

(' - . . - ') _ (' - . . - .
 (00) . - . (00) (00) /
 / . - - . / , - - . , - - . / ' . _ , - - . , - - .
 | \ - . \ | | | | | ' - - . . _) | | | |
 . - ' - ' | | | | | . - ') ' - - . . - - ' | . | | |
 \ | | _ . ' | | | | | (00) | | | | |
 | | . - . | | | | | \ - ' / | | | | |
 | | | | | (' - - ' (_ . - ' | | | | |
 \ | | \ | | \ | | \ | | \ | | \ | | \ | | \

.-'') (00).	.-'') (00).	('-. .-'') _ (00) (00).	
,--.'') (_)---_) (_)---_) 00)/ _ / _ \ \ : \ : \ : (_ / ' . . ' ' .) ' . . ' ' .) , _ . ' . . _) \ . . _) \ (_ \ \ / \ / (' ' ' (_ . ' '	,--.'') (_)---_) 00)/ _ / _ \ \ : \ : \ : (_ / ' . . ' ' .) ' . . ' ' .) , _ . ' . . _) \ . . _) \ (_ \ \ / \ / (' ' ' (_ . ' '	(,-----.(_)---_) .---.' / _ \ \ : \ : '---. ' . . ' ' .) , _ . ' . . _) \ . . _) \ \ \ / \ / (' ' ' (_ . ' '	,--.'') (_)---_) 00)/ _ / _ \ \ : \ : \ : (_ / ' . . ' ' .) ' . . ' ' .) , _ . ' . . _) \ . . _) \ (_ \ \ / \ / (' ' ' (_ . ' '

Authentication Overview

- Authentication of users is important for confidentiality
- As websites move from simply displaying information to becoming web applications authentication becomes more and more important
- Authentication systems utilize a lot of technologies/systems

Authentication Walkthrough

- Note: assuming an already registered user
- User enters username and password
- Username and password are sent to the server
- The password is hashed
- The database is checked for a user containing that name and password hash that matches the submitted hash
- If the username and password match a record then a session cookie is sent back and that acts as a form of authentication for future requests
- The login is about issuing a valid session token in the form of a cookie; an ID badge sent with future requests

Can you think of any attacks on this system?

Attacks on Authentication

- Session Fixation
- Improper Session Invalidation
- Timing Attacks
- User Enumeration
 - Timing
 - Error Messages
- Insecure Cookie Settings
- Insufficient Entropy

