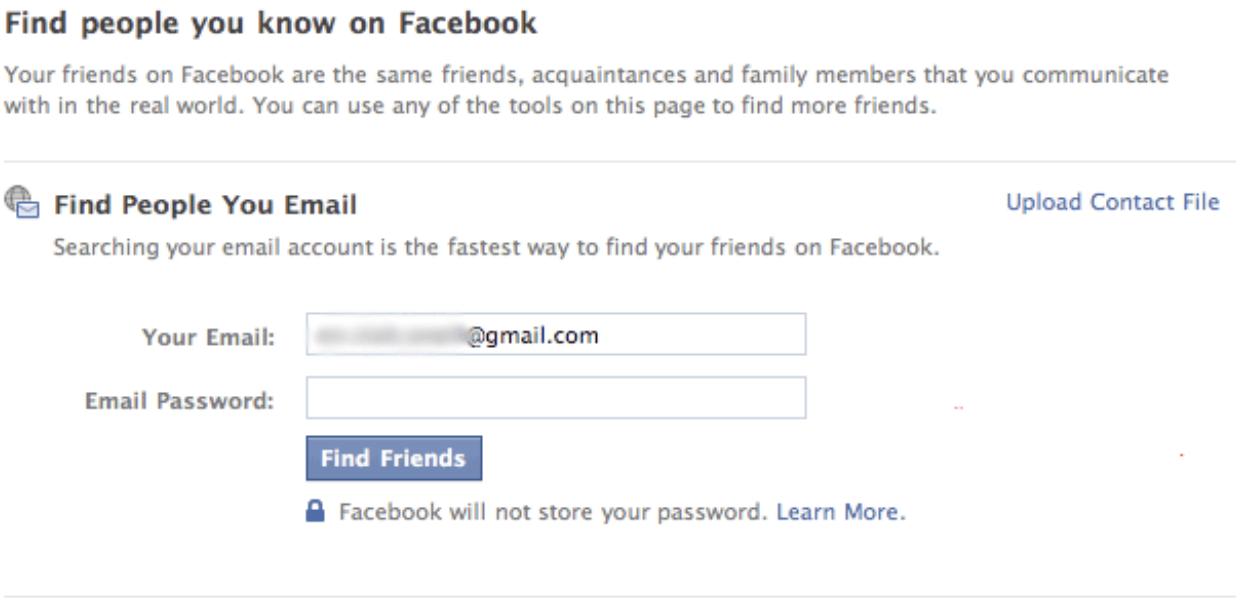
R	Relying party (www.example.com)
P	OpenID Provider (Facebook, Google, etc.)
$\mathbf{U}_E$	End user (a human)
$\mathbf{U}_A$	User agent (a browser)

$\mathbf{U}_E$	$\longrightarrow$	**R**	I'm **U**@**P**!
R	$\longleftrightarrow$	**P**	$K_{\text{R-P}}, n \leftarrow$ D-H key exchange
$\mathbf{U}_A$	$\longleftarrow$	**R**	OK, go verify with **P** (`HTTP 302`)
$\mathbf{U}_A$	$\longrightarrow$	**P**	I want to talk to **R**, here's my cookie c
$\mathbf{U}_A$	$\longleftarrow$	**P**	Okay, use $\mathbf{MAC}_{K_{\text{R-P}}}(\mathbf{U}, \mathbf{P})$
$\mathbf{U}_A$	$\longrightarrow$	**R**	$\mathbf{MAC}_{K_{\text{R-P}}}(\mathbf{U}, \mathbf{P})$! See, I'm **U**@**P**

(`auth-immediate`)

