Vulnerabilitati de Securitate WEB
Studiu de caz

Alexandru Damian Cirlan

Dan-Stefan Florescu

Universitatea Titu Maiorescu

Bibliografie

Standford University – CS 253 Web Security

Web Programming Adventure

HTML

JavaScript

Node.js

Journey to the Dark Side

XSS

Oh What a Tangled Web We Weave

Notiuni teoretice despre practici bune pentru securitatea cibernetica si igienizare

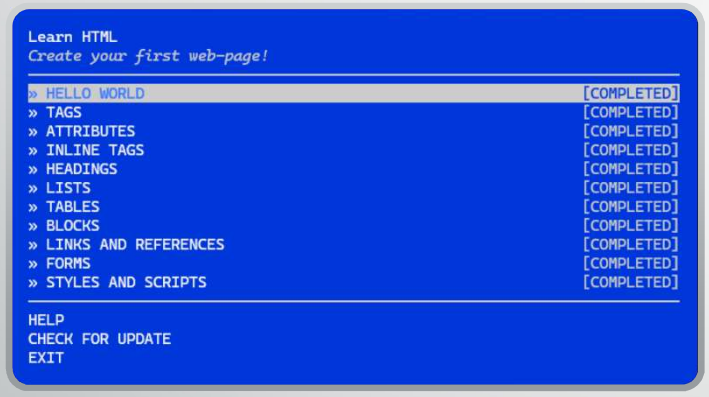
Somebody's Always Watching

Amprentarea sesiunilor web ale utilizatorilor

Swiss Cheese Security

Atac pagina web Gruyere

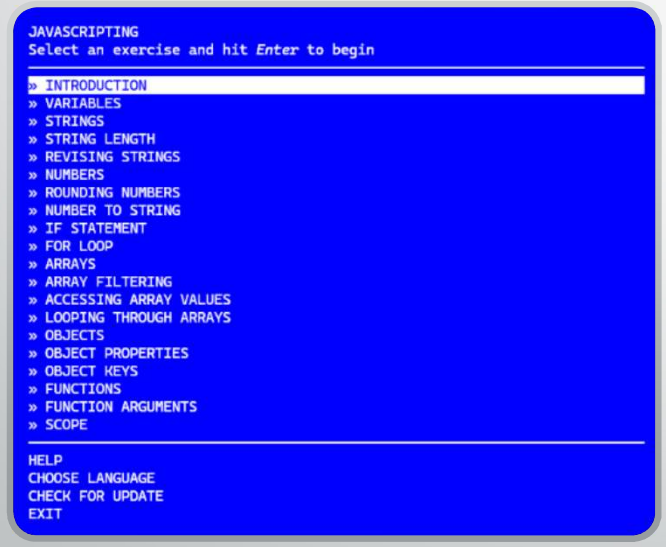
Web Programming Adventure



learnyouhtml – Denys Dovhan
Laborator cu aplicatii practice pentru HTML

- HTML (HyperText Markup Language) este limbajul standard utilizat pentru a crea pagini web
- El defineste structura unei pagini web prin etichete (tags)
  - Exemple: <html>, <head>, <body>, <h1>, <p>
- Aceste etichete organizeaza continutul in titluri, paragrafe, liste, linkuri, imagini si alte elemente vizuale sau functionale

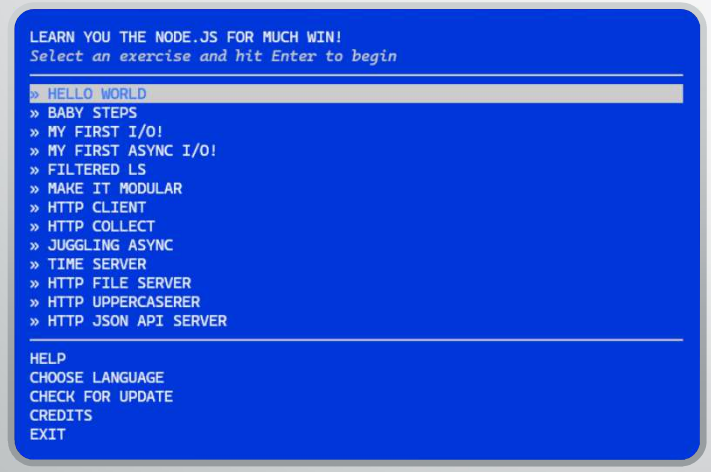
Web Programming Adventure



javascripting – Denys Dovhan
Laborator cu aplicatii practice pentru JavaScript

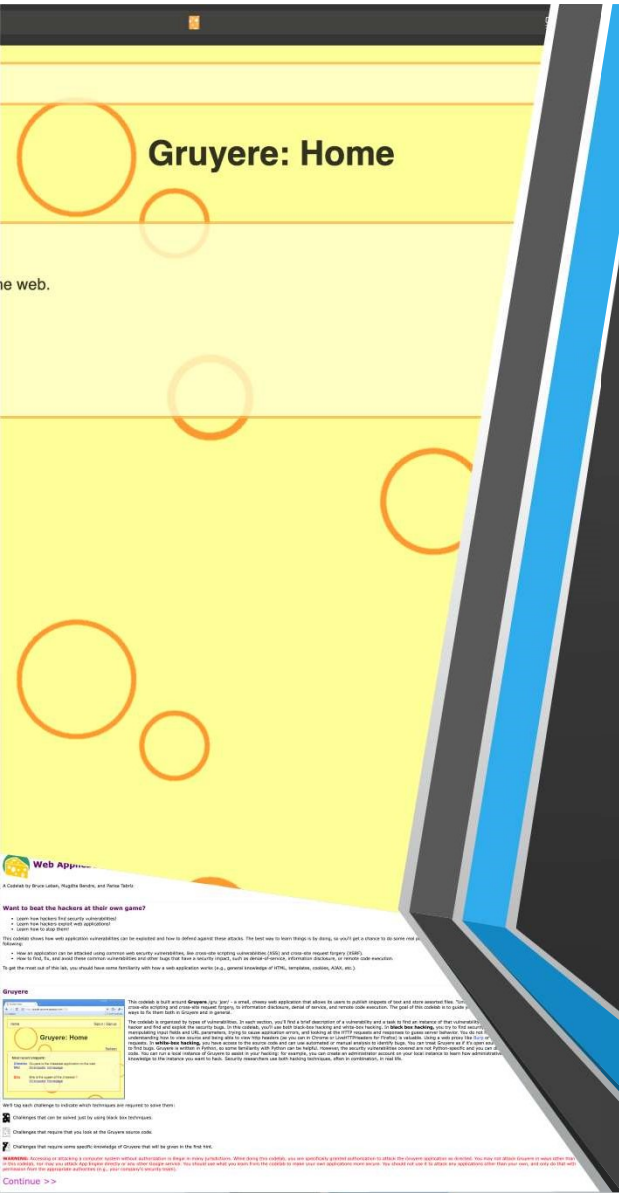
- JavaScript este un limbaj de programare folosit in principal pentru a face paginile web interactive si dinamice
  - Raspunde la actiuni ale utilizatorului (clickuri, apasari de taste, miscarea mouseului etc)
  - Poate adauga, sterge sau modifica elemente HTML din mers (fara reincarcarea paginii)
  - Trimite si primeste date in fundal folosing API-uri (fetch, AJAX)
  - Creeaza tranzitii, glisari, fade-uri etc

Web Programming Adventure



learnyounode – Denys Dovhan
Laborator cu aplicatii practice pentru Node.js

- Node.js este un mediu de executie JavaScript construit pe motorul V8 (acelasi utilizat de Google Chrome) care permite rularea elementelor de tip JavaScript in afara unui browser (pe computer/server)
- Este folosit pentru:
  - Crearea de aplicatii web si API-uri
  - Rularea scripturilor de automatizare
  - Dezvoltarea instrumentelor CLI (ex: **learnyounode, npm** etc)



