Departamentul Informatică și Ingineria Sistemelor

Disciplina: SECURITATEA PRODUSELOR PROGRAM (SPP)
SOFTWARE SECURITY (SS)

STUDII SUPERIOARE DE MASTER, CICLUL II

Prelegeri –20 ore
Lucrări practice/Lucrări de laborator -20 ore
(total 5 credite)

gr. ȘD-241M
Disciplina SPP pe ELSE: *SD.M.SPP24.1*

Titularul disciplinei: conf. univ., dr Kulev Mihail (prelegeri, l. practice/lab.)

Note de curs

T1. Introducere în securitatea software-ului
T2. Principiile de Bază ale Securității Produselor Program

- Definiții, importanța securității software
- Principalele amenințări și vulnerabilități
- Confidențialitate, integritate, disponibilitate (CIA)
- Principiul privilegiului minim, separarea responsabilităților, principiul apărării în profunzime**.**

Obiectivele prelegerii

- Înțelegerea importanței securității software
- Familiarizarea cu principalele amenințări și vulnerabilități
- Conștientizarea riscurilor și impactului atacurilor cibernetice

---

1. Definiția și importanța Securității Software

- Ce este securitatea software (securitatea produselor program)?
- Diferența dintre securitatea software și securitatea informațională
- Exemple de atacuri majore și impactul acestora asupra companiilor și utilizatorilor

Securitatea produselor program (Software Security) este domeniul științei calculatoarelor și disciplina care se ocupă cu protecția produselor program (aplicațiilor și programelor) împotriva atacurilor, vulnerabilităților și accesului neautorizat. **Scopul principal este de a asigura că software-ul funcționează corect și nu poate fi exploatat de atacatori.** Într-o lume digitală tot mai complexă, securitatea devine un element esențial al dezvoltării software-ului.

Securitatea informațională (Information Security) are un domeniu mai larg și se referă la protecția **datelor** (indiferent de unde sunt stocate sau procesate) împotriva accesului neautorizat, modificării sau distrugerii. Aceasta include **securitatea software, securitatea hardware, securitatea rețelelor și protecția utilizatorilor**.

Securitatea software și securitatea informațională sunt două domenii strâns legate, dar cu obiective și abordări diferite.

Diferențele cheie

Caracteristică	Securitatea software	Securitatea informațională
Obiectiv	Protejarea softului de vulnerabilități	Protejarea datelor împotriva accesului neautorizat
Domeniu	Se concentrează pe cod, algoritmi, arhitectură software	Include software, hardware, rețele, politici de acces
Amenințări	Exploatări de vulnerabilități, atacuri de tip SQL Injection, buffer overflow, XