Imperial College London

Network and Web Security

Authentication

Dr Sergio Maffeis
Department of Computing
Course web page: https://331.cybersec.fun

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: the Internet works
  - We'll discuss main 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("A@#GH$1F3","wonderland")="F4653BB4ACB0F3E"
  - Decrypt("A@#GH$1F3","F4653BB4ACB0F3E")="wonderland"

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

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

3. 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, resistant to *pre-image* attacks:
  given z = Hash(x) for some unknown x, find a y such that Hash(y) = z
  - 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
...
```

3. 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 a *rainbow table:* a large dictionary of (hash,password) pairs
  - Look for a stolen hash in the dictionary: if present, the corresponding password is *one* valid password for the target account

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
...
```

3. Check that presented password is correct
   - Find user salt, see if Hash(password|salt) matches entry for that user

- Steps 2,3,5 as before
- Password file still a valuable target, but less so
  - Impractical to run a generic offline dictionary attack
    - A different dictionary is needed for each 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

```
Root: Hash (6) Salt(0swk...)
Password (KGPLI...)

    username:password-data:parameters
```

- Password data
  `$hash-function-id$salt$password`
  - = disabled
- Hash-function -id
  1 = md5
  `2a, 2y` = Blowfish
  5 = sha256
  6 = sha512
- Parameters
  `16827` = 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
```

Usability (1)

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
  - Users tend to use memorable passwords
  - Users with common interests may use similar passwords
    - *"17,000 customers of UK financial services company have been using the password* `Arsenal1`*"* (ft.com, 23/1/15)
  - Offline dictionary attack does not have to try *all possible* passwords
    - Start from a vocabulary of common, memorable words
    - Apply *password-mangling* rules to generate realistic variants
      - Leetspeak: A -> 4, B ->8, E -> 3, T -> 7
      - Use one or two upper case letters
      - Append common birth years, recent years
    - Lots of tools available: John the Ripper, HashCat, Ophcrack
- Users tend to use the same password on different websites

Online dictionary attack

Imperial College London

- Attacker submits username/password combinations to a running authentication system
- Usernames are relatively easy to find
  - May be email addresses, may appear in blogposts, may be people's surnames
- Previously used passwords are easy to find
  - Lists of passwords from hacked websites can be found in the public domain, or purchased on the dark web
  - Check if your password has been leaked: https://haveibeenpwned.com
    - Since version 76, Firefox does it for you
- Defenses
  - Limit numbers of tries per username or per IP before blocking access
  - Use CAPTCHAs (but it inconveniences legitimate users)
  - *Honeypot* passwords
    - Create fake account with easy to guess username and password
    - Block requests from hosts that logs in to one of those accounts, as they may attempt to compromise legitimate accounts

Usability (2)

Imperial College London

- Complex password rules are a burden to users
  - At some point BT had 100+ employees dedicated to password reset
- Security questions are dangerous
  - Answers can be found via social media, other online footprints
    - Pet's name, favourite team, first employer
- Password *hints* (reminders) functionality should be avoided
  - Adobe hack (2013) leaked 3M encrypted (!) passwords and hints
  - People tend to choose hints 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	

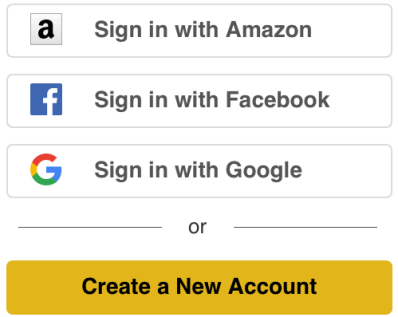
Best-practices

Imperial College London

- Use filters to ensure user selects long enough, random looking password
- Hash passwords using dedicated functions like `PBKDF2` or `bcrypt` to make it time-consuming to build rainbow tables
- Don't ask user to change password often
  - Better to have a strong, long-lived password than tempt users to choose easy-to-remember ones
- 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
  - Slow down attempts: introduce artificial delay, display CAPTCHA
  - Ask an additional security question
- 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
- See also:

Enhancements

- Password managers
  - \+ Handle strong passwords for many different websites
  - \+ Help you avoid phishing sites
  - – Help you lose access to all of your accounts in one go
  - – Help hackers get all your passwords in one go
  - – Online password manager: exposed to hackers
  - – Offline password manager: potentially unavailable
- 2FA: *2nd factor authentication*
  - \+ Prevents attacks based on weak/stolen passwords
  - – You can get locked out of your account when 2nd factor is not available
  - – Has been shown to give false sense of security
    - Users choosing guessable pin because of reliance on 2FA
  - – Introduce new device/channel in your Trusted Computing Base
- OAuth and Single Sign On
  - \+ Authenticate user via trusted identity provider
    - Social networks
    - Enterprise *Single Sign On* providers: Okta, RSA Secure ID, Azure AD, ...
  - – Protocols and deployments not immune to hacks/flaws
    - High value identity may be compromised as a result of low-value sign-on



Compromise low-value websites to get credential for Google's Account.

Alternatives to passwords

Imperial College London

- Hardware tokens from banks, one-time-password booklets from governments
  - Expensive
  - Hard to replace
- Biometric authentication
  - Impossible to replace once “lost”
  - Not that hard to “steal”, spoof
- Authentication via RFID tags
  - Risk or theft, misplacement
  - Proximity attacks
- Passwordless authentication
  - Rely on a different channel to authenticate user, typically email
  - Very easy to use: user submits email address, receives short-lived pin, submits pin
  - Email needs to be transported securely
  - If email account is compromised, all passwordless accounts for that email are too
  - So better to use for low-value accounts