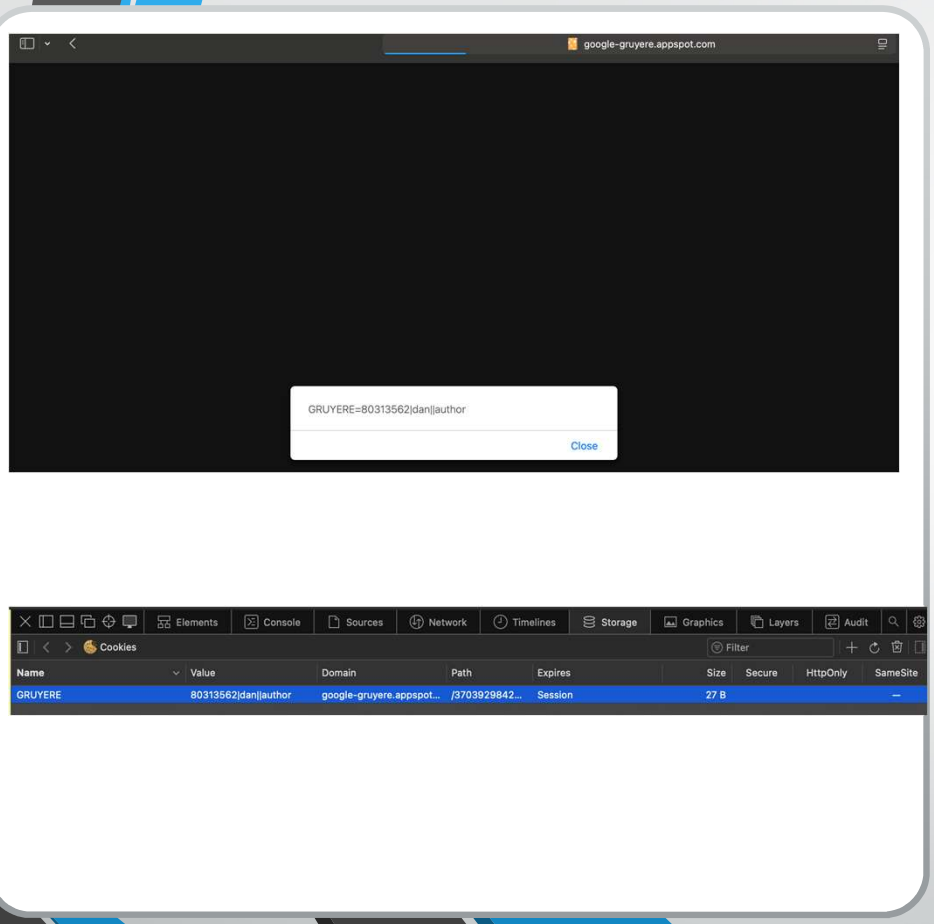
Ce este gruyere ?

- Gruyere este un mediu controlat, creat de Google, care permite ca exercitii diverse atacuri pe server

Google-Gruyere

Ce este XSS ?

- XSS (CrossSite Scripting) este o vulnerabilitate de securitate care apare atunci cand un site web permite injectarea de cod JavaScript malitios in paginile sale, cod care este apoi executat in browserul altor utilizatori
- Acest tip de atac poate fi folosit in a:
  - Fura cookie-uri, token-uri de autentificare, date personale
  - Crea redirecturi catre site-uri malitioase
  - Afisa mesaje false (phishing)
  - Controla pagina din browserul victimei



Gruyere – Reflected XSS

- Este un atac in care un cod malitios este injectat intr-un url sau formular, iar apoi este reflectat in raspunsul serverului fara sa fie stocat.
- Codul ruleaza o singura data, la request-ul respectiv.
- Folosit in phishing.
- **Protectii:** CSP, SameSite cookies si altele

