Introduction

Account credentials and sessions tokens are often not properly protected
- A third party can access to one's account
- Attacker compromise password, keys or authentication token

Risks
- Undermine authorization and accountability controls
- cause privacy violation
- Identity Theft

Method of attack: use weaknesses in authentication mechanism
- Logout
- Password Management
- Timeout
- Remember me
- ...

Broken Authentication
Brute Force Attack

- **Automated process of trial and error**
  - Guess a person username and password, credit-card number, cryptographic key, ... 
  - System sends a value and waits for the response, then tries another value, and so on. 
  - Often done off-line with extracts of the Database 
  - Can be done on-line on unprotected sites 

- **Many systems allow the use of weak passwords**
  - An attacker will cycle through a dictionary (word by word) 
  - Generates thousands (potentially millions) of incorrect guesses 
  - When the guessed password is OK, attacker can access the account! 

- **Same technic can be used to guess encryption keys**
  - When the size of the key is small, 
  - An attacker will test all possible keys 

---

Brute Force Attack (Cont.)

- **Normal Brute Force**
  - For one username, 
  - Attacker tests many passwords 

Username = Emmanuel 
Passwords = 1234567, qwertz, asdfgh, abcd, .... [pet names], [birthdays], [car names], [dictionary]...

- **Lists of known passwords can be found**
  - Connection Username - Password (or hashed passwords) on the Darknet 
  - Lists of passwords (without usernames) 
    - https://github.com/danielmiessler/SecLists/tree/master/Passwords
Brute Force Attack (Cont.)

- **Normal Brute Force**
  - For one username,
  - Attacker tests many passwords

Username = Emmanuel
Passwords = 1234567, qwertz, asdfgh, abcd, ....
[pet names], [birthdays], [car names], [dictionary]...

- **Lists of known passwords can be found**
  - Connection Username - Password (or hashed passwords) on the Darknet
  - Lists of passwords (without usernames)
    https://github.com/danielmiessler/SecLists/tree/master/Passwords

Session Spotting

- **Attacker has the possibility to listen to the traffic of the victim**
  - Listens to the traffic at the IP level (sniffer)

- **Client connects to the HTTP server** www.mysite.com
  - Visits a page containing a login form (url is HTTPS)
  - Receives a cookie containing his session ID
  - Sends his credentials encrypted (HTTPS)

- **Attacker receives following information**
  - Session ID
  - Sees that the user has sent his credentials (using an encrypted connection to the server)

- **Attacker can use the cookie to be recognized as the legitimate user!**

Unsecure cookies

- **Attacker has the possibility to listen to the traffic of the victim**
  - Listens to the traffic at the IP level (sniffer).

- **Client connects to the HTTPS server**
  - https://www.mybank.com
    - Client receives a cookie containing the session ID.
    - This cookie is resent each time the browser accesses this site.
    - The cookie is linked to an active session on the secure server.

- **Victim visits a page on the unsecure web site**
  - http://www.mybank.com
    - For seeing some advertisement for instance.
    - The cookie (if not “secure”) will be sent unencrypted to the server.

- **Attacker can see the sessionID**
  - Attacker can impersonate the victim

- **Solution:**
  - Use only secure cookies (set the bit secure on)
  - Do not reuse existing cookies.
Replay Attack

Suppose the Victim wants to log on a web site
Victim sends username and password
Web Site verifies the couple
If an attacker can listen to the information transferred
Sniffer (unencrypted) / Trojan (encrypted) / Fishing / Man in the Middle . . .
He can log-in the system using Username and Password
Solution: Use challenge response
The site sends a challenge
The message sent by the user is a response to this challenge

Replay Attack (Cont.)
Examples of Challenge/Response systems

- UBS login system
  - User receives a card and an autonomous card reader system
  - when the user wants to log in, he first need to be recognized by the card
  - Types a PIN on the card reader
  - User receives a challenge sent by UBS
  - User types the challenge in the card reader
  - The card computes a response (can be used only one time)
  - The user types the response of the system on the screen
  - User is logged in!

- No replay Attack is possible here, since the information transferring on the network is only usable once.
Session Fixation Attack

- **Attacker creates a session on a web site**
  - Sends a Request,
  - Get a Response containing a cookie (SESSION_ID=1234abcd5678)
  - Attacker needs to maintain this session alive (send requests regularly)

- **Attacker sends this Session ID to the victim**
  - Can be included in a phishing. 
    He sends an email containing the reference to the following URL: https://www.ebanking.com/?SESSION_ID=1234abcd.
  - Can be a link:

\[\text{<a href="https://www.ebanking.com/?SESSION_ID=1234abcd">}\]

Victim clicks on the given link

- Browser sends automatically the SessionID within the request

- **Session 1234abcd is used by the victim**
  - Victim logs in.
  - The session is valid.

- When the attacker checks the session he/she receives the rights of the victim!

Session Fixation Attack (Cont.)