OAuth—Delegating API access

The Dark Ages

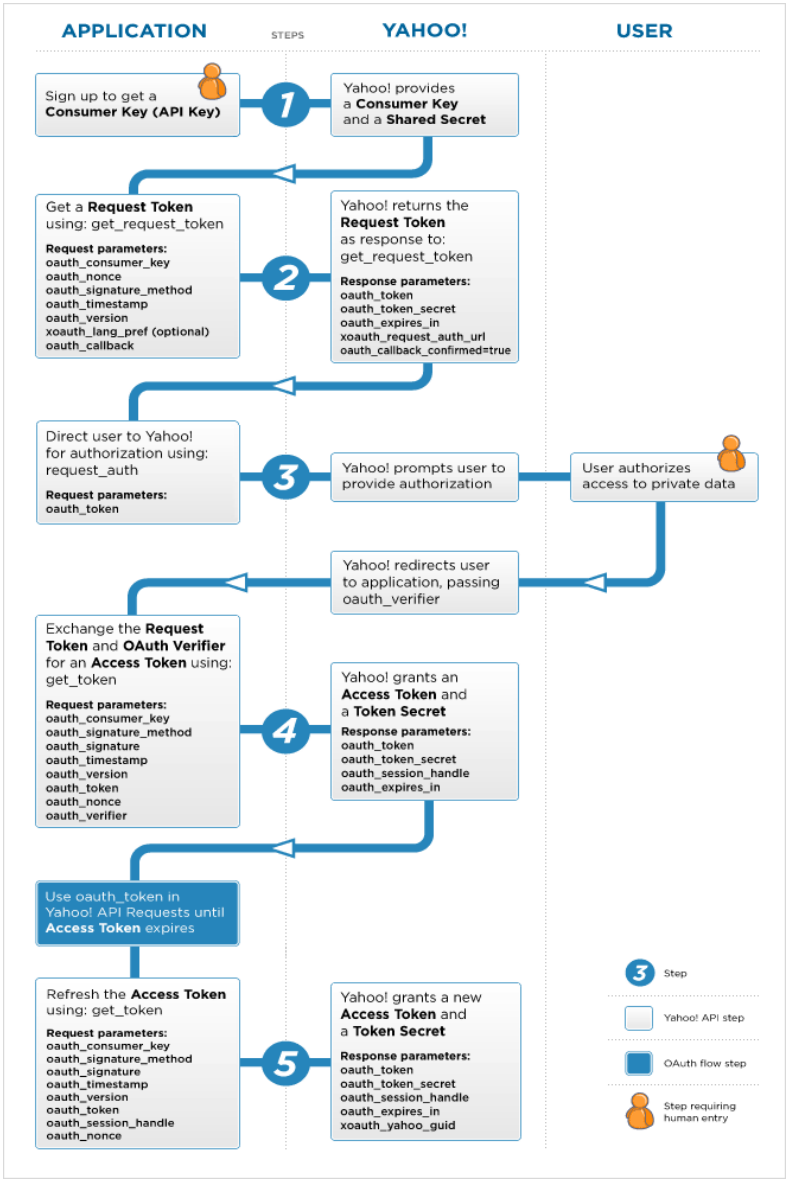


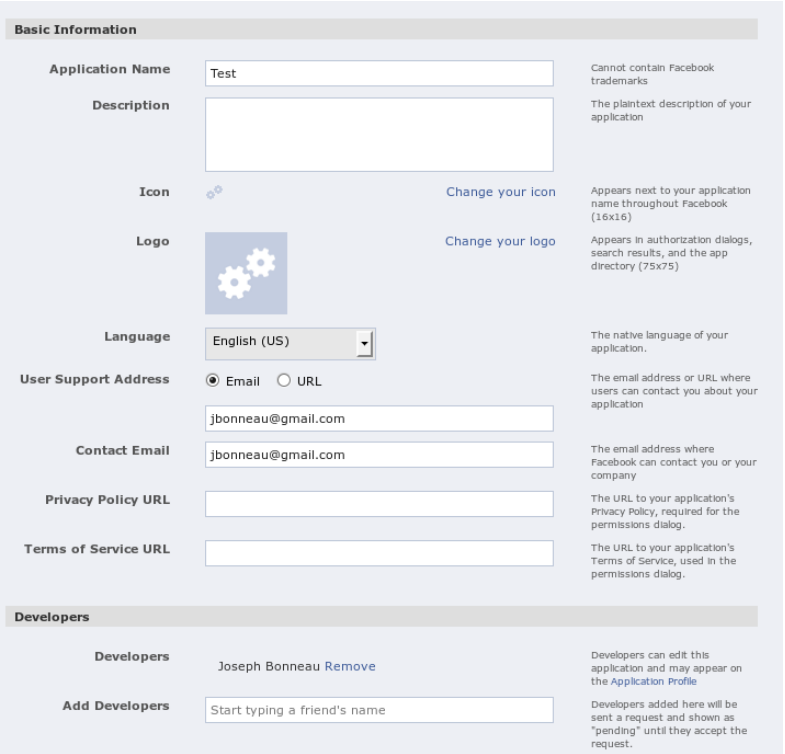
OAuth—Delegating API access

The Middle Ages

1. Facebook Connect
2. Google AuthSub
3. Yahoo BBAuth
4. Twitter API: HTTP `basic-authentication`

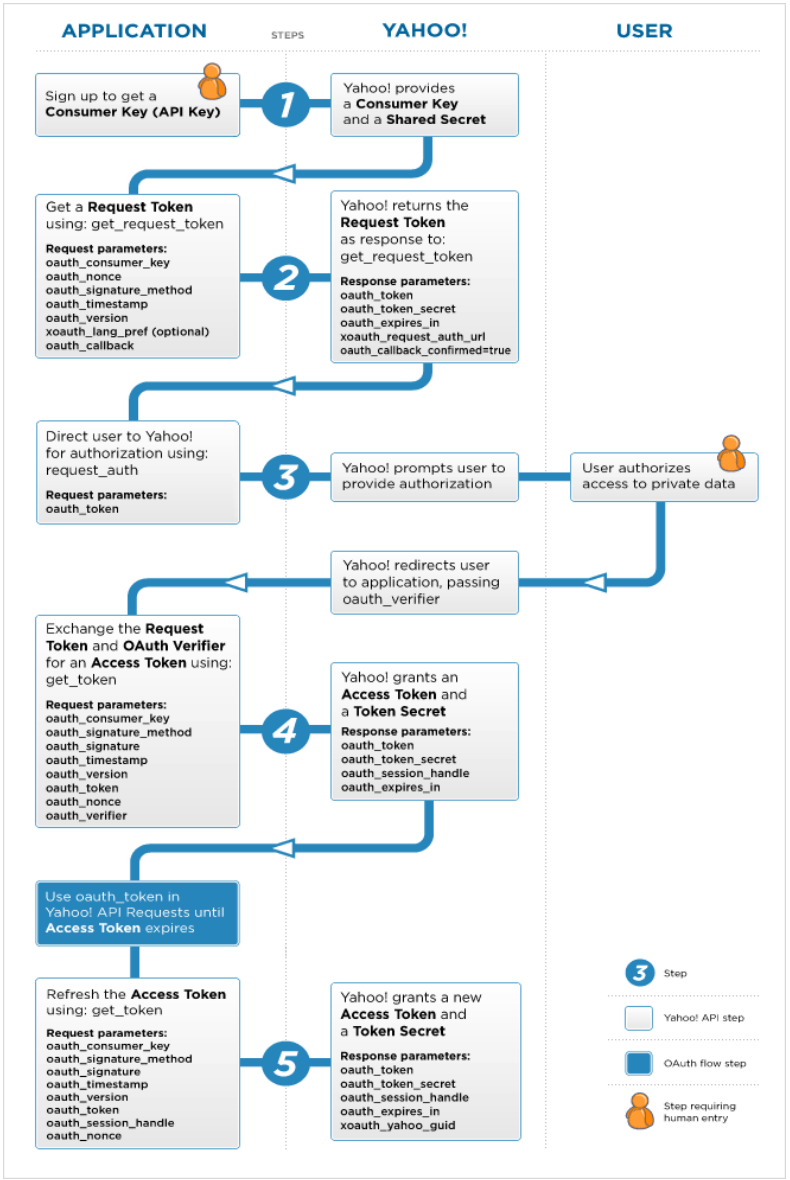
OAuth—Delegating API access

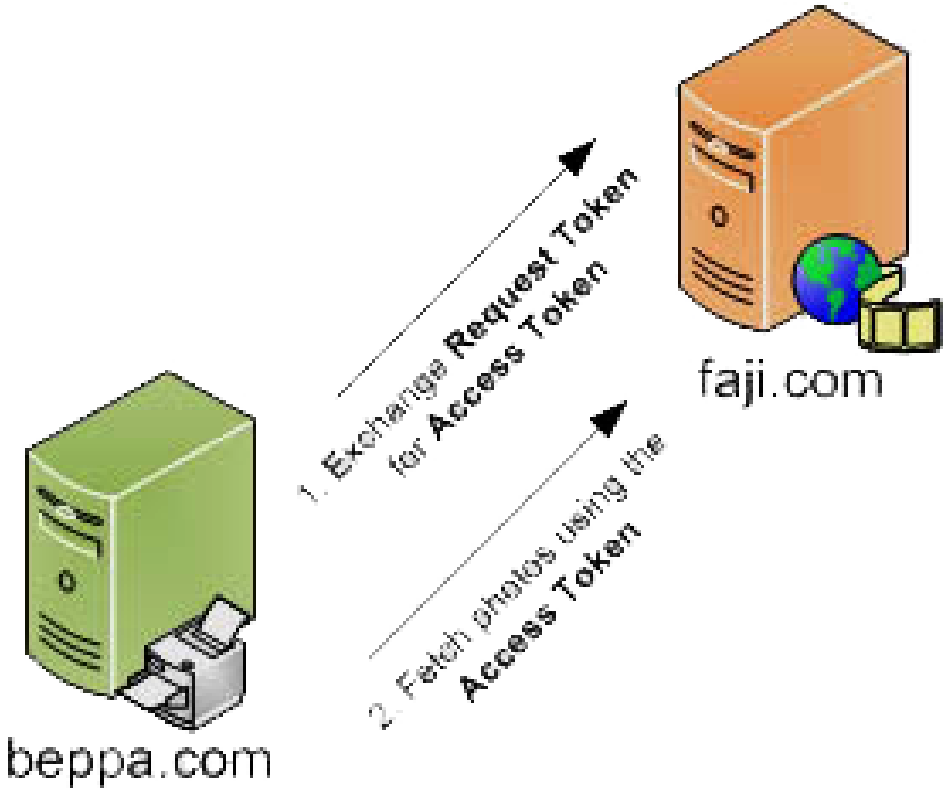




1. App registration
2. Access request
3. User approval
4. API Access

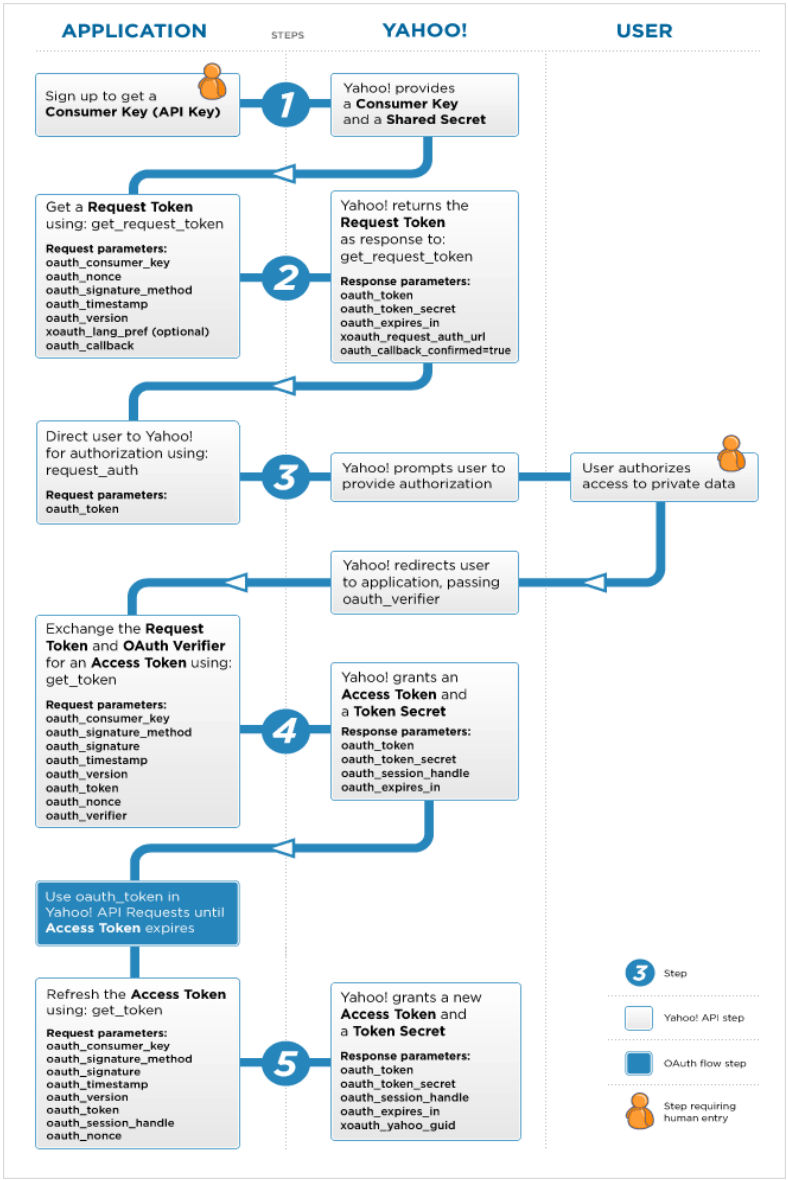
OAuth—Delegating API access

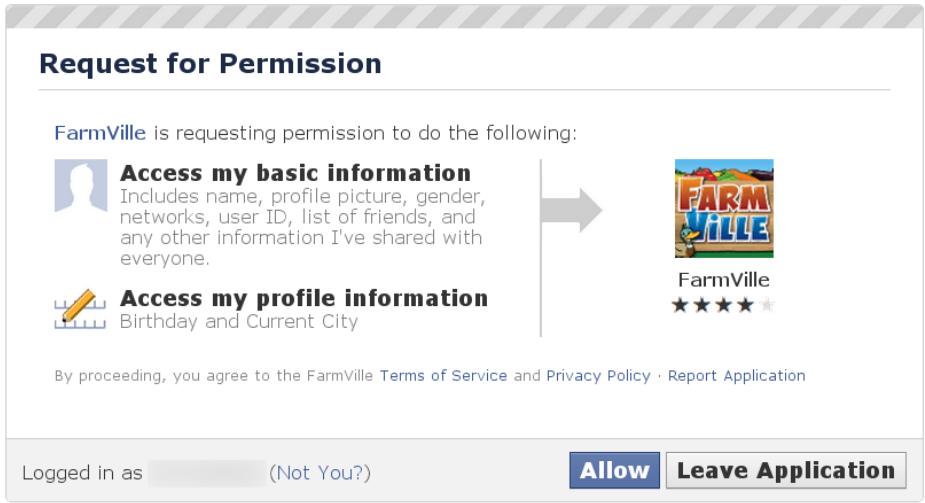




1. App registration
2. Access request
3. User approval
4. API Access

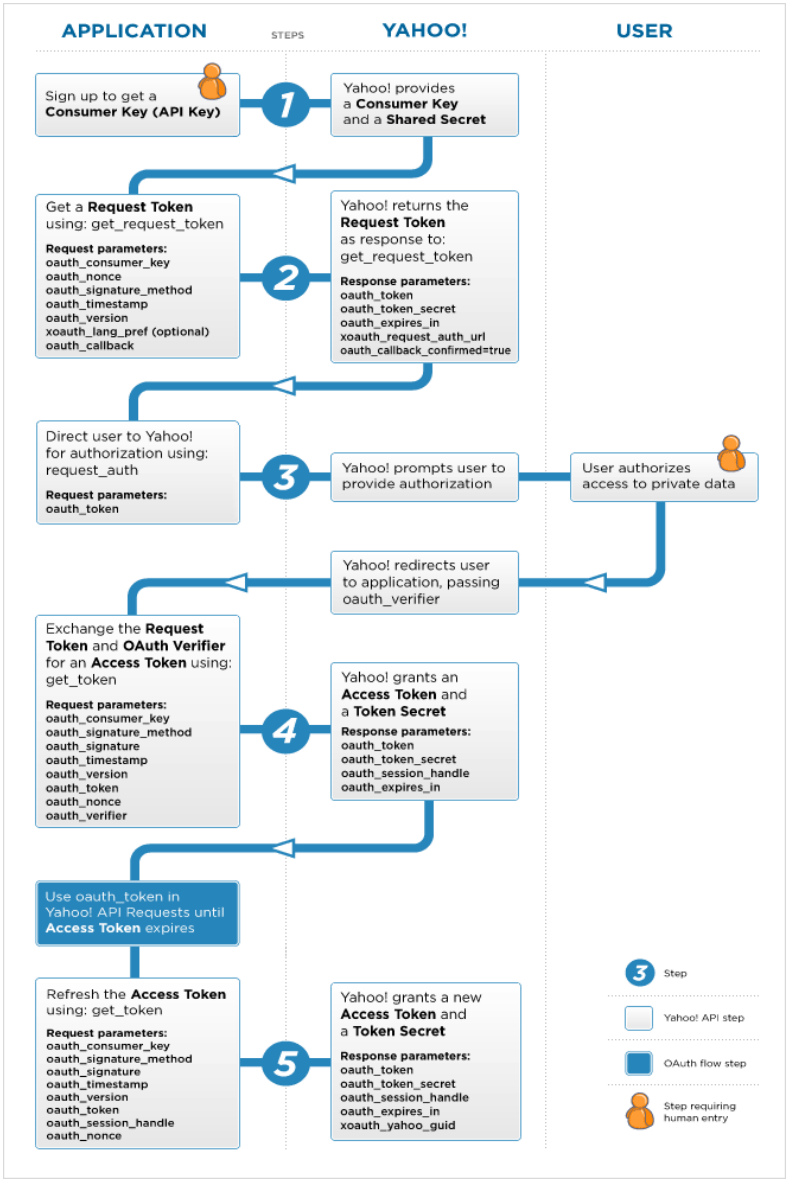
OAuth—Delegating API access





1. App registration
2. Access request
3. **User approval**
4. API Access

OAuth—Delegating API access



PLAINTEXT:

$$M||K_{\text{app}}||K_{\text{user}}$$

HMAC_SHA1:

$$\mathbf{MAC}_{K_{\text{app}}||K_{\text{user}}}(M)$$

RSA_SHA1:

$$\mathbf{Sign}_{K_{\text{app}}}(M)$$

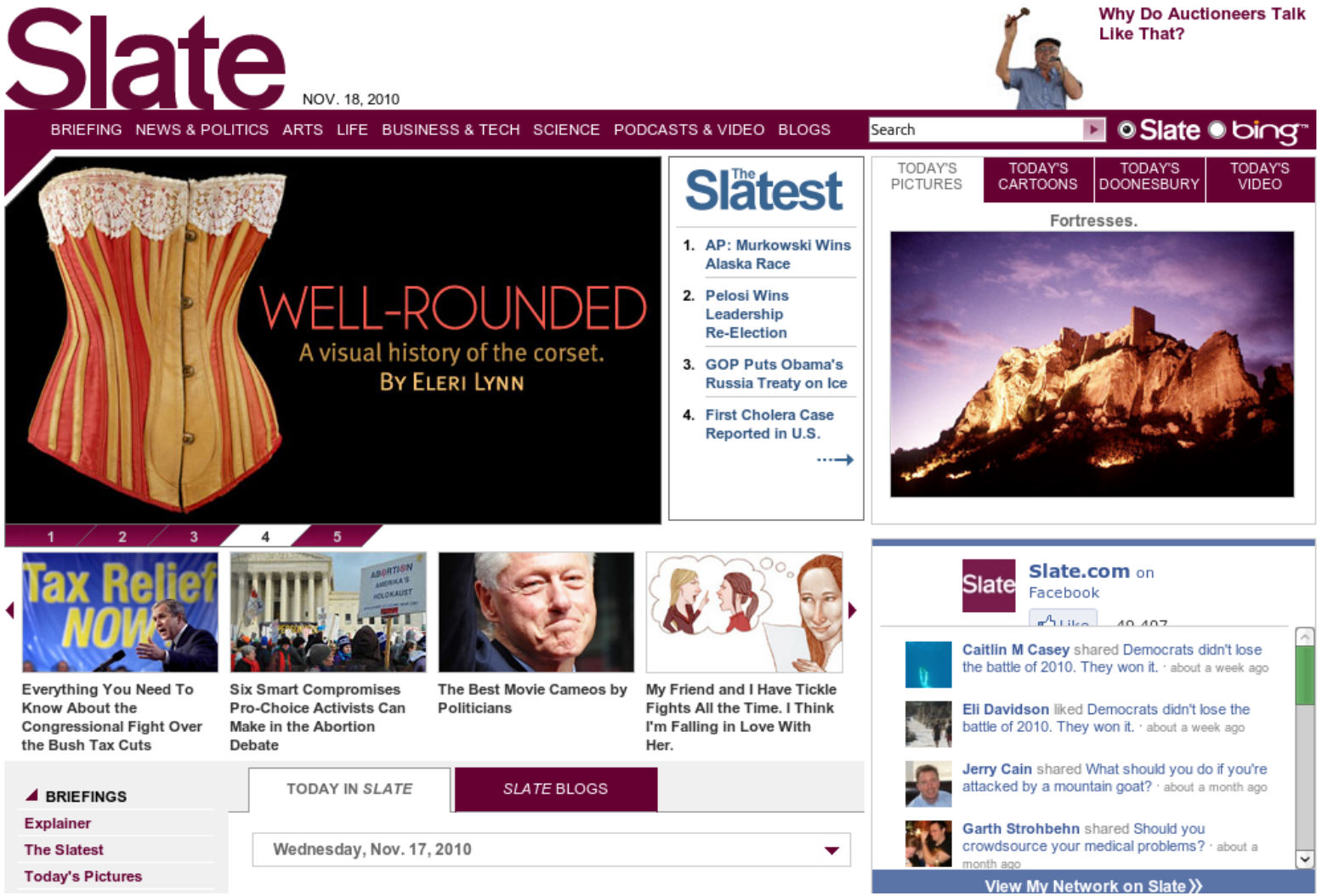
1. App registration
2. Access request
3. User approval
4. API Access

OAuth—Delegating API access

Open issues

1. Standardisation
2. Branding
3. Security level
4. Service discovery

Interaction via `iframe`



Preventing surrepititious authentication

```
<img id="test" style="display:none">

<script>
test = document.getElementById('test');
var start = new Date();
test.onerror = function()
{ time = new Date() - start;}

test.src = "http://www.example.com/";
</script>
```

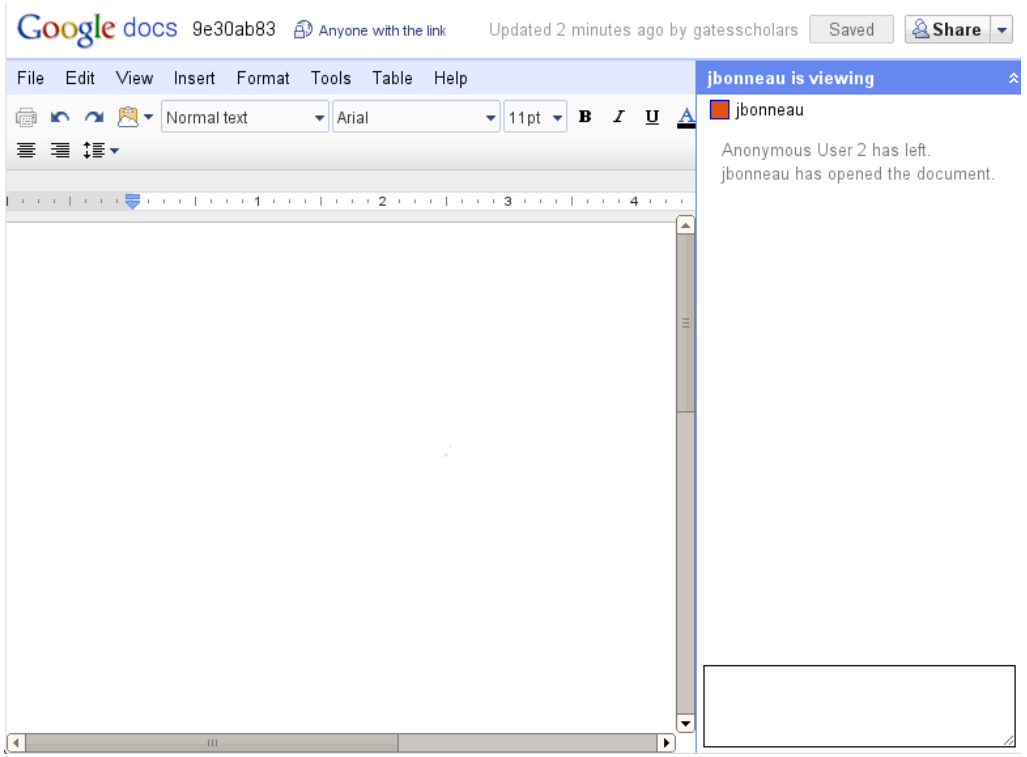
Bortz et al. 2007

Preventing surrepititious authentication

```
# Send users to my detector...
<iframe name="detector"
width="0" height="0" frameborder="0"
src="https://docs.google.com/document/d/
1TUV9x1lFAQcVWvhP4EAHQZIPrVmo3_vrz5Sz8Wo">
</iframe>
```

