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
“Identity” websites
"E-Commerce" websites
“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
<table>
<thead>
<tr>
<th>feature</th>
<th>scoring</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enrolment</strong></td>
<td></td>
</tr>
<tr>
<td>Password selection advice given</td>
<td>+1 pt</td>
</tr>
<tr>
<td>Minimum password length required</td>
<td>+1 pt</td>
</tr>
<tr>
<td>Dictionary words prohibited</td>
<td>+1 pt</td>
</tr>
<tr>
<td>Numbers or symbols required</td>
<td>+1 pt</td>
</tr>
<tr>
<td>User list protected from probing</td>
<td>+1 pt</td>
</tr>
<tr>
<td>Cleartext password sent in email after enrolment</td>
<td>−1 pt</td>
</tr>
<tr>
<td><strong>login</strong></td>
<td></td>
</tr>
<tr>
<td>Password hashed in-browser before POST</td>
<td>+1 pt</td>
</tr>
<tr>
<td>Limits placed on password guessing</td>
<td>+1 pt</td>
</tr>
<tr>
<td>User list protected from probing</td>
<td>+1 pt</td>
</tr>
<tr>
<td>Federated identity login accepted</td>
<td>+1 pt</td>
</tr>
<tr>
<td><strong>password update</strong></td>
<td></td>
</tr>
<tr>
<td>Password re-entry required to authorise update</td>
<td>+1 pt</td>
</tr>
<tr>
<td>Notification email sent after password reset</td>
<td>+1 pt</td>
</tr>
<tr>
<td><strong>password recovery</strong></td>
<td></td>
</tr>
<tr>
<td>Password update required after recovery</td>
<td>+1 pt</td>
</tr>
<tr>
<td>Cleartext password sent in email upon request</td>
<td>−1 pt</td>
</tr>
<tr>
<td>User list protected from probing</td>
<td>+1 pt</td>
</tr>
<tr>
<td><strong>encryption</strong></td>
<td></td>
</tr>
<tr>
<td>Full TLS for all password submission</td>
<td>+2 pts</td>
</tr>
<tr>
<td>POST only TLS for password submission</td>
<td>+1 pt</td>
</tr>
</tbody>
</table>
The realities of web authentication

![Graph showing frequency of password collection.](image)

**Frequency of password collection**

- Proportion of sites collecting passwords appears to decrease as traffic rank increases.
- The graph suggests that a larger number of sites collect passwords from lower traffic rank sites compared to higher traffic rank sites.
The realities of web authentication

- All websites collect email address as username
- All websites use email for password reset
- 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

RockYou, which provides widgets popular with MySpace and Facebook users, has been hacked and 32.6m users are being urged to change their passwords.

RockYou SQL injection hack
January 2010
Many schoolbook errors are quite common

<table>
<thead>
<tr>
<th>countermeasure</th>
<th>I</th>
<th>E</th>
<th>C</th>
<th>Tot.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTCHA</td>
<td>11</td>
<td>2</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>timeout</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>reset</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>none</td>
<td>37</td>
<td>43</td>
<td>46</td>
<td>126</td>
</tr>
</tbody>
</table>

Many websites allow unlimited brute-force guessing
Many schoolbook errors are quite common

User probing is rarely prevented
Many schoolbook errors are quite common

<table>
<thead>
<tr>
<th>interface</th>
<th>I</th>
<th>E</th>
<th>C</th>
<th>Tot.</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrolment</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>login</td>
<td>43</td>
<td>41</td>
<td>38</td>
<td>132</td>
</tr>
<tr>
<td>reset</td>
<td>11</td>
<td>7</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>all</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

User probing is rarely prevented
Many schoolbook errors are quite common.
Many schoolbook errors are quite common

<table>
<thead>
<tr>
<th>TLS Deployment</th>
<th>I</th>
<th>E</th>
<th>C</th>
<th>Tot.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>10</td>
<td>39</td>
<td>10</td>
<td>59</td>
</tr>
<tr>
<td>Full/POST</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Inconsistent</td>
<td>14</td>
<td>6</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>None</td>
<td>23</td>
<td>4</td>
<td>34</td>
<td>61</td>
</tr>
</tbody>
</table>

TLS deployment remains uneven, poorly done
Many schoolbook errors are quite common.
Security policies vary far more than requirements.
More popular sites do better
Economic failures

Twitter accounts compromised in torrent site scam

Angela Moscatiolo  February 03, 2010

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.

