Imperial College London

CO331 – Network and Web Security

5. Passwords

Dr Sergio Maffeis
Department of Computing
Course web page: http://www.doc.ic.ac.uk/~maffeis/331

Computer passwords

Imperial College London

- Main applications
  - Protection of cryptographic keys
    - Unlock keys for SSH, PGP, etc
    - Access encrypted files or disks
  - User authentication
    - On a machine, on a website, on an application
- Password-based authentication
  - Passwords are an intuitive way to authenticate
  - The technology is well-understood
    - Easy to implement and to deploy
  - Passwords are proven in the field
    - If you agree that the Internet works well enough
  - We'll discuss some limitations

Plain-text passwords

Imperial College London

1. Store credentials in a password file:

```
alice:wonderland
bob:builder
charlie:brown
…
```

   – Linux: file used to be `/etc/passwd`, now `/etc/shadow`
2. User presents username and password
3. Check if username is present, and if so
4. Check that presented password matches stored one
5. Grant or deny access

- Password file is a very valuable target for hackers
  – Can impersonate any user in the system

Encrypted passwords

Imperial College London

- Symmetric encryption
  - Encrypt(key,plaintext) = ciphertext
  - Decrypt(key,ciphertext) = plaintext
  - Example
    - Encrypt(key,“wonderland”)=`F4653BB4ACB0F3E`
    - Decrypt(key,“`F4653BB4ACB0F3E`”)=wonderland

1. Store **encrypted** credentials in a password file:

```
alice:F4653BB4ACB0F3E
  bob:DF7E258D59B5BBD
charlie:52885B2B72EADC3
...
```

4. Check that presented password matches decryption of stored password

- Steps 2,3,5 as before
- Key becomes another valuable target for hackers
  - Situation hasn’t changed that much
  - Key management issues: cannot store in the password file itself

Password hashes

Imperial College London

- Cryptographic hashing
  - Hash(plaintext) = hashvalue
  - Theoretically a one-way function: cannot be reversed
  - In practice very hard to find a collision Hash(x) = Hash(y) where x ≠ y
    - Depends on computing power: SHA-1 and MD5 now deprecated, better SHA-256, SHA-512

1. Store **hashed** credentials in a password file:

```
alice:8921FD3A711D2ED
  bob:22D34F1D7EA9937
charlie:EFDAC1E36B1E815
      …
```

4. Check that presented password is correct
   - Apply Hash() to password provided by user
   - Look for result in the table

- Steps 2,3,5 as before
- Password file still a valuable target
  - *Offline dictionary attack*
    - Attacker builds, once and for all, a large database table of common passwords, hash values
    - Look for a stolen hash in the database: if present, you know the original password

Salted hashes

Imperial College London

- Salted hashing
  - Salt: a cryptographically random string
    - Picked at random, and looks random: not "0000000000000"
  - Salted hash: Hash(plaintext|salt) = hashvalue

1. Store **salted hashes** of credentials in a password file
   - Format: username:**salt**:salted_hashed_password

```
alice:61C82:5C0E35473DA573EAE74B5A
  bob:8B4D8:C92A77164142EC14DC2F67
charlie:D9103:2D64320A38D8DE877AA1BD
…
```

4. Check that presented password is correct
   - Compute Hash(password|user_salt), check that it matches stored entry

- Steps 2,3,5 as before
- Password file still a valuable target, but less so
  - Impractical to run a generic offline dictionary attack: different dictionary for every possible salt
  - Offline dictionary attack against one specific user is still practical
    - Given salt for target user, build a targeted dictionary
    - But no benefit of sharing dictionaries among attackers