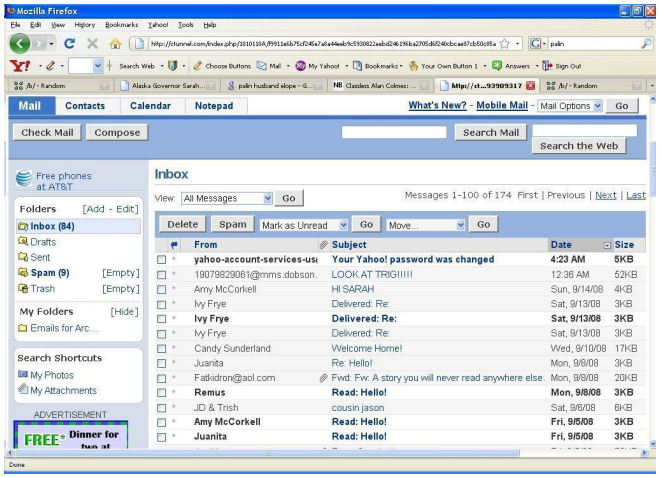
Narayanan 2009

Preventing surrepititious authentication



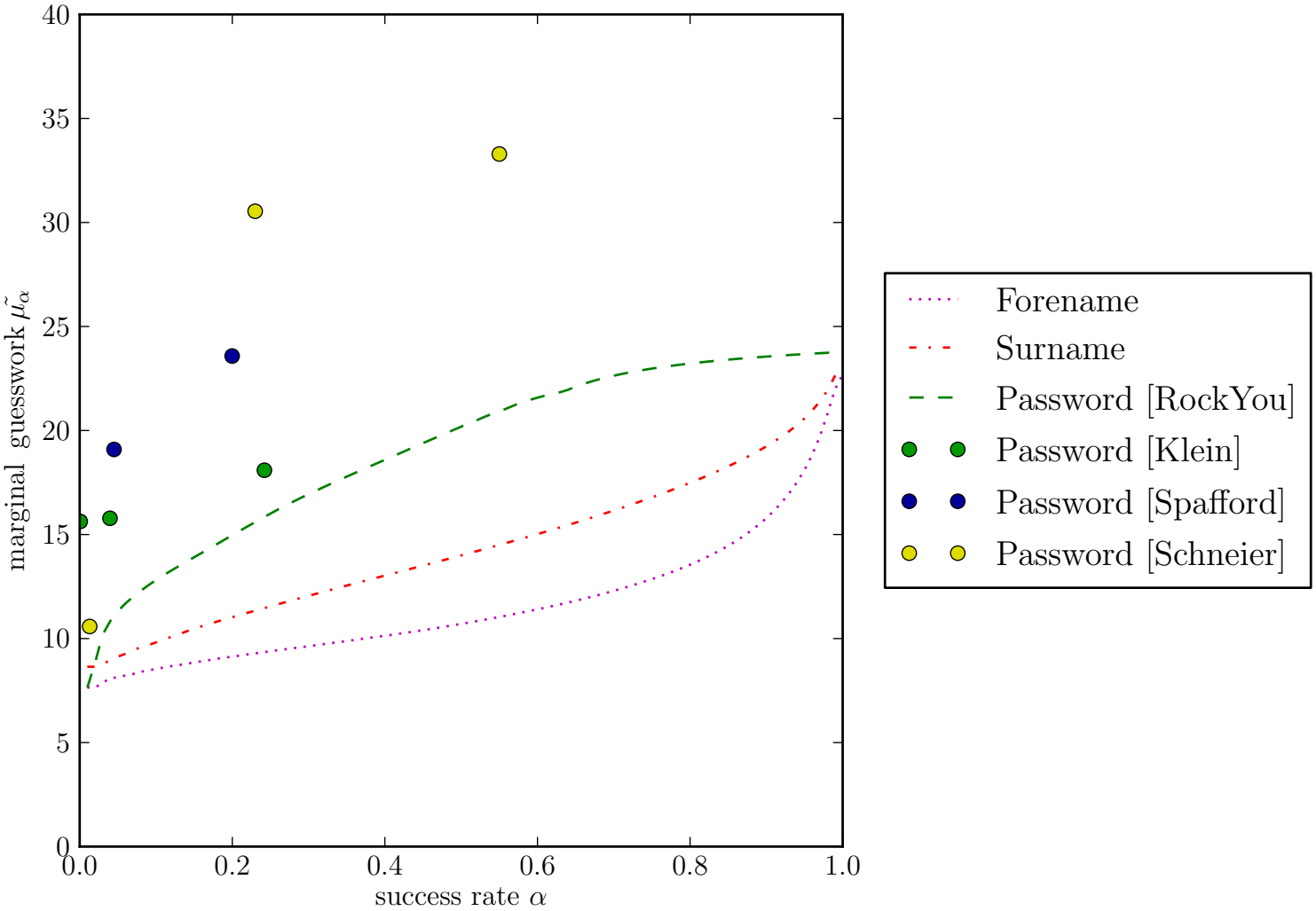
Narayanan 2009

Workable backup authentication



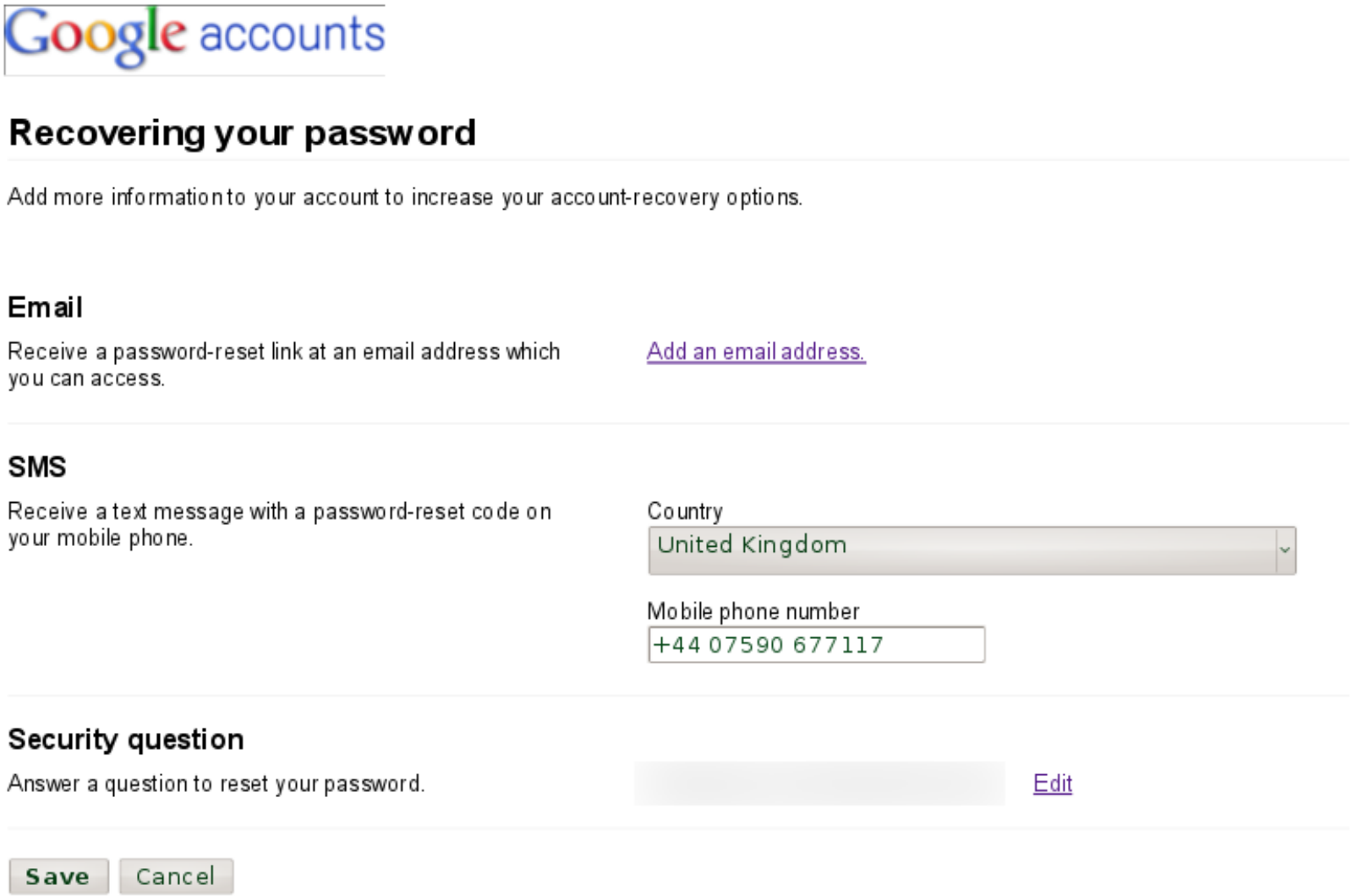
- Web search
  - Reaching a head with OSNs
- Public records
  - Griffith et. al: 30% of individual's mother's maiden names
- Social engineering
- Dumpster diving, burglary
- Acquaintance attacks
  - Schecter et. al: $\sim$ 25% of questions guessed by friends, family

