19.3 Angreifertyp 3: Websicherheit

Angreifer nutzt Schwachstellen im Browserkonzept.

Sicherheitsrelevante Funktionen des Browsers:

- Session Management (Session ID z.B. über Cookies)
- Zugriffskontrolle für Scripte der Webseite

Sicherheitsstrategie: Same Origin Policy (SOP)

- Skripte haben Zugriff auf Kommunikation zwischen Browser und Server
  Auslesen, Manipulation, Senden, Empfangen von Daten
- Realisiert über das Document Object Model (DOM)
- SOP: Zugriff nur innerhalb eines Kontextes erlaubt
- Kontext: Internetdomain, Port, Protokoll (http, https)

Beispiel. Zugriff eines Skripts in http://www.server.de/page1.html auf

- http://www.server.de/page.html: erlaubt
- http://www.server.de/page.html:81: nicht erlaubt (anderer Port)
- https://www.server.de/page.html: nicht erlaubt (anderes Protokoll)
- http://vi.server.de/page.html: nicht erlaubt (anderer Host)

Unser Ziel: Umgehen der SOP über verschiedene Angriffe
Elevation of Privilege

- Cookie Poisoning: Lesen, Manipulation, Mißbrauch der SID
- Mitlesen der Verbindung: Benutzername, Passwort, Daten

DNS Rebinding

Domain www.attacker.de erhält vertrauenswürdige IP-Adresse

Vorbereitung:

- Anmeldung der IP-Adresse www.attacker.de im DNS
- Anmedung des autoritativen Name Servers (AN) im DNS
  (unter Kontrolle des Angreifers)

Angriff 1:

- Client C ruft www.attacker.de auf (mit bösartigem Skript)
- C fragt IP-Adresse im DNS ab (u.a. AN-Server)
- A sendet zwei Adressen zurück, seine und die des Clients
  (Cs IP-Adresse bekannt aus Abfrage des AN-Servers)
- Skript hat jetzt alle Zugriffsrechte
- Gegenmaßnahme: SOP nicht über Domain, sondern über IP-Adr.
  (frühere Netscape-Versionen waren betroffen)

Angriff 2:

- A gibt seine IP-Adresse zurück mit sehr kurzer TTL
- Nach kurzer Zeit stellt Skript eine erneute Anfrage an www.attacker.de
- C fragt DNS ab, A liefert IP-Adresse von C
- Gegenmaßnahme: Browser realisiert eigenes Pinning
  in allen gängigen Browsern realisiert

Angriff 3:

- A beantwortet Pinning nicht

Cross Site Scripting (XSS)

Ausführen eines bösartigen Skripts in einem vertrauenswürdigen Kontext

- reflected XSS: Script wird über C an S geschickt
  - S hat ein Suchfeld (erreichbar unter www.server.com?search=...)
  - S zeigt Sucheingabe an (Sie haben nach ... gesucht)
  - A plaziert auf seiner Seite www.attacker.com folgenden Link:

    `<a "href=http://www.server.com?search=SCRIPT">Click</a>`
  - Skript wird im Kontext von S ausgeführt
- stored XSS: Skript wird direkt auf S plaziert
  - z.B. als Inhalt eines Beitrags im Gästebuch
- DOM-based XSS: Ohne Beteiligung von S
  - Skript liest einen Argumentwert aus der URL

    `http://www.server.de/index.html?arg=Argumentwert`

    und stellt diesen im html-Dokument dar.
  - Angreifer hat auf www.attacker.de folgenden Link:

    `http://www.server.de/index.html?arg=SCRIPT`
  - Skript wird im Kontext von S ausgeführt

Gegenmaßnahmen:

- XSS ist typischer Code Injection Angriff (also Eingaben prüfen)
- Ersetzung der Metazeichen

Cross Site Request Forgery (XSRF)

Angriffe auf eine Webseite mit Rechten eines authent. Nutzers
Authentisierung z.B. über tls, Passwort (SID mit Cookie)

Erläuterung an einem Beispiel:

- C ist auf www.server.de angemeldet
  Aktionen können z.B. wie folgt umgesetzt werden

  ```
  GET https://server.de/chpasswd?user=margraf&passwd=123456
  ```

- Angreifer schiebt C einen http-Request unter
  (z.B. vial Link auf der Seite www.attacker.de)

  ```
  <a href="https://server.de/chpasswd?
                user=attacker&passwd=654321">Click</a>
  ```

Gegenmaßnahme:

- Client: Ausloggen, wenn weitere Seiten aufgerufen werden
- Server: Nutze zusätzliche Zufallszahl
  - Nach Authentisierung erhält C von S random_number
  - Diese muss bei jeder sensiblen Transaktion übergeben werden

    ```
    GET https://server.de/chpasswd?
            rn=random_number?user=margraf&passwd=123456
    ```

  - Attacker kann Zufallszahl nicht auslesen (und damit nicht nutzen)
  - Aber: XSS-Angriff kann dies umgehen (wie?)

Weitere Maßnahme: Transaktionsnummern