HotSec08 Presentation – July 29, 2008

Digital Objects as Passwords

Mohammad Mannan and P.C. van Oorschot
mmannan@scs.carleton.ca

Carleton University, Canada
The fun of password generation
Use random generators?

![Advanced Password Generator](image)
What we focus on

1. Usable strong password
   - password generation
   - password recall

2. Infrequently-used password
   - Personal Verification Questions (PVQs)
   - tax filing password

“easy to remember = easy to guess”
Your object is your password: **ObPwd**

(a) Generic steps in ObPwd

1. User selected content (image, text, binary)
2. Apply hash function
3. Hashed value
4. Hash2Text
5. Password

(b) An example of ObPwd

1. SHA-1 (base64 output)
2. XLVe1DSkCHEEWA2qhK6QSwOJXA
3. PwdHash encoding
4. e1DSkCHEERXLV
Password objects

1. Object features
   - personal or personally meaningful
   - stable (long-lived) content

2. Object sources
   - private objects: inaccessibility
   - web objects: vast richness
Password objects (cont.)

1. Private objects
   - local disk, mobile media (USB stick)
   - images, documents, text passages, executables, emails

2. Web/public objects
   - Internet Archive, Project Gutenberg, Google Books, ACM/IEEE digital archive
   - images, text passages, files
ObPwd variants

1. Append a salt with the selected object
   ▶️ $pwd = \text{Hash2Text}( \text{Hash}(\text{object, salt}) )$
   ▶️ harder to generate password from compromised objects

2. Append a URL
   ▶️ $pwd = \text{Hash2Text}( \text{Hash}(\text{object, URL}) )$
   ▶️ may prevent password phishing (cf. PwdHash)

Better protection but ... usability, portability?
Prototype implementations

1. Firefox add-on (cross platform, web objects)
2. Windows XP application (local objects)
3. Linux/Mac command-line program (local objects)
Prototype implementations

ObPwd extension menu in Firefox

Password generated from the selected image

ObPwd Win32 application
Implementation choices

1. PwdHash encoding as Hash2Text
   - 12 characters, alphanumeric
   - omit special character option

2. Min. object size $= 30$ bytes, truncate at: $100,000$ bytes
Limitations

1. Shoulder surfing

2. Obvious public objects
   - Facebook profile photo

3. Password objects visible to network attacker
   - mostly affects web login (use Tor?)

4. Interference: passwords from different objects

5. Rootkits 😞
Related ideas

1. TrueCrypt allows files as an encryption key
   - resulting key isn’t exposed to users

2. Photos as PVQs (Ariel Rabkin, SOUPS 2008)
   - upload a selected photo to an authenticating site
   - answer “who is the person in the photo?”
Some benefits

1. Reduced memory load: remember only a hint

2. Generating global password dictionary seems difficult
   - dictionaries for regular and passphrase/mnemonic password exist

3. Written backup: not feasible for graphical passwords
   - middle ground between text and image based schemes
   - rich selection space: human seeded attacks are harder

4. Password sharing through hints
   - better than email password sharing?
Open issues

1. Is ObPwd a usable technique to generate strong password?
   - user testing required

2. Can we expose more options to users without confusing them?
   - password length, special chars, look-alike chars (1, l, 0, O)

3. How to deal with site-specific password requirements?

Try from:

http://www.ccs1.carleton.ca/~mmannan/obpwd