Workable backup authentication



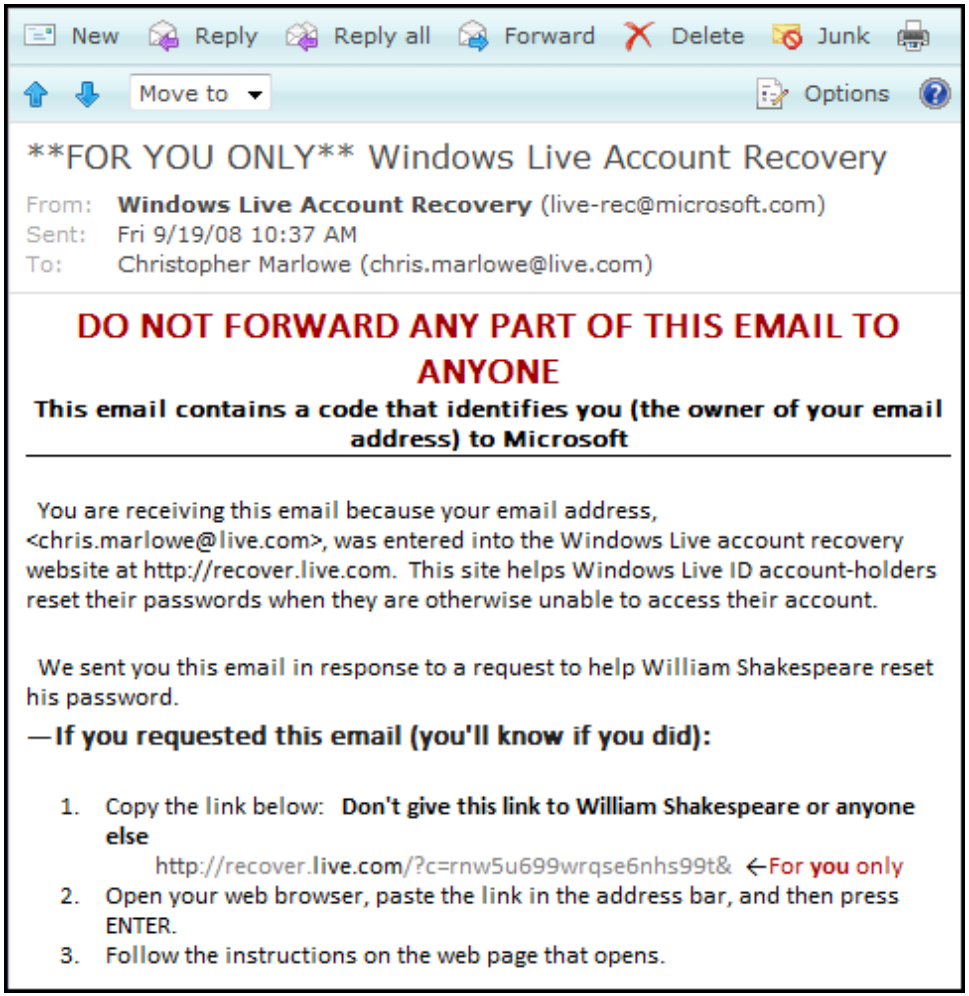
Personal knowledge worse than passwords (Bonneau et al. 2010)

Workable backup authentication



Google—backup authentication by mobile phone

Workable backup authentication



New Reply Reply all Forward Delete Junk

Move to Options

FOR YOU ONLY Windows Live Account Recovery

From: **Windows Live Account Recovery** (live-rec@microsoft.com)
Sent: Fri 9/19/08 10:37 AM
To: Christopher Marlowe (chris.marlowe@live.com)

DO NOT FORWARD ANY PART OF THIS EMAIL TO ANYONE

This email contains a code that identifies you (the owner of your email address) to Microsoft

You are receiving this email because your email address, <chris.marlowe@live.com>, was entered into the Windows Live account recovery website at http://recover.live.com. This site helps Windows Live ID account-holders reset their passwords when they are otherwise unable to access their account.

We sent you this email in response to a request to help William Shakespeare reset his password.

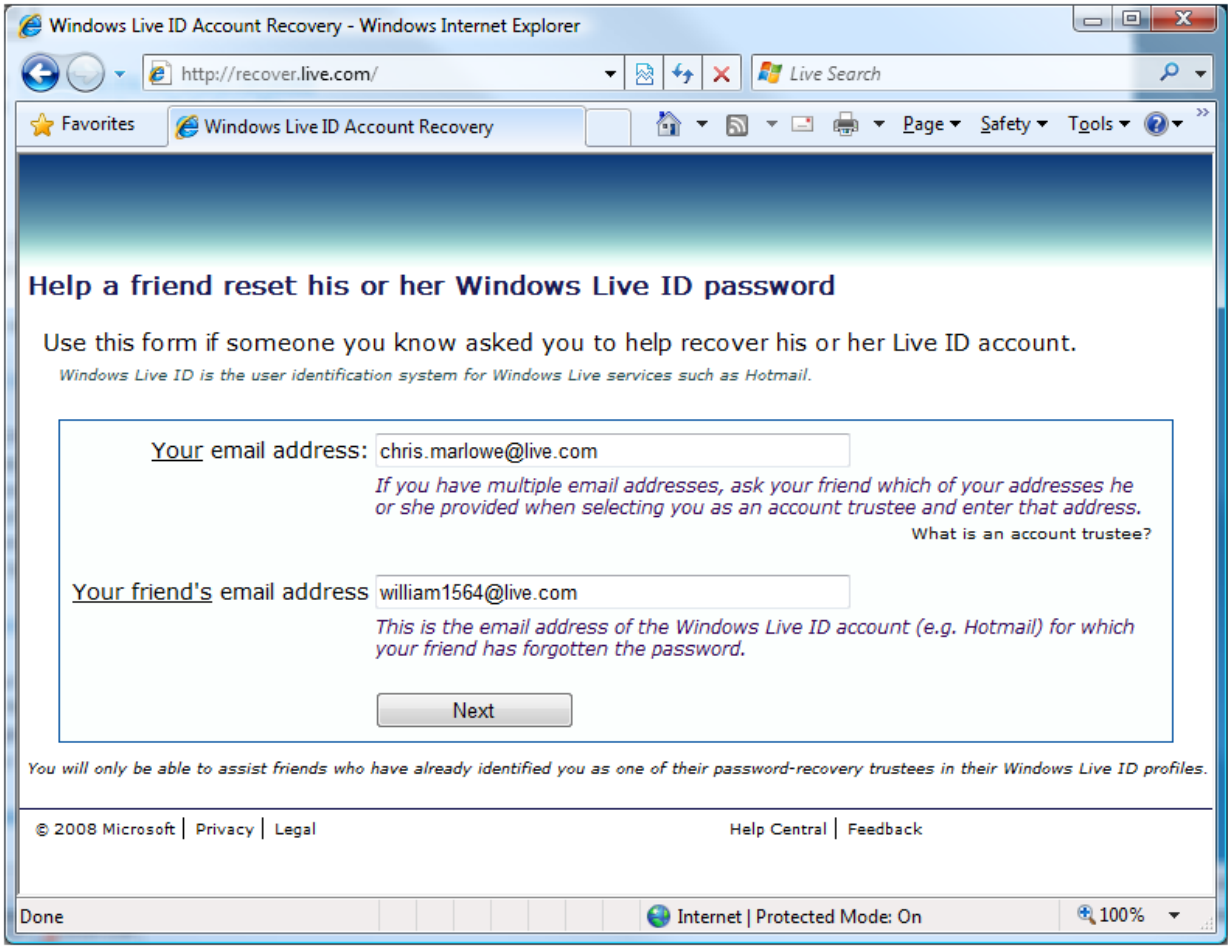
—If you requested this email (you'll know if you did):