Do not accept preset or invalid session identifiers

- It is the door for Session Fixation Attack

- Reset the SessionID when a login occurs

Session Hijacking
Session Hijacking

- Credential/Session Prediction
  - Attackers deduce or guess the session id
  - Attackers can use the web site with victim’s privileges
- Rights are stored in a session, only the session id is used to link the browser and its session
  - HTTP is session-less
  - Information is not resent in each request
- Guessing the Session ID permits to be the user

Real life Example

- One web site has a “password lost” page
  - Users having lost their password ask for a renewal
  - They receive an email containing a link:
    
    ```
    <a href="https://site.com/reset?token=34349ab9938bc">Renew Password</a>
    ```
  - User clicks on this link,
    - he accesses a page where he can reset his password.
- Token is generated by a PRNG
  - Pseudo Random Number Generator are well known.
  - PRNG of PHP, Java, C, Phython, ... are well documented
  - A number is used as seed for the next number
  - If one knows one number, one can generate the next one

Session Hijacking: Example

- Many web sites generate session ID with proprietary algorithms
  - Increment static numbers
  - Can be more complicated (factoring in time and other computer specific variables)
  - Session ID is sent to the client
- An attack can be:
  - Attacker connects to the web site and gets a session ID
  - Attacker calculates or Brute Forces the next session ID
  - Attacker switches the value of the cookie and assumes the identity of the next user!

Real life Example (Cont.)

- Exploit
  - Ask for renewal of password of a real user
  - Get the token
  - Ask for the renewal of password of the administrator
  - Mail is sent to the admin (attacker can not read it)
  - Use the first token as seed to get the new token
  - Use the new token to reset the password of the admin.
Insufficient Session Expiration

- Can be exploited on a shared computing environment
  - More than one person has physical access to a computer
- Suppose logout function sends the victim to site’s home-page without deleting the session
  - Or more likely, that the user just closed the window without logging-out
- Another user could go through the browser’s history and view pages accessed by the victim
  - Since the victim’s session ID has not been deleted,
  - The attacker would be able to get the privileges of the victim.

Protection

- Authentication relies on secure communication and credential storage
- SSL should be the only option for all authenticated parts of the application
  - Otherwise, listening to credential is possible
- All credentials should be stored in hashed or encrypted form
  - Attack on the database or file system should not compromise credentials
  - Password should systematically be hashed
  - Private keys should never be stored clear text
No self-made session or SSO system

- Only use inbuilt session management mechanism
  - Do not write or use secondary session handlers!
- Do not use “remember me” or home grown Single Sign On
  - Does not apply to robust SSO or federated authentication solutions
- Writing a robust and secure solution requires high knowledge in security
  - Cryptography
  - Storage
  - ...

Protection (Cont.)

- Use a single authentication mechanism
  - With appropriate strength and number of factors
  - Ensure it is hard to spoofing and replay attacks
- Do not make the mechanism overly complex
  - It may become subject to an attack

Start login process from an encrypted page

- Do not allow the login process to start from an unencrypted pages
- Always start login from a second page
  - Encrypted
  - Using a fresh or new session token
- Prevents credential or session stealing
  - Phishing attacks
  - and Session Fixation attacks

Take Care of Logout

- Ensure that every page has a logout link
  - Users should not have to go to the start page to logout
- Logout should destroy the credentials
  - All server side session state
  - Client cookies
- Consider Human Factor
  - Do not ask for confirmation
  - Users will end up closing the window rather than logging out successfully
  - Give the users information about closing sessions
- Use a timeout period
  - Automatically logs out an inactive session
Use only strong ancillary authentication functions

- **Ancillary authentication functions?**
  - Questions and answers for password reset
- **Example:**
  - Maiden name of the mother: can be known from social engineering
  - Date of birth: can be found in Facebook (or even Wikipedia)
  - City of birth: can be tested using a catalog attack (try all the cities in Switzerland)
- **Answers should never be stored clear text**
  - Always use a one way hash function (SHA2 for instance)

No spoofable credentials as authentication

- Do not rely on credentials that can be spoofed
- **TCP/IP spoofing**
  - IP Addresses
  - Address range masks
  - DNS
  - or reverse DNS lookups
  - ...
- **HTTP spoofing**
  - Referrer Header

Be careful with e-mails

- Do not send e-mails containing passwords
  - Can be read
- Use limited-time-only random numbers to reset access
  - And send a follow up e-mail as soon as the password has been reset
- Be careful of allowing users to change e-mail
  - Send a message to the previous e-mail address before enacting the change

Specifications on Passwords

1. **Setting or Changing of Passwords**
   - Server shall verify that it is not in a previously leaked list
   - shall test if it is not too simple (‘aaaaaa’ or ‘1234abcd’)
   - shall test also context-specific words (name of the server for instance).
   - When password is rejected
     - Server shall advise the user of the rejection,
     - site must provide a reason for it.
     - and provide a metter to measure quality
   - SHOULD NOT impose composition rules
     - Like requiring mixtures of different character types or
     - prohibiting consecutively repeated characters
   - SHOULD NOT require memorized secrets to be changed arbitrarily (e.g., periodically)
     - Should force the change if there is evidence of compromise of the authenticator.

Conclusion

 Attacks on Credentials are numerous
  ▶ Session / Username and passwords / Keys
  ▶ From Brute Force to Session Hijacking
 Protection may be related with risk
  ▶ Risk for a guestbook ≠ e-banking
  ▶ Security can not be maintained at the same level
  ▶ Ratios Cost/Efficiency/Usability
 New development
  ▶ Use Biometrics for providing the credentials
  ▶ Axionics Cards used fingerprint
  ▶ Keystroke biometrics may be used for password recovery.

References

 ▶ OWASP Top 10, A2:2017
   http://www.owasp.org/index.php/Top_10_2017
 ▶ OWASP - A Guide for Building Secure Web Applications
   and Web Services
 ▶ Web Application Security Consortium: Threat
   Classification
   http://www.webappsec.org
 ▶ NIST Special Publication 800-63B, Digital Identity
   Guideline