Link-ul inserat:

https://google-gruyere.appspot.com/3703929842400372002062466928707297864 58/%3Cscript%3Ealert(document.cookie)%3C%2Fscript%3E

Gruyere – Stored XSS

- Este un tip de atac prin care codul malitios este salvat permanent pe server si este livrat tuturor utilizatorilor care acceseaza acea pagina
- Codul este salvat si reexecutat de fiecare data cand cineva incarca pagina afectata
- **Mecanisme de protectie impotriva Stored XSS:**
  - Escape la afisare
  - Validare de input pe keywords precum: <script>, onload=, javascript: , etc;
  - CSP (Content Security Policy)

Tipuri de atacuri XSS via AJAX (Asynchronous JavaScript and XML)

1. Reflected XSS via AJAX

- Reflected XSS apare cand codul malitios este inclus direct in URL si executat imediat, ca parte dintr-un raspuns al serverului (de obicei la o cerere AJAX). Exploit-ul se executa pe loc, fara sa fie stocat pe server.
- Masuri de protectie:
  - validare pe server
  - escape la afisare
  - folosirea CSP
  - setarea corecta a content-type in raspunsuri
- **Caz practic:** un atacator introduce in URL un script care foloseste fetch pentru a trimite cookie-ul victimei catre un server controlat de el. Cand victima acceseaza linkul, frontend-ul trimite automat datele prin AJAX catre un endpoint. Serverul raspunde cu un HTML care contine payload-ul XSS, iar scriptul se executa si fura cookie-ul, trimitandu-l catre serverul atacatorului, unde poate fi salvat.

2. Stored XSS via AJAX

- **Stored XSS** apare cand un atacator salveaza cod malitios pe server, iar acesta este returnat si inserat in DOM (de exemplu printr-un request AJAX). Codul se executa cand este incarcat in browser.
- Masuri de protectie:
  - escape pe server si client
  - evitarea folosirii functiilor periculoase precum eval ()

Caz practic: un atacator salveaza un comentariu cu <script>alert('XSS')</script>. Cand comentariile sunt incarcate prin AJAX, codul ruleaza si poate fura cookie-ul unui cont admin. Cu acel cookie, atacatorul se poate autentifica. Exemplu: aplicatia Gruyere.

Alte tipuri de atacuri XSS

Cross-Site Request Forgery (XSRF)

- Este un tip de atac in care un atacator forteaza browserul unei victime sa trimita o cerere catre o aplicatie legitima in momentul in care victima viziteaza un site de tip evil.
- Victima detine un cookie de sesiune pentru site-ul bank.com -> acceseaza site-ul victim.com care contine un script malitios care face o cerere catre site-ul victimei prin care poate transfera o suma de bani -> Cookie-ul de sesiune pentru bank.com este existent, deci serverul executa comanda
- **Mecanisme de protectie:**
  - Token CSRF => se adauga in formulare; serverul verifica valabilitatea token-ului CSRF si doar atunci un user poate initia o actiune bancara
  - SameSite Cookie => blocheaza cookie-urile pentru cereri cross-site => cookie-ul de sesiune nu permite request-uri de la alte site-uri
  - Verificare Origin => se compara headerul Origin cu domeniul asteptat (serverul verifica de pe ce domeniu este facuta originea)
- **Caz practic:** Injectarea unui request malitios pe un site de tip victim.com ; Request-ul malitios trebuie sa execute o actiune de pe alt server

Cross-Site Script Inclusion (XSSI)

- Este o vulnerabilitate prin care un atacator include un script de pe un site legitim intr-un site rau intentionat si astfel poate obtine date sensibile neautorizate.
- **Metode de protectie:**
  - setarea header-ului X-Content-Type-Options: nosniff pentru a preveni interpretarea gresita a continutului
  - folosirea token-urilor anti-CSRF si autentificare pentru accesul la resurse
  - implementarea CORS strict pentru a controla accesul cross-origin
  - evitarea expunerii datelor sensibile in scripturi accesibile public

Client State Manipulation

Cand o pagina web este accesata, browserul trimite un request HTTP de tip GET catre server pentru a cere continutul paginii.
Totusi, deoarece browserul ruleaza pe un device care poate fi compromis sau controlat de un atacator,
aplicatia web nu trebuie sa aiba incredere in datele primite de la browser.
Acestea pot fi modificate intentionat pentru a exploata vulnerabilitati.

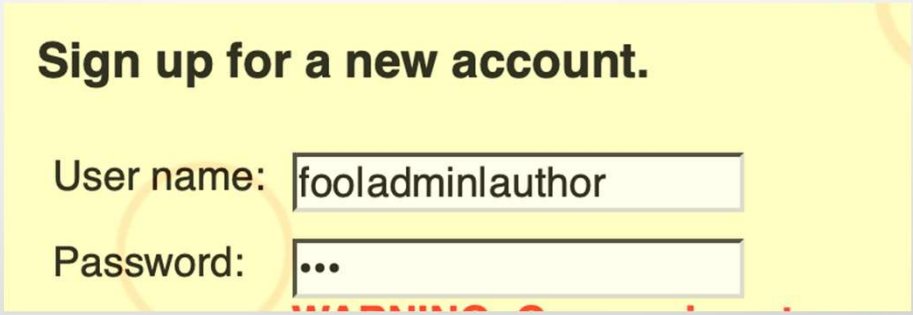
1. Elevation Of Privilege

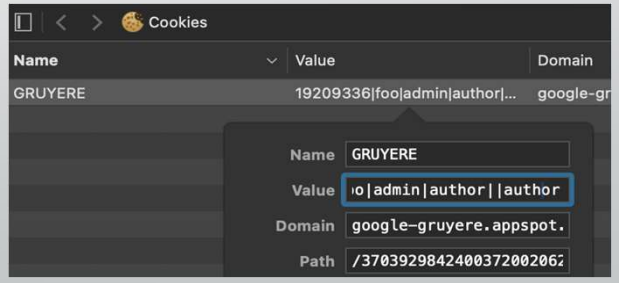
- Elevation of Privilege (EoP) este o vulnerabilitate de tip Broken Access Control, unde aplicatia nu controleaza corect ce actiuni poate face fiecare utilizator. Astfel, un user poate obtine acces la resurse sau functii la care nu are drepturi.
- Accesare link: *https://google-gruyere.appspot.com/370392984240037200206246692870729786458/saveprofile?**action=update&is_admin=True&uid=damian***
  - Masuri de protectie:
    - verificare in backend daca utilizatorul are voie sa faca acel request
    - verificare a rolului si permisiunilor pentru fiecare actiune

Client State Manipulation

2. Cookie Manipulation

- Cookie Manipulation apare cand informatii sensibile sunt stocate in cookie-uri fara protectie, permitand modificarea lor de catre un atacator.
- **Masuri de protectie:**
  - folosirea unui token generat random si apoi validat pe server
  - semnarea criptografica a cookie-ului (HMAC, JWT)
  - cookie fara date sensibile (ex: parole, roluri)
  - sesiunea se sterge la logout si are expiration date





Path Traversal

- **Path Traversal**
  - **Information disclosure via path traversal**
    - **Information disclosure via path traversal** este o vulnerabilitate care permite unui atacator sa acceseze fisiere din afara directorului aplicatiei prin manipularea cailor fisierelor, folosind secvente precum ../ pentru a urca in structura de directoare.
    - Caz practic: Cautam in structura de foldere a serverului fisierul secret.txt
    - **Metode de protectie:**
      - Normalizeaza si verifica calea fisierului sa ramana in directorul permis
      - Foloseste o lista de fisiere permise (whitelist)
      - Nu include fisiere pe baza inputului utilizatorului fara validare
      - Ruleaza aplicatia cu drepturi minime
  - **Data tampering via path traversal**
    - Data tampering via path traversal este o vulnerabilitate care permite unui atacator sa modifice, stearga sau inlocuiasca fisiere de pe server, accesand cai neautorizate prin secvente precum ../. Exploatarea apare cand inputul utilizatorului este folosit nesigur pentru scrierea in fisiere.
    - Metode de protectie:
      - Verifica si normalizeaza calea fisierului inainte de scriere
      - Permite scrierea doar in directoare prestabilite
      - Nu folosi inputul utilizatorului direct in operatii de scriere
      - Ruleaza aplicatia cu permisiuni limitate (fara acces la fisiere sensibile)