- 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.)
- **UE** End user (a human)
- **UA** User agent (a browser)

\[ UE \rightarrow R \quad \text{I'm U@P!} \]
Registering for Mixx is fast, fun, and easy! Here at Mixx, we don't think you should have to create yet another username and password. We work with several sites that you may already use. Simply select the account you'd like your new Mixx account to work with and we'll handle the rest!

Register using your OpenID URL

[Image of OpenID logo]
OpenID—Single sign-on

- **R**: Relying party (www.example.com)
- **P**: OpenID Provider (Facebook, Google, etc.)
- **U_E**: End user (a human)
- **U_A**: User agent (a browser)

\[
U_E \rightarrow R \quad \text{I'm } U@P! \\
R \leftrightarrow P \quad K_{R-P}, n \leftrightarrow \text{D-H key exchange}
\]
OpenID—Single sign-on

R  Relying party (www.example.com)
P  OpenID Provider (Facebook, Google, etc.)
UE  End user (a human)
UA  User agent (a browser)

UE  →  R  I’m U@P!
R  ←→  P  $K_{R-P}, n \leftarrow$ D-H key exchange
UE  ←  R  OK, go verify with P (HTTP 302)
UE  →  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.)
UE  End user (a human)
UA  User agent (a browser)

UE  →  R  I’m U@P!
R  ←→  P  $K_{R-P}, n \leftarrow$ D-H key exchange
UE  ←  R  OK, go verify with P (HTTP 302)
UE  →  P  I want to talk to R, who you share n with
UE  ←  P  Sure you want to talk to R?
OpenID—Single sign-on

You are signing in to Mixx.com with your Google Account jbonneau@gmail.com

Sign in  Cancel

Remember me

You can always change your Google Account approval settings. Mixx.com is not owned, operated or controlled by Google or its owners. Learn more
OpenID—Single sign-on

- **R**: Relying party (www.example.com)
- **P**: OpenID Provider (Facebook, Google, etc.)
- **UE**: End user (a human)
- **UA**: User agent (a browser)

** protocol flows:

- **UE** → **R** I’m **U@P**!
- **R** ↔ **P** $K_{R-P}, n \leftarrow$ D-H key exchange
- **UE** ← **R** OK, go verify with **P** (HTTP 302)
- **UE** → **P** I want to talk to **R**, who you share $n$ with
- **UE** ← **P** Sure you want to talk to **R**?
- **UE** → **P** Yes, here’s my password: $p$
OpenID—Single sign-on

R  Relying party (www.example.com)
P  OpenID Provider (Facebook, Google, etc.)
UE  End user (a human)
UA  User agent (a browser)

UE  →  R  I’m U@P!
R  ←→  P  $K_{R-P}, n \leftarrow$ D-H key exchange
UE  ←  R  OK, go verify with P (HTTP 302)
UE  →  P  I want to talk to R, who you share n with
UE  ←  P  Sure you want to talk to R?
UE  →  P  Yes, here’s my password: $p$
UE  ←  P  Okay, use $MAC_{K_{R-P}}(U, P)$ (HTTP 302)
UE  →  R  $MAC_{K_{R-P}}(U, P)$! See, I’m U@P
OpenID—Single sign-on

Feeling geeky?

When you log in to a website that supports OpenID login we’ll send your OpenID identifier to the website so it can identify you.

To make things easy, we have generated this identifier for you:
https://me.yahoo.com/a/OU2iCjRytdHt3TZVle...

You don’t need to save this identifier. While logging in to websites, you can simply look for a Yahoo! button or type yahoo.com in the OpenID text field. You can also choose additional custom identifiers for your Yahoo! account below.
OpenID—Single sign-on

- **R** Relying party (www.example.com)
- **P** OpenID Provider (Facebook, Google, etc.)
- **UE** End user (a human)
- **UA** User agent (a browser)

**UE** → **R** I’m **U@P**!

**R** ↔ **P** $K_{R-P}, n \leftarrow$ D-H key exchange

**UA** ← **R** OK, go verify with **P** (HTTP 302)

**UA** → **P** I want to talk to **R**, here’s my cookie $c$

**UA** ← **P** Okay, use $\text{MAC}_{K_{R-P}}(U, P)$

**UA** → **R** $\text{MAC}_{K_{R-P}}(U, P)$! See, I’m **U@P**

(auth-immediate)
The Dark Ages

Find people you know on Facebook
Your friends on Facebook are the same friends, acquaintances and family members that you communicate with in the real world. You can use any of the tools on this page to find more friends.