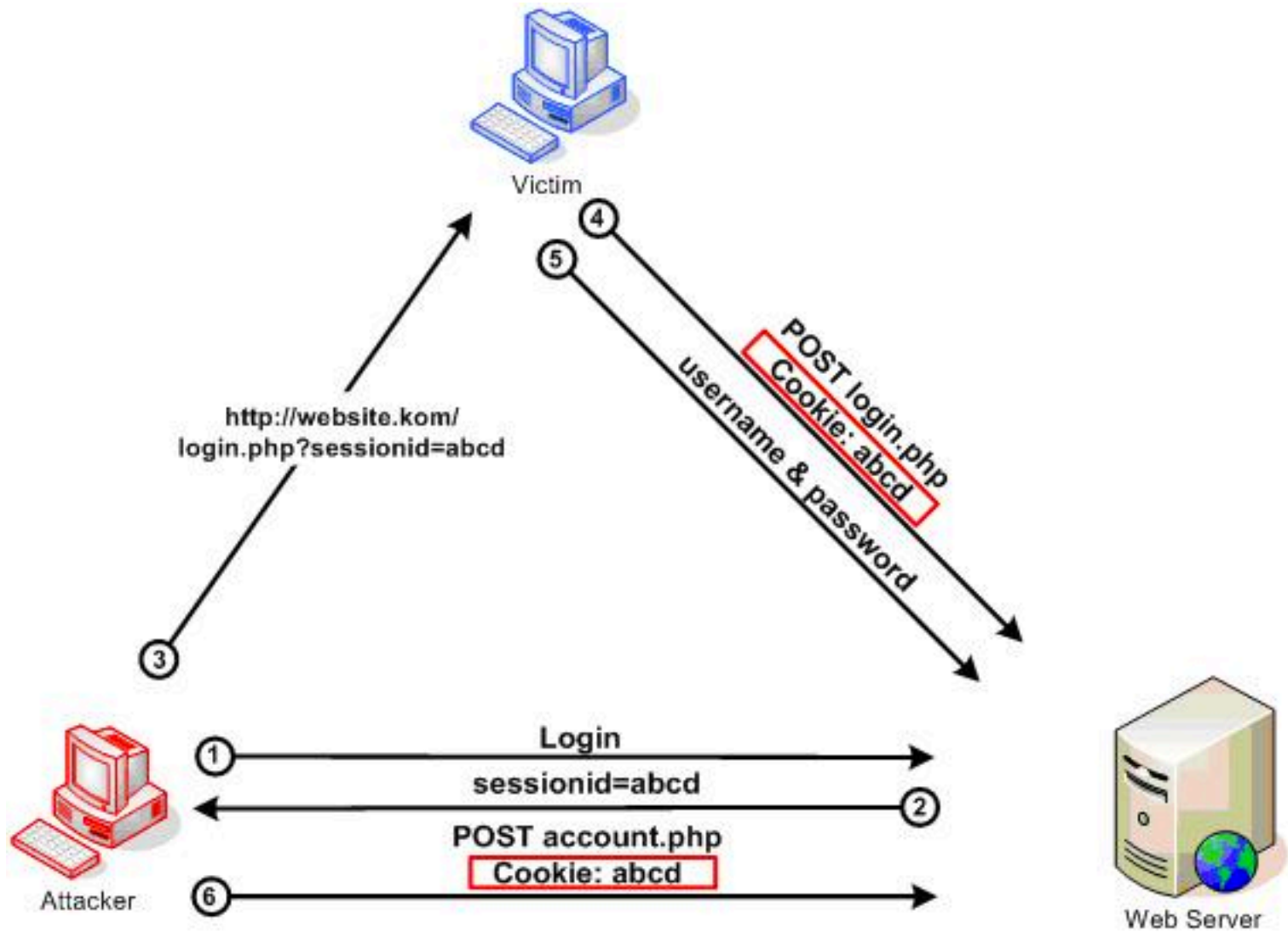
Session Fixation

Session Fixation

- An attack that allows an attacker to hijack a valid session
- Due to a system that, when authenticating a user, uses an existing session token instead of assign a new session ID
- An attacker forces a user to authenticate him/herself with a known session ID
- The known session ID then becomes valid and an attacker can use it to do evil things

Session Fixation

- Several methods execution
 - Session token in the URL argument
 - Attacker tricks victim into clicking on a crafted hyperlink containing the session token
 - Session token in a hidden form field
 - A user is tricked into authenticating using a crafted login form hosted by the attacker containing the session token
 - Session ID in a cookie
 - An XSS attack can be used to set a cookie value in the victims browser to a known value
 - Using `<meta http-equiv=Set-Cookie content="sessionid=foo">`
 - An attacker with a MITM position can send a Set-Cookie header to the victim, setting their cookies
 - If user generated code can be hosted on the same domain legitimately it can be used to set the session cookie



Testing for Session Fixation

- Login
- Change the session token (keep the same format)
- Login again
- See if a new token is issued

Fixing Session Fixation

- When a user logs in clear old session cookies and always set a new cookie

Improper Session Invalidation

Improper Session Invalidation

- When a user logs out the session should be deleted
- Sometimes it is only deleted client-side
 - Using JavaScript or setting the cookie expiration date to sometime in the past
- An attacker with a stolen session won't receive the client-side delete and can keep using the session
- Greatly increases the value of a stolen session

Testing for Improper Session Invalidation

- Login
- Copy the session token
- Logout
- Put the old session token back
- See if you are logged in

Fixing Improper Session Invalidation

- Sessions must be deleted server side!!
- Clear the session token from the database

Timing Attacks & User Enumeration

Timing Attacks

- Operations take time
 - Comparing two strings
 - Looking up a user, then their password
 - String validation checks
- The time it takes for an operation to complete leaks information about what is happening under the hood
- This can be exploited to find valid users, or even valid passwords

User Enumeration

- Timing attack
 - If the authentication form looks up the user then matches the password hash those two operations occur one after the other
 - That means that an attacker can potentially watch how long a reply takes and figure out if they have found a valid user
 - A request with a valid user will take slightly longer to reply because first the user lookup occurs, then the password check

User Enumeration

- Error messages
 - “please enter a valid user”
 - “the password you entered is incorrect”
 - “login for the user foobar failed”
- These messages can tell you if a user is in the system or not
- A valid username can then be used in a brute forcing or spear phishing attack
 - Can even be used in a fake leak

Testing for User Enumeration

- Timing attacks
 - Try logging in with several usernames (some valid and some invalid) several times
 - Use invalid passwords
 - Watch to see if a reply to a valid user takes longer than an invalid user
- Error messages
 - Try logging in with several usernames (some valid and some invalid) several times
 - Use invalid passwords
 - See if the error messages change

Fixing User Enumeration

- Lookup users in a consistent manner
- Use SQL that checks both the username and password in one statement
- Pad the response times so they take too long to brute force
- Make all error messages consistent
 - “Sorry the username or password you entered is incorrect.”

Cookie Settings & Entropy

Cookie Settings

- Name - the cookie name
- Value - the cookie value
- Path - the path the cookie applies to
- Expiration - when the cookie expires
 - Don't set! Otherwise it is written to disk
 - If expiration or Max-Age aren't set it will be deleted when the browser closes
- HttpOnly - whether the cookie is accessible to JS
 - Set to True
- Secure - whether the cookie is sent encrypted
 - Set to True if the site uses SSL/TLS

Cookie Entropy

- Make sure your session tokens use a cryptographically secure random number generator
- If not then they will be predictable and an attacker just has to try a bunch
- Make sure you have ~150 bits of entropy
- Demo time?
 - Burp sequencer