Denial Of Service (DoS)

- Este un tip de atac cibernetic care are ca scop blocarea accesului legitim la un serviciu, sistem sau retea

- DoS - Quit the Server
  - Apelam un URL dedicat inchiderii serverului care face parte din fisierul **manage.gtl**
  - **Masuri de protectie**
    - Introducem URL-ul la o lista accesibila numai administratorilor
    - Important de luat in considerare faptul ca apelurile la URL-uri sunt case sensitive
  - https://google-gruyere.appspot.com/5539088297170062870828359566057703539 88/quitserver
- DoS - Overloading the Server
  - Acest tip de atac se realizeaza prin utilizarea metodei de atac **path traversal** pentru a introduce payload-ul in zona dorita pentru a supraincarca server-ul cu date redundante
  - Exemplu: template-ul **menubar.gtl** se regaseste in fiecare pagina din Gruyere, astfel modificand continutul acestuia cu un payload care va apela in mod continuu acest template, rezultand in supraincarcarea serverului
  - Masuri de protectie
    - Aceleasi masuri de protectie trebuie abordate ca la atacul de tip **path traversal**
    - Trebuie impuse limite la tipul de fisiere care pot fi incarcate in server

AJAX Vulnerabilities

- **Codul vulnerabil de tip AJAX poate permite atacatorilor sa modifice parti din aplicatie in moduri in care nu te ai putea astepta**
- DoS via AJAX
  - Pentru acest atac, exploatam modul in care site-ul web interpreteaza anumite input-uri pentru a interzice vizualizarea anumitor elemente de catre utilizatori
  - Exemplu: in cazul site-ului web Gruyere, cream un utilizator numit **private_snippet,** un snippet normal pe acest cont. Astfel, modificam continutul snippet ului privat al utilizatorului care urmeaza dupa acesta
  - Masuri de protectie
    - Modificam structura codului AJAX pentru a diferentia variabilele server ului si input urile utilizatorilor
- Phishing via AJAX
  - In acest caz, exploatam modul in care se interpreteaza valorile date de utilizatori pentru a introduce referinte catre site-uri malitioase
  - Exemplu: in cazul Gruyere, creeam un utilizator **menu-right** cu un snippet care sa contina referinte la Sign in si Sign up catre site ul malitios
  - Masuri de protectie
    - Modificam modul in care sunt create id urile utilizatorilor pentru a se distinge functiile HTML de input urile utilizatorilor

Configuration Vulnerabilities

- Configuration Vulnerabilities
  - Este un tip de vulnerabilitate care intervine atunci cand aplicatiile sunt instalate cu setarile default
  - Exemplu: combo username x password: admin // admin
  - Exemplu: afisarea template-urilor redundante care fac parte din structura de baza a site ului
    - In structura site-ului web Gruyere, exista un template default numit dump.gtl care afiseaza continutul bazei de date
    - Acest template poate fi apelat printr un URL
    - Masuri de protectie
      - Eliminarea oricarui fisier redundant (bloatware)
    - Dupa implementarea solutiei precedente, baza de date tot poate fi accesata prin introducerea template-ului dump.gtl inapoi in structura paginii
    - Masuri de protectie
      - Limitarea/interzicerea anumitor tipuri de fisiere sa fie incarcate in server de utilizatori
    - Chiar si cu aceste masuri de protectie implementate, baza de date poate fi accesata prin introducerea liniei de cod {{_db:pprint}} intr un snippet privat
    - Vulnerabilitatea apare pentru ca sistemul de template extinde variabilele de doua ori. Acest fapt permite unui atacator sa execute cod arbitrar sau sa acceseze date sensibile
    - Masuri de protectie
      - Modificam logica de expansiune din codul template ului astfel incat sa nu mai evalueze variabilele deja expandate

