Web Application Security

Content

- Goal: Security Analysis case study
  - SocialNet Mock Application
  - Operation: Blue team vs Red team
- Mock Application attacks
  - View profile page of unauthorized user
  - Add new user via admin page in a single query
  - Change profile page of any user
  - List all usernames existing in the system
  - Login to any user account / Login as a user that does not exist.
  - Hijack session of another user

Security analysis for a project

- A new project includes
  - Project development : Focus on features
  - Project deployment and testing : Feature
  - ***Project security analysis***: Security vulnerabilities related to application features.
- How to proceed
  - Two teams: Red team tries to attack ; Blue team tries to defend.

Red team

- Analyze the application (including source code)
- Perform attack / exploitation
  - Detailed procedures and results
  - Detailed document:
    - Attack name/code
    - Attack vector specification : Steps in details with precondition and context
    - Affected components, source code file, version , …
    - Can use security framework (MITRE ATT&CK) as guidelines
- Can be persons from community
  - Bounty hunter.

Blue team

- Reactive protection
  - Received exploitations from Red team and fix them.
- Proactive protection

ATT-1: View profile page of unauthorized user

- Attack Vector: Cross-Site Request Forgery (CSRF)
  1) Login with an account (session user)
  2) Identify an account to view profile (targeted user, e.g. userX)
  3) Enter an URL according to this pattern in address bar and go: http://server:port/socialnet/profile.php?owner=userX
- DEMO
- Preventions & Mitigations:
  - Test if session user is eligible to view profile of targeted user.
  - Use CSRF token

ATT-1 (cont.): CSRF token

- Generate a "profile" link at the server with CSRF token:
  - http://server:port/socialnet/profile.php?owner=userX&CSRF=<token>
  - <token> should be generated and validated by server
  - <token> should be hashed to hide token generation logic.
- Demo
- Discussion

ATT-2: Add new user via admin page

- ❑Attack Vector: Cross-Site Request Forgery
  - ➢ Step 1: Analyze admin page to find a request pattern
  - ➢ Step 2: craft the request and go
- ❑DEMO
- ❑Prevention and mitigation
  - ➢ Add basic authentication. (Demo nginx Basic Authentication configuration to protect admin link)

ATT-2: Add new user via admin page (cont.)

- ❑ Discussion

ATT-3: Change profile page of any user

- Attack vector: SQL Injection (form input exploitation)
  - Step 1: Login with an account (session user)
  - Step 2: Identify victim user account in the system
  - Step 3: Open setting page and design a payload (depending onyour implementation)
- DEMO:
  - Change profile page of a user or all users.
- Prevention
  - Parameterized query

ATT-3: Change profile page of any user (cont.)

- Discussion

ATT-4: List all users in the system

- Attack Vector: SQL Injection (query string exploitation, UNION query attack)
  - Step 1: Open profile page with a basic owner parameter – verify that it works
  - Step 2: Craft and exploitation payload using UNION query (Depending on your server implementation)
- DEMO
- Prevention:
  - Parameterized query
  - Input sanitized checking
  - Mod Security
- Discussion

ATT-5: Login to any user

- Attack Vectors: SQL Injection (Form input exploitation, UNION query attack)
  - Step 1: open Signin page
  - Step 2: generate hash for your chosen password (e.g.: 'abc123'). Copy it
  - Step 3: use the generated hash to craft an attack payload
    "' AND 1=0 UNION SELECT 'user5','your_hash' -- " in username input and your chosen password in password input
  - Press Login
- DEMO
- Discussion

ATT-6: Hijack a login session

- Attack Vector: Social Engineering, Cross-Site Scripting (Stored XSS), Session Hijacking
  - Step 1: write simple exploitation script as the content of your profile page. The script sends 'document.cookie' to a 'data grabber' server (step 2)
  - Step 2: write simple data grabber server that read request param (e.g. $_GET) and save it to a file.
  - Step 3: wait for a victim to view your profile page, then check the saved file.
  - Step 4: copy the captured cookie in the saved file and set it into your browser's cookie
  - Step 5: reload the page. You should be logged in as the victim user.

ATT-6: Hijack a login session

- How cookies work:
  - Server -- send response --> browser
  - In **response** headers there is "Set-Cookie" header --> set cookie at the browser
  - Cookie at browser = a database keyed by domain name (and port)
  - Cookie are automatically sent to server in "Cookie" header of **request**
- How session works:
  - Session is a storage space for a client (browser 'session')
  - Session is keyed by "sessionID" (PHPSESSID)
  - Session data is stored at server
  - SessionID is set to browser via cookie.
  - Browser automatically sends sessionID to server via cookie and server load session data that belong the the sessionID. --> client state is loaded.

ATT-7: Access a logged in session via session fixation

- Attack vector: Social Engineering, Cross-Site Scripting (StoredXSS), Session Fixation
  - Step 1: write simple exploitation script (in profile page) that set your 'document.cookie' to a victim's browser. Save the profile page
  - Step 2: Log out your session
  - Step 3: wait for a victim view your profile page. After viewing the profile page, his sessionID is changed to your sessionID and he is logged out (since you are logged out)
  - Step 4: The victim logins again.
  - Step 5: you load your page and access victim's account.
- Discussion: Same as session hijacking with a different trick.