Find People You Email
Searching your email account is the fastest way to find your friends on Facebook.

- Your Email: [redacted]@gmail.com
- Email Password:

Find Friends

Facebook will not store your password. Learn More.
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

**Application**

1. **App registration**
   - Sign up to get a Consumer Key (API Key)
2. **Access request**
   - Get a Request Token using `get_request_token`
   - Request parameters: `oauth_consumer_key`, `oauth_nonce`, `oauth_signature_method`, `oauth_timestamp`, `oauth_version`, `oauth_lang` (optional), `oauth_callback`
3. **User approval**
   - Yahoo! prompts user to provide authorization
   - User authorizes access to private data
4. **API Access**
   - Yahoo! redirects user to application, passing `oauth_verifier`
   - Exchange the Request Token and OAuth Verifier for an Access Token using `get_token`
   - Request parameters: `oauth_consumer_key`, `oauth_signature_method`, `oauth_timestamp`, `oauth_version`, `oauth_nonce`, `oauth_token`, `oauth_token_secret`, `oauth_callback`, `oauth_verifier`

**Yahoo!**

- Yahoo! provides a Consumer Key and a Shared Secret
- Yahoo! returns the Request Token as response to `get_request_token`
- Response parameters: `oauth_token`, `oauth_token_secret`, `oauth_expired_in`, `oauth_request_token_url`, `oauth_callback_confirmed`

**User**

- Use `oauth_token` in Yahoo! API Requests until Access Token expires
- Refresh the Access Token using `get_token`
- Request parameters: `oauth_consumer_key`, `oauth_signature_method`, `oauth_timestamp`, `oauth_version`, `oauth_token`, `oauth_nonce`

---

**Basic Information**

- **Application Name**: Test
- **Description**: (blank)
- **Icon**: Change your icon
- **Logo**: Change your logo
- **Language**: English (US)
- **User Support Address**: Email or URL
- **Contact Email**: Jbonneau@gmail.com
- **Privacy Policy URL**: (blank)
- **Terms of Service URL**: (blank)

**Developers**

- **Developers**: Joseph Bonneau
- **Add Developers**: Start typing a friend's name

---

**Steps**

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

---

J. Bonneau (U. of Cambridge)  
SOCIALNETS  
November 18, 2010
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

Request for Permission

FarmVille is requesting permission to do the following:

- Access my basic information
  Includes name, profile picture, gender, networks, user ID, list of friends, and any other information I've shared with everyone.
- Access my profile information
  Birthday and Current City

By proceeding, you agree to the FarmVille Terms of Service and Privacy Policy - Report Application
**OAuth—Delegating API access**

**PLAINTEXT:**

\[ M | K_{app} | K_{user} \]

**HMAC_SHA1:**

\[ MAC_{K_{app}}(M) \]

**RSA_SHA1:**

\[ Sign_{K_{app}}(M) \]

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

J. Bonneau (U. of Cambridge)
SOCIALNETS
November 18, 2010 7 / 10
Preventing surrepititious authentication

```html
<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
# Send users to my detector...
<iframe name="detector"
width="0" height="0" frameborder="0"
src="https://docs.google.com/document/d/1TUV9x11FAQcVWvhP4EAHQZIPrVmo3_vrz5Sz8Wo">
</iframe>
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)
Recovering your password

Add more information to your account to increase your account-recovery options.

Email
Receive a password-reset link at an email address which you can access. Add an email address.

SMS
Receive a text message with a password-reset code on your mobile phone.

Country
United Kingdom

Mobile phone number
+44 07590 677117

Security question
Answer a question to reset your password. Edit

Save  Cancel
Workable backup authentication

Figure 2. Trustee-authentication email. This email contains a link that identifies the trustee to our website.

discourage trustees from responding to requests for account-recovery codes that arrive via email or text messages (they are easy to spoof), we also discourage account holders from contacting their trustees using these channels.

We were not sure how many account-recovery codes should be required to authenticate an account holder. We configured the system to require a threshold of three codes so that we could measure the time required to obtain both the second and third code. To obtain an account-recovery code, a trustee must perform four steps.

**Initiation**
When the trustee first visits the account recovery system, she is asked to enter her email address and the address of the account holder she is assisting (Figure 1).

Next, the trustee receives an email from the account recovery system (Figure 2). If she is indeed a trustee for the specified account holder, the system creates a record to track the request and the email sent to the trustee will contain a code pointing to this record. The trustee copies this link into her browser's address bar to continue.

This emailed link and code are all that are required to prove the trustee's identity and retrieve the account-recovery code. An attacker who could convince a trustee to forward the email would be able to retrieve the code. Two countermeasures against this attack are the email's subject, which begins with "**FOR YOU ONLY**", and the message body, which begins with a conspicuous warning "do not forward any part of this email to anyone" (see Figure 2).

**Query of intent**
When the trustee pastes the link from the trustee-authentication email into her browser, she is asked to explain why she is requesting an account-recovery code by choosing from a set of options, illustrated in Figure 3. These options may convey that she has heard from the account holder personally or that she is responding to a request from a third party.

The options that indicate the highest risk of fraud are listed at the top in order to maximize the chance that the trustee will read them before making a choice. If the trustee chooses either of the top two options, she encounters a warning page that describes telltale signs of fraud and encourages her to contact the account holder by phone or in person. She is, however, given the option to disregard these warnings and continue.

**Pledge**
Finally, the trustee is asked to pledge to her previous answer and to her understanding of the potential consequences of giving an account-recovery code to someone other than the account holder. This pledge requires her to type her name, as provided by the account holder, and to press a button that says "I promise the above pledge is true". For example, if a trustee reports receiving a request from the account holder via voicemail, she would be asked to pledge that she will only provide a code after she reaches him "in person", as illustrated in Figure 4.

After the trustee has signed the pledge, the system presents the six character account-recovery code. If this is the first account-recovery code requested for this account holder, the system will then email the remaining trustees to notify them of the event and encourage them to call the account holder.

To further protect against attack, the account holder will be notified whenever he next logs in (or if he is already online). If an attack were underway, a call from his trustees would alert the account holder to login and halt the recovery process before the attacker can complete it.

Schecther et al. 2008

MS Live (proposed)—social backup authentication
Workable backup authentication

The ubiquity of mobile phones has made them an attractive option for authentication. Because we want to avoid having users change their security behavior, we focused on the inherent strengths and weaknesses of mobile phones rather than attempting to exploit them. The primary threat to a social authentication system is that an attacker – someone other than the account holder – will attempt to log into the victim's account and will then try to convince or trick the account holder's trustees to vouch that the attacker is the account holder. That is, the attacker would convince or trick the account holder's trustees to vouch that he or she is acting on behalf of the victim. When an account holder needs to recover his account, he must obtain account-recovery codes from his trustees. Account-recovery codes are sent to the account holder via email to a pre-selected trustee they called a “helper” [1]. In this section we provide an overview of the system, user experience, and countermeasures to defend against attacks.

In 2006, Brainard et al. of RSA proposed a two-factor primary authentication system (PIN and token) for enterprise organizations, the responsibility to authenticate a user who fails primary authentication is often shifted to system administrators, corporate security, or other support staff. Microsoft has long employed a form of trustee-based account recovery in which a user who lost her token could receive help from a pre-selected trustee they called a “helper” [1]. In 2006, Brainard et al. of RSA proposed a two-factor primary authentication system (PIN and token) for enterprise organizations, the responsibility to authenticate a user who fails primary authentication is often shifted to system administrators, corporate security, or other support staff. Microsoft has long employed a form of trustee-based account recovery in which a user who lost her token could receive help from a pre-selected trustee they called a “helper” [1].

Schecther et al. 2008

MS Live (proposed)—social backup authentication

Schechter et al. 2008

MS Live (proposed)—social backup authentication
Facebook—social questions backup
Workable backup authentication

Facebook—social questions backup
jcb82@cl.cam.ac.uk