Were mentioned

CSP (Content Security Policy)

- CSP (Content Security Policy) este un mecanism de securitate implementat printr-un header HTTP sau un tag meta care controleaza ce resurse pot fi incarcate de o pagina, cum ar fi scripturi, imagini, iframe-uri sau date trimise. Scopul este prevenirea atacurilor de tip XSS si injection.
- De exemplu, o politica **script-src 'self'** permite doar scripturi de pe acelasi domeniu, iar **img-src 'self'** permite doar imagini de pe acelasi domeniu.

Atacuri asupra sesiunilor

- SameSite este un atribut al cookie-urilor care controleaza daca acestea pot fi trimise in request-uri cross-site. Este folosit pentru a preveni atacuri de tip CSRF.
- Exemplu:
  SameSite=Lax – permite trimiterea cookie-ului doar in request-uri GET simple, nu si in POST-uri cross-site.
- Scenariu:
  Un utilizator viziteaza evil.com, care incearca sa trimita un formular POST catre bank.com. Daca cookie-ul de sesiune de la bank.com are SameSite=Lax, el nu va fi trimis, iar atacul esueaza.

Were mentioned

Sanitizarea codului

- **Sanitizarea codului** inseamna validarea si curatarea datelor externe (ex: input de la utilizator) pentru a preveni atacuri precum SQL Injection sau XSS. Aceasta trebuie facuta **la afisare,** nu la salvare, pentru a pastra continutul.
- O solutie eficienta: **interogari parametrizate**.

User-agent

- **User-Agent** este un header HTTP ce contine metadate despre browser, sistemul de operare si alte detalii tehnice.
- Este trimis automat de browser la fiecare request, dar poate fi modificat de client.

Were mentioned

SOP (Same Origin Policy)

- **SOP** (Same-Origin Policy) este o regula impusa de browsere care blocheaza citirea raspunsurilor de la alt site, nu trimiterea request-urilor.
- Ex: JS din siteA.com nu poate citi raspunsuri de la siteB.com fara permisiune.
- Exceptii se definesc prin **CORS**.

CORS (Cross-Origin Resource Sharing)

- CORS (Cross-Origin Resource Sharing) este un mecanism care relaxeaza regula SOP si permite unui server sa accepte request-uri din alte origini.
  Exemplu: daca frontend-ul este pe localhost:3000 si backend-ul pe localhost:5000, serverul de pe :5000 trebuie sa permita explicit accesul din :3000.
  Best practice: nu se foloseste Access-Control-Allow-Origin: * in productie, ci se definesc clar domeniile permise.

Alte tipuri de vulnerabilitati

Buffer Overflow

- Atac cybernetic care exploateaza modul in care buffer ul (o zona de memorie alocata) stocheaza datele din fluxul current. Aceasta vulnerabilitate poate fi folosita in:
  - DoS
  - Privilege Escalation
  - Instalare de malware

SQL Injection

- Asemanatoare cu injectiile de tip XSS, aceste exploit uri au ca scop manipularea bazei de date a paginii web in mod neautorizat.

Concluzii

Securitatea trebuie integrata inca din faza de proiectare

Testarea constanta este esentiala

- OWASP ZAP (scanner), Burp Suite (cereri HTTP/S), Postman (API)

Cele mai frecvente vulnerabilitati sunt adesea rezultatul unor greseli simple

- Respectarea practicilor bune este esentiala pentru prevenirea atacurilor

Multumim!