Stronger Password Authentication Using Browser Extensions

Presented at 14th USENIX Security Symposium, July 31 – August 5, 2005. Baltimore, MD

Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, John Mitchell (Stanford University)

By John Leggio
Tuesday, October 17, 2006
ECE 1776
Objectives

- Enhance web password security authentication.
- Provide transparent method for customizing “per site passwords”
- No server changes & little or no change to user experience
- Want to reduce password attack threat
Result: PwdHash

- Allows users to invisibly generate site specific passwords
- Lightweight browser extension
  - Invisible to server
  - Invisible to user
- Helps stop password theft
- Easy to use
How PwdHash Works

- Generates unique per site passwords
- Uses password, domain name of site (the "salt") & Pseudo Random Function (PRF):
  \[
  hash(password, domain) = PRF_{password}(domain)
  \]
- Implemented with Firefox extension & IE Browser Helper Object
- Uses the MD5 cryptographic hash function
How PwdHash Works (Continued)

<table>
<thead>
<tr>
<th>Uses 2 different approaches to validate user experience:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• password-prefix &amp; password-key</td>
</tr>
<tr>
<td>• Automatically generates per site passwords by prefixing password with @@ or pressing F2 key</td>
</tr>
<tr>
<td>• Visual enhancement identifies when password mode is on</td>
</tr>
</tbody>
</table>

![Traffic light diagram]
PwdHash In Use:

You typed the PwdHash password prefix, but you are not currently in a password field that starts with the password prefix.

It is possible, though unlikely, that the site trying to steal your password.

Do not enter your PwdHash password into this page.
Why Use PwdHash?

- Most commercial websites have weak form of password authentication
- Combats password phishing problem
- Protection mechanism “built in” to per site password
- User decision not needed
- Same prefix works for everyone
- Distinguishes secure passwords from normal passwords & PINs
Challenges

- Password Reset
- Public computers
- Dictionary Attacks
- Spyware, DNS poisoning (no protection)
- Encoding hashed password
- Additional attacks and defenses

(Source: http://crypto.stanford.edu/PwdHash/)
# Pros

<table>
<thead>
<tr>
<th>Paper</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Problem thoroughly defined &amp; analyzed</td>
<td>• Usable &amp; unobtrusive technique</td>
</tr>
<tr>
<td>• Contains useful user study of PwdHash</td>
<td>• Good for phishing scams</td>
</tr>
<tr>
<td>• Related work that complements PwdHash</td>
<td>• Simple &amp; elegant solution</td>
</tr>
<tr>
<td>• Javascript attacks well defined</td>
<td>• Client-side security solution</td>
</tr>
<tr>
<td></td>
<td>• Useful tool</td>
</tr>
</tbody>
</table>
Cons

• Focus of the paper was implementation challenges
• Domain name changes?
• How it would manage access to a webpage using the IP address?
• Roaming
• Not sure if user study is generalizable.
• User study could have been defined better
• Traffic light maybe spoofed
Cons

- How it would handle length of password at different sites?
- People need to first reset their passwords.
- Having a remote website doing hashing for you is troublesome and insecure.
- Interactive applications using Ajax may communicate to the server the password character by character.
- Users may be feel uncomfortable with not knowing actual passwd
Cons

• Details of the implementation of the config file have not been revealed
• Inconvenience of using different passwords at sister websites like gmail.com and google.com
• Expecting the user to always type @@ before typing the password. It might even appear in the keystream
• User experience study was shallow as presumably a lot of learning and annoyance would be introduced as a result of ‘@@’, change of length of passwd, roaming, password reset.
Related Work


