Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

**Referat**

# La disciplina „Testarea Produselor Program”

# Tema: “Securizarea aplicaţiilor web.”

A efectuat: st. gr. SI-211 S. Chirita

A verificat: dr. conf.univ. A. Prisacaru

Chișinău – 2024

**Cuprins:**

1. **Introducere**
   * Importanța securității aplicațiilor web
   * Rolul și influența recomandărilor OWASP
2. **Injecții**
   * Descrierea atacurilor de injectare
   * Metode de prevenire și mitigare
3. **Autentificare ruptă**
   * Probleme comune în autentificare
   * Soluții pentru întărirea proceselor de autentificare
4. **Dezvăluirea datelor critice**
   * Riscuri asociate cu gestionarea necorespunzătoare a datelor
   * Strategii de protecție a datelor sensibile
5. **Entități XML externe (XXE)**
   * Explicarea atacurilor XXE
   * Prevenirea și securizarea analizoarelor XML
6. **Controlul accesului întrerupt**
   * Importanța controlului strict al accesului
   * Metode de asigurare a unui control eficient al accesului
7. **Configurare de securitate incorectă**
   * Consecințele unei configurații de securitate inadecvate
   * Abordări pentru o configurare corectă și sigură
8. **Cross-Site Scripting (XSS)**
   * Natură și impactul atacurilor XSS
   * Tehnici de protecție împotriva XSS
9. **Deserializare nesigură**
   * Pericolele deserializării nesigure
   * Abordări pentru securizarea proceselor de serializare și deserializare
10. **Utilizarea componentelor cu vulnerabilități cunoscute**
    * Problemele asociate cu componentele învechite sau nesigure
    * Strategii de gestionare și actualizare a componentelor
11. **Jurnalele insuficient detaliate și monitorizarea slabă**
    * Impactul unei monitorizări ineficiente
    * Importanța jurnalizării detaliate și a monitorizării proactive
12. **Injecție de comenzi de sistem**
    * Explicarea atacurilor prin injecție de comenzi
    * Metode de prevenire a acestor tipuri de atacuri
13. **Securitate slabă la nivel de API**
    * Vulnerabilitățile comune în API-uri
    * Soluții pentru întărirea securității API-urilor
14. **Concluzii**

**Rezumat:**

O vulnerabilitate este o gaură sau o slăbiciune a aplicației, care poate fi un defect de proiectare sau un bug de implementare, care permite unui atacator să provoace prejudicii părților interesate ale unei aplicații. Părțile interesate includ proprietarul aplicației, utilizatorii aplicației și alte entități care se bazează pe aplicație. Vă rugăm să nu publicați vulnerabilități reale în produse, servicii sau aplicații web. Aceste rapoarte de dezvăluire ar trebui postate pe bugtraq sau pe listele de corespondență cu divulgare completă.

**Cuvinte cheie:**

OWASP-Open Web Application Security Project (https://ru.wikipedia.org/wiki/OWASP).

Injecție SQL- este o tehnică de injectare a codului.

XXE- Atacul unei entitati externe.

**Introducere:**

Utilizarea recomandărilor Open Web Application Security Project (OWASP) a fost mult timp standardul pentru securitatea aplicațiilor web. Să vedem care sunt 10 vulnerabilități pe care organizația le consideră cele mai periculoase în acest moment.

1. **Injecții**

Atacurile de injectare apar atunci când datele neîncrezătoare sunt transmise interpretorului de cod prin introducerea formularului sau printr-o altă modalitate de trimitere a informațiilor către o aplicație web. De exemplu, un atacator ar putea introduce cod SQL într-un formular care așteaptă un nume de utilizator. Dacă intrarea nu este protejată corespunzător, va duce la executarea codului - astfel de atacuri sunt cunoscute sub numele de injecții SQL.

Atacurile prin injecție pot fi prevenite prin validarea și/sau igienizarea datelor trimise de utilizator. În primul caz, datele suspecte sunt respinse complet, iar în al doilea caz, doar partea suspectă a acestora este ștearsă. În plus, DBA poate seta controale speciale pentru a minimiza cantitatea de informații pe care o poate dezvălui un atac de injecție SQL.

1. **Autentificare ruptă.**

Vulnerabilitățile de autentificare pot permite atacatorilor să obțină acces la conturile de utilizator, inclusiv la cele privilegiate, care pot fi apoi utilizate pentru a obține controlul asupra sistemelor de informații corporative.

Site-urile web au adesea probleme cu mecanismele de autentificare. În primul rând, acestea sunt asociate cu gestionarea incorectă a sesiunii, care poate fi folosită de atacatori pentru a obține acces la conturile de utilizator și acreditările de conectare.

OWASP Top 10 conține o listă întreagă de astfel de vulnerabilități:

* Posibilitatea de a completa automat acreditările și/sau de a selecta o parolă.
* Abilitatea de a utiliza parole standard, slabe și binecunoscute.
* Procese ineficiente de recuperare a acreditărilor.
* Lipsa autentificării cu mai mulți factori (MFA) sau implementare slabă.
* Furnizarea ID-urilor de sesiune într-un URL (Uniform Resource Locator), fără intercalarea ID-urilor de sesiune și revocarea incorectă a ID-urilor de sesiune și a simbolurilor de autentificare atunci când un utilizator se deconectează sau după o perioadă de inactivitate.

Astfel de vulnerabilități apar adesea din lipsa de experiență a dezvoltatorilor, probleme de testare și lansări de software prea grăbite.

Numărul de vulnerabilități poate fi redus prin implementarea autentificării cu mai mulți factori, precum și prin introducerea de restricții care fac imposibile atacurile automate cu forță brută (de exemplu, prin forță brută). La fel de importantă este lipsa de grabă și timpul necesar dezvoltatorilor pentru a testa modificările înainte de lansarea în producție, precum și implementarea procedurilor de testare a codului pentru securitate. De asemenea, trebuie să implementați o politică puternică de parole și să utilizați manageri de sesiune securizat.

1. **Dezvăluirea datelor critice**

Principalul motiv al riscului dezvăluirii datelor critice se datorează lipsei de criptare sau utilizării unor metode nesigure de generare și gestionare a cheilor, algoritmi slabi de criptare, metode nesigure de stocare a parolelor etc. În plus, dezvoltatorii de aplicații web stochează adesea date sensibile chiar și atunci când nu sunt necesare.

1. **Entități XML externe (XXE)**

Atacurile XXE vizează aplicațiile web care analizează Extensible Markup Language (XML). Acestea apar atunci când intrarea care conține un cod XML care face referire la un obiect extern este procesată de un parser cu o configurație slabă. Analizatoarele XML sunt adesea vulnerabile la XXE în mod implicit, ceea ce înseamnă că dezvoltatorii trebuie să elimine manual vulnerabilitatea.

Atacurile XXE pot fi evitate dacă aplicațiile web acceptă forme mai puțin complexe de date (cum ar fi jetoanele web JavaScript Object Notation (JSON)), repară analizatoarele XML sau dezactivează utilizarea entităților externe. Vă puteți proteja împotriva atacurilor XXE prin implementarea gateway-urilor de securitate pentru interfața de programare a aplicațiilor (API), patch-uri virtuale și firewall-uri pentru aplicații web (WAF).

1. **Controlul accesului întrerupt**

Problemele de control al accesului permit atacatorilor să ocolească restricțiile și să obțină acces neautorizat la sisteme și date sensibile și, potențial, să obțină acces la conturile de administrator și utilizatori privilegiați.

Riscul de încălcare a controlului accesului poate fi redus prin implementarea conceptului de acces cel mai puțin privilegiat, verificarea regulată a serverelor și site-urilor web, aplicarea MFA și eliminarea utilizatorilor inactivi și a serviciilor inutile de pe servere. De asemenea, puteți securiza controalele de acces utilizând jetoane de autorizare atunci când utilizatorii se conectează la o aplicație web și le invalidați atunci când se deconectează. Alte recomandări includ înregistrarea și raportarea erorilor de acces și utilizarea limitării ratei pentru a minimiza daunele cauzate de atacurile automate.

1. **Configurare de securitate incorectă**

Configurațiile greșite de securitate sunt considerate cea mai comună vulnerabilitate din Top 10 OWASP și sunt cel mai frecvent asociate cu utilizarea setărilor implicite ale site-ului web sau a unui sistem de management al conținutului (CMS). Erorile obișnuite de configurare includ, de asemenea, neremedierea defectelor software, pagini web neutilizate, directoare și fișiere neprotejate, permisiuni implicite de partajare pentru serviciile de stocare în cloud și servicii neutilizate sau care nu sunt necesare.

Configurarea greșită a securității poate fi oriunde: în aplicații și servere web, baze de date, servicii de rețea, cod personalizat, cadre, mașini virtuale preinstalate și containere.

Configurația de securitate poate fi corectată prin modificarea setărilor implicite ale serverului web sau CMS, eliminând caracteristicile de cod neutilizate și controlând datele utilizatorului și vizibilitatea informațiilor utilizatorului. De asemenea, dezvoltatorii ar trebui să elimine documentația, caracteristicile, cadrele și mostrele inutile, să segmenteze arhitectura aplicației și să automatizeze validarea eficienței configurațiilor și setărilor mediului web.

1. **Cross-Site Scripting (XSS)**

Vulnerabilitățile XSS permit infractorilor cibernetici să injecteze scripturi într-un site web și să-l folosească pentru a distribui cod rău intenționat care se execută în browserul unui utilizator. De regulă, acest lucru este necesar pentru a intercepta sesiunile utilizatorilor, pentru a fura date confidențiale sau pentru a redirecționa utilizatorul către site-uri rău intenționate.

Puteți preveni exploatarea vulnerabilităților XSS utilizând paravanele de protecție pentru aplicații web (WAF), în timp ce dezvoltatorii pot reduce șansa de atacuri XSS prin separarea datelor nesigure de browserele active. Aceasta include utilizarea cadrelor care evită XSS prin proiectare, utilizarea igienizării și validării datelor, evitarea datelor de solicitare HTTP (Hypertext Transfer Protocol) nesigure și implementarea Politicii de securitate a conținutului (CSP).

1. **Deserializare nesigură**

În termeni de stocare a datelor și informatică, serializarea înseamnă convertirea obiectelor sau structurilor de date în șiruri de octeți. Deserializarea înseamnă convertirea acestor șiruri de octeți în obiecte. Deserializarea nesigură implică atacatorii care modifică datele înainte ca acestea să fie deserializate.

Avizul de securitate OWASP pentru deserializarea nesigură se ocupă de supercookie-urile care conțin informații serializate despre utilizatori. Dacă atacatorii pot deserializa cu succes obiectul, ei își pot acorda un rol de administrator, se pot serializa datele și pot compromite toate aplicațiile web.

1. **Utilizarea componentelor cu vulnerabilități cunoscute**

Componentele software, cum ar fi cadrele și bibliotecile sunt adesea folosite în aplicațiile web pentru a oferi anumite funcționalități. Cu toate acestea, aceste componente pot conține vulnerabilități care permit unui atacator să lanseze un atac cibernetic.

Adesea, dezvoltatorii nu actualizează componentele de la terți, deoarece codul lor învechit nu funcționează cu cele mai recente versiuni ale software-ului, iar webmasterii fie sunt îngrijorați că actualizările întrerup site-urile, fie nu au experiența necesară pentru a aplica actualizări. În plus, atacatorii caută în permanență potențiale vulnerabilități pe care dezvoltatorii nu le-au descoperit încă (cunoscute sub numele de vulnerabilități zero-day) și pe care le pot exploata.

Acest lucru poate fi evitat cu un patch virtual care protejează site-urile web vechi de a fi exploatate de firewall-uri, sisteme de detectare a intruziunilor (IDS) și WAF-uri. Exploatarea vulnerabilităților poate fi prevenită și prin păstrarea doar a componentelor care sunt efectiv necesare și prin eliminarea celor care sunt neutilizate sau neîntreținute. Cea mai bună metodă este să instalați componente din surse de încredere și să le păstrați la zi.

1. **Jurnalele insuficient detaliate și monitorizarea slabă**

Un atac de hacker de succes sau o scurgere de date nu este întotdeauna ușor de detectat. Adesea, răufăcătorii nu numai că primesc acces neautorizat la sistemele informaționale, dar le găzduiesc luni sau ani, rămânând invizibili. Pentru a preveni acest lucru, este necesar să se înregistreze și să urmărească comportamentul unei aplicații web pentru a recunoaște activitatea suspectă în timp util și fie pentru a preveni un atac, fie pentru a minimiza consecințele acestuia.

11. ****Injecție de comenzi de siste****

**Injecția de comenzi de sistem, cunoscută și sub numele de „Command Injection”, este o vulnerabilitate de securitate care apare atunci când un atacator reușește să execute comenzi arbitrare pe sistemul de operare al unui server prin intermediul unei aplicații web. Această vulnerabilitate este posibilă datorită unei validări insuficiente a intrărilor utilizatorului, permițând astfel introducerea și executarea de comenzi sistem care nu au fost intenționate de către dezvoltatorul aplicației.**

### **Cum funcționează atacul?**

1. **Input nesecurizat**: Totul începe când aplicația primește un input de la utilizator, de exemplu printr-un formular web sau printr-un parametru URL, și nu validează sau nu curăță acest input pentru a elimina elementele potențial dăunătoare.
2. **Construirea comenzilor**: Atacatorul structurează inputul astfel încât să includă comenzi de sistem valide, adesea prefăcându-se că este un input legitim. Aceasta poate include folosirea de caractere speciale care modifică fluxul comenzii inițiale a aplicației, cum ar fi semnele de punctuație sau operatorii de control specifici sistemului de operare (de exemplu, ;, &&, | în Linux/Unix).
3. **Execuția comenzii**: Atunci când aplicația web procesează acest input, parser-ul interpretă și execută comanda sistem încorporată fără să-și dea seama că a fost manipulat. Aceasta permite atacatorului să execute orice comandă pe serverul pe care rulează aplicația, în limitele permisiunilor acordate procesului aplicației.

### ****Exemplu de atac****

Un exemplu simplu ar putea fi un formular web care solicită utilizatorului să introducă un nume de fișier pentru a fi verificat pe server. Dacă formularul este vulnerabil la injecție de comenzi, un utilizator rău intenționat ar putea introduce filename.txt; rm -rf /, care ar putea duce la executarea comenzii de ștergere a fișierelor (rm -rf /), potențial ștergând conținutul serverului.

### Prevenire

Prevenirea injecțiilor de comenzi necesită validarea strictă și sanitizarea tuturor intrărilor utilizatorilor, precum și limitarea drepturilor procesului aplicației web la strictul necesar. Este de asemenea recomandată utilizarea unor metode de execuție a comenzilor care nu permit inserția de comenzi suplimentare, cum ar fi funcții specifice limbajului de programare care execută comenzi individuale fără a interpreta operatorii de comandă (de exemplu, utilizarea listelor de argumente în locul stringurilor complete de comandă). Implementarea unei politici stricte de securitate pentru procesele și utilizatorii sistemului poate reduce și mai mult riscurile asociate cu această vulnerabilitate.

**12. Securitate slabă la nivel de API:**

**Securitatea slabă la nivelul API-urilor (Interfețe de Programare a Aplicațiilor) este o problemă frecventă în dezvoltarea aplicațiilor web și mobile moderne. API-urile facilitează comunicarea între diferite componente software, dar, dacă nu sunt bine securizate, pot deveni puncte vulnerabile care permit accesul neautorizat la datele și funcționalitățile sistemului.**

Atacurile asupra API-urilor pot varia foarte mult în funcție de natura și complexitatea API-ului, dar în general, acestea exploatează deficiențele în implementarea securității API-ului. Iată câteva exemple comune:

1. **Autentificare și autorizare insuficiente**: Dacă un API nu gestionează corect autentificarea și autorizarea, atacatorii pot accesa resurse restricționate. De exemplu, un API care nu verifică dacă un utilizator are dreptul să acceseze o anumită funcție sau set de date poate fi exploatat pentru a obține acces la informații confidențiale.
2. **Expoziția excesivă a datelor**: Un API care returnează mai multe date decât sunt necesare pentru o operație specifică poate permite atacatorilor să colecteze date sensibile care altfel nu ar fi accesibile.
3. **Lipsa ratelor de limitare**: Fără limitări adecvate ale ratei de cereri, un API poate fi vulnerabil la atacuri de tip DDoS (Distributed Denial of Service) sau brute-force, unde un atacator încearcă repetat să ghicească credențialele sau să exploateze alte vulnerabilități.
4. **Management slab al sesiunilor**: API-urile care nu gestionează corespunzător sesiunile (de exemplu, folosind token-uri de sesiune expirate sau ușor de furat) pot permite atacatorilor să preia controlul sesiunilor utilizatorilor.

### **Exemple de atacuri și impactul lor**

Un exemplu tipic este "Broken Object Level Authorization" (BOLA), unde un atacator modifică ID-ul unui obiect într-o cerere API pentru a accesa date la care nu ar trebui să aibă acces. Dacă API-ul nu verifică corespunzător dacă utilizatorul are permisiunea de a accesa obiectul respectiv, acesta poate fi exploatat pentru a accesa datele altor utilizatori.

Un alt exemplu este "Injection", unde atacatorii introduc inputuri malițioase (cum ar fi cod SQL, comenzi OS, sau scripturi XSS) prin API-uri, care sunt apoi executate neintenționat de server, ducând la pierderea integrității datelor sau compromiterea serverului.

### **Strategii de prevenire:**

Securizarea API-urilor implică mai multe strategii, inclusiv:

* **Implementarea autentificării și autorizării robuste**, folosind standarde precum OAuth, OpenID Connect și JSON Web Tokens (JWT) pentru gestionarea accesului.
* **Validarea și sanitizarea strictă a tuturor inputurilor** primate de la clienți pentru a preveni atacurile de tip injection.
* **Implementarea ratelor de limitare** pentru a preveni abuzurile și a reduce riscul atacurilor de tip DDoS.
* **Securizarea comunicațiilor** prin utilizarea TLS pentru a cripta datele transmise între clienți și servere.
* **Audierea regulată și testarea penetrării** pentru a identifica și remedia vulnerabilitățile înainte ca acestea să fie exploatate de atacatori.

**Concluzie**:

Pe măsură ce dependența de sistemele digitale crește, la fel crește și necesitatea de a proteja aceste sisteme împotriva amenințărilor tot mai sofisticate și variate.

Atacurile descrise, de la injecții SQL până la exploatarea API-urilor nesecurizate, ilustrează varietatea portilor prin care atacatorii pot pătrunde în sisteme, subliniind nevoia de vigilență și adaptare continuă a strategiilor de securitate. Metodele de prevenție și mitigare, cum ar fi validarea strictă a datelor, autentificarea multifactorială, criptarea datelor sensibile, și monitorizarea activității suspecte, sunt esențiale pentru a contracara aceste riscuri.

Referatul evidențiază, de asemenea, rolul critic al educației și al actualizărilor continue în domeniul securității cibernetice. Dezvoltatorii și administratorii de sisteme trebuie să fie mereu la curent cu cele mai noi practici, vulnerabilități descoperite și soluții de securitate pentru a menține apărările eficiente împotriva atacurilor.

În concluzie, securitatea aplicațiilor web nu este o sarcină ce poate fi completă sau definitivă, ci un proces continuu de îmbunătățire și adaptare. Adoptarea și implementarea riguroasă a standardelor OWASP nu doar că îmbunătățește securitatea aplicațiilor web, ci servește ca o fundație solidă pentru construirea unui mediu digital mai sigur și mai rezistent.

**Bibliografie**:

1. **Open Web Application Security Project (OWASP)**
   * OWASP. (2023). OWASP Top 10 - 2023: The Ten Most Critical Web Application Security Risks. [Online]. Available: https://owasp.org/www-project-top-ten/
2. **Articole și ghiduri specifice**
   * OWASP. (2023). SQL Injection Prevention Cheat Sheet. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/SQL\_Injection\_Prevention\_Cheat\_Sheet.html
   * OWASP. (2023). XXE Prevention Cheat Sheet. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/XML\_External\_Entity\_Prevention\_Cheat\_Sheet.html
   * OWASP. (2023). Authentication Cheat Sheet. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Authentication\_Cheat\_Sheet.html
3. **Literatură academică și cărți**
   * Shostack, A. (2014). Threat Modeling: Designing for Security. Wiley.
     + Această carte oferă o perspectivă detaliată asupra modelării amenințărilor, o componentă esențială în identificarea și mitigarea vulnerabilităților în aplicațiile web.
   * Howard, M., & Lipner, S. (2006). The Security Development Lifecycle. Microsoft Press.
     + Oferă o viziune detaliată asupra ciclului de viață al dezvoltării securității, un proces crucial pentru crearea de software sigur.
4. **Documente de standardizare și guidelines**
   * National Institute of Standards and Technology (NIST). (2023). NIST Special Publication 800-53 (Rev. 5) Security and Privacy Controls for Information Systems and Organizations. [Online]. Available: <https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final>
   * ISO/IEC 27001:2013. Information technology — Security techniques — Information security management systems — Requirements.