1. Copy the link below: **Don't give this link to William Shakespeare or anyone else**
   http://recover.live.com/?c=rnw5u699wrqse6nhs99t& ←For **you** only
2. Open your web browser, paste the link in the address bar, and then press ENTER.
3. Follow the instructions on the web page that opens.

Schecther et al. 2008

MS Live (proposed)—social backup authentication

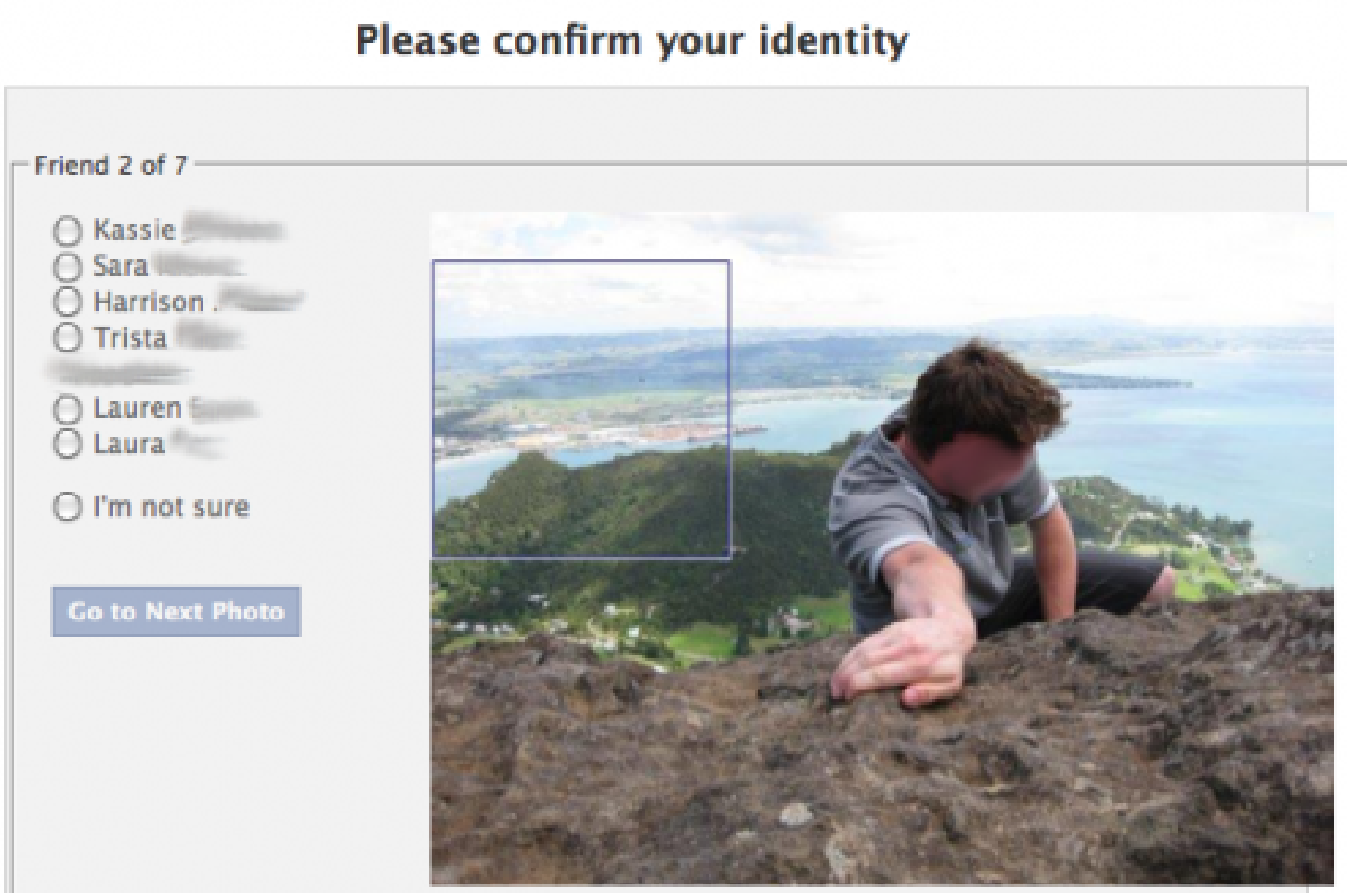
Workable backup authentication



Schecther et al. 2008

MS Live (proposed)—social backup authentication

Workable backup authentication



Facebook—social questions backup

Workable backup authentication

Account Activity

View your recent account activity. If you notice an unfamiliar device or location, click "end activity"

Note: Locations and device types reflect our best guesses based on your ISP or wireless carrier.

Most Recent Activity

Last Accessed: **Today at 3:12pm**
Location: Cambridge, ENG, GB *(Approximate)*
Device Type: Firefox on Linux

Also Active

Last Accessed: **Yesterday at 6:54pm** — end activity
Location: Cambridge, ENG, GB *(Approximate)*
Device Type: Mozilla/5.0 (X11; U; Linux i686; en-US) AppleWebKit/534.12 (KHTML, like Gecko) Ubuntu/9.10 Chromium/9.0.576.0 Chrome/9.0.576.0 Safari/534.12

Last Accessed: **November 1 at 2:12pm** — end activity
Location: London, ENG, GB *(Approximate)*
Device Type: Chrome on Win7

Last Accessed: **October 29 at 8:17pm** — end activity
Location: Cambridge, ENG, GB *(Approximate)*
Device Type: Mozilla/5.0 (X11; U; Linux i686; en-US) AppleWebKit/534.11 (KHTML, like Gecko) Ubuntu/9.10 Chromium/9.0.566.0 Chrome/9.0.566.0 Safari/534.11

Last Accessed: **October 24 at 1:26am** — end activity
Location: Cambridge, ENG, GB *(Approximate)*
Device Type: Firefox on Linux

Facebook—social questions backup

Questions

jcb82@cl.cam.ac.uk