Linux password file

Imperial College London

username:*password-data*:*parameters*

- Password data

  `$hash-function-id$salt$password`

  `*` = disabled

- Parameters

  Days since last change

  : `0` = can be changed at any time

  : `99999` = doesn't have to be changed

  : `7` = warn 1 week before expiry

  : …

```
root:$6$OswkkQDM$KGPLIQ4vIo4dkaHMorxWRJ
daemon:*:16826:0:99999:7:::
bin:*:16826:0:99999:7:::
sys:*:16826:0:99999:7:::
sync:*:16826:0:99999:7:::
games:*:16826:0:99999:7:::
man:*:16826:0:99999:7:::
lp:*:16826:0:99999:7:::
mail:*:16826:0:99999:7:::
news:*:16826:0:99999:7:::
uucp:*:16826:0:99999:7:::
proxy:*:16826:0:99999:7:::
www-data:*:16826:0:99999:7:::
backup:*:16826:0:99999:7:::
list:*:16826:0:99999:7:::
irc:*:16826:0:99999:7:::
gnats:*:16826:0:99999:7:::
nobody:*:16826:0:99999:7:::
systemd-timesync:*:16826:0:99999:7:::
systemd-network:*:16826:0:99999:7:::
systemd-resolve:*:16826:0:99999:7:::
systemd-bus-proxy:*:16826:0:99999:7:::
Debian-exim:!:16826:0:99999:7:::
messagebus:*:16826:0:99999:7:::
statd:*:16826:0:99999:7:::
csn:$6$xO2PhNvy$qyEyduxy2clticxpdH/nG0r
sshd:*:16827:0:99999:7:::
user001:$1$vYWKS/SW$83ske/x/gL516tJ/PXE
user002:$1$TMVfmM8s$M.LqPgsDxdDmdKUnKiC
user003:$1$DtEyT8j1$uDayB66kVru6jesocXe
user004:$1$gufEv/bi$ROjUVqfJ52sBpNTSSUM
user005:$1$6kqAZlKj$NdlScxR7WYbBx48GSry
user006:$1$yVhBC/YC$wb0K8isudOg5IzOKkbe
```

Online dictionary attack

Imperial College London

- Attacker tries username/password combinations on a running system
- Usernames are relatively easy to find
  - May be email addresses, may appear in blogposts, may be people's surnames, etc
- Passwords
  - Lists of common passwords
    - *"17,000 customers of UK financial services company have been using the password* `Arsenal1`*"* (ft.com, 23/1/15)
  - Passwords previously used on hacked websites
    - In the public domain, or purchased on the dark web
    - Users choose same password across sites (bad idea)
- Defenses
  - Limit numbers of tries per username or per IP before blocking access
  - Use CAPTCHAs (but it inconveniences legitimate users)
  - *Honeypot* passwords: create fake, easy to guess accounts and alert if accessed

Usability

Imperial College London

*"Choose a password you **can't** remember, and **don't** write it down"*

- Hard for humans to choose and remember good passwords
  - At some point BT had 100+ employees dedicated to password reset
- Security questions are dangerous
  - Can be found via social media, other online footprints
- "Password hints" functionality should be avoided
  - Adobe hack (2013) leaked 3M encrypted (!) passwords and security questions
  - People tend to choose reminders that give away password too easily

Leaked password “hints”

Imperial College London

Adobe password data		Password hint	
110edf2294fb8bf4	->	numbers 123456	❶ 123456
110edf2294fb8bf4	->	==123456	
110edf2294fb8bf4	->	c'est "123456"	
8fda7e1f0b56593f e2a311ba09ab4707	->	numbers	❷ 12345678
8fda7e1f0b56593f e2a311ba09ab4707	->	1-8	
8fda7e1f0b56593f e2a311ba09ab4707	->	8digit	
2fca9b003de39778 e2a311ba09ab4707	->	the password is password	❸ password
2fca9b003de39778 e2a311ba09ab4707	->	password	
2fca9b003de39778 e2a311ba09ab4707	->	rhymes with assword	
e5d8efed9088db0b	->	q w e r t y	❹ qwerty
e5d8efed9088db0b	->	ytrewq tagurpidi	
e5d8efed9088db0b	->	6 long qwert	
ecba98cca55eabc2	->	sixxone	❺ 111111
ecba98cca55eabc2	->	1*6	
ecba98cca55eabc2	->	sixones	

Alternatives to passwords

Imperial College London

- Hardware tokens from banks, one-time-password booklets from governments
  - Expensive
  - Hard to replace
- 2nd factor authentication via mobile phone
  - SMS 2nd factor now deprecated by NIST
- Biometric authentication
  - Impossible to replace once “lost”
  - Not that hard to “steal”, spoof
- Authentication via RFID tags
  - Risk or theft, misplacement
  - Proximity attacks

Best-practices

Imperial College London

- Use filters to ensure user selects long enough, random looking password
- Don't ask user to change password often
  - Better to have a strong, long-lived password than tempt users to choose easy-to-remember, shorter ones
- Store **salted hashes** of passwords in a protected file
  - Even more: salt >= 32 bits, keyed HMAC hash using SHA-1/2/3, >=10,000 rounds of PBKDF2 to "stretch" the key.
  - Don't fail with "User not found": attacker can learn about valid users
- After few failed login attempts for same username or from same IP
  - Ask an additional security question (*2nd factor authentication*)
    - Not via SMS! (risk of SS7 hack)
  - Slow down attempts: introduce artificial delay, display CAPTCHA

Best-practices

Imperial College London

- After many failed attempts, block user account, or requests from same IP
- Upon successful login
  - Show information about last login: user can report fraud
  - Notify user via email/sms if login is from unexpected machine/IP/location
    - Google currently does that
- Password managers
  - Help to handle strong passwords for many different websites
  - Huge risk when they are compromised (see related reading on DJS)
- Login via authentication provider
- See also: