

IIG University of Freiburg

Web Security, Summer Term 2012

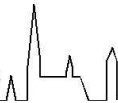
Broken Authentication and Session Management

Dr. E. Benoist

Sommer Semester



- Introduction
- Examples of Attacks
 - Brute Force
 - Session Spotting
 - Replay Attack
 - Session Fixation Attack
 - Session Hijacking
 - Session Expiration
- Protection
- Conclusion



- ▶ **Account credentials and sessions tokens are often not properly protected**
 - A third 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
 - ...



▶ **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.

▶ **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



► Normal Brute Force

- For one username,
- Attacker tests many passwords

Username = Emmanuel

Passwords = zizou, zidane, michael-schumacher,
[pet names], [birthdays], [car names],...

► Reverse Brute Force

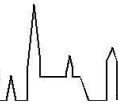
- For one password,
- Attacker tests many usernames
- Efficient if the system has millions of users
- The chance that many users use the same weak password dramatically increases.

Usernames= Emmanuel, Jan, Eric, Guenter,...

Password = 12345678



- ▶ **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!**



- ▶ **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



► UBS (Swiss Bank) 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.**



► **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 : http:
//www.gmail.com/?page=...&SESSION_ID=1234abcd.
- Can be just a reference to an image on the targeted site:

``



- ▶ **The session can be transferred using two means:**
 - URL parameter
 - Cookie
- ▶ **Targeted Web site receives the request from the victim**
 - Receives a valid SESSION_ID,
 - Resends it in the links contained in the page + as cookie
 - The page is not evaluated (browser expects an image or a javascript or a CSS or anything)
 - But the cookie is stored in the browser.
- ▶ **Next time the victim visits the target**
 - Browser sends automatically the cookie in the Request.
 - Victim logs in
- ▶ **When the attacker checks the session he/she receives the rights of the victim!**



- ▶ **Do not accept preset or invalid session identifiers**
 - It is the door for Session Fixation Attack



- ▶ **Credential/Session Prediction**
 - Attacker deduce or guess the session id
 - Attacker 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**



- ▶ **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!



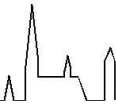
- ▶ **Can be exploited on a shared computing environment**
 - More than one person has physical access to a computer
- ▶ **Suppose logout function sends the victim to site's home-page without deleting the session**
 - Or more likely, that the user just closed the window without logging-out
- ▶ **Another 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.



- ▶ **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



- ▶ **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
 - ...



- ▶ **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



- ▶ **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



- ▶ **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



► **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
- City of birth : can be tested using a catalog attack (try all the cities in Germany)

► **Answers should never be stored clear text**

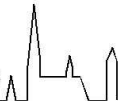
- Always use a one way hash function (SHA2 for instance)



- ▶ **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



- ▶ **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



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



- ▶ **OWASP Top 10 - 2007**
http://www.owasp.org/index.php/Top_10_2007
- ▶ **A Guide for Building Secure Web Applications and Web Services**
<http://www.lulu.com/content/1401012>
- ▶ **Web Application Security Consortium: Threat Classification (2004)**
<http://www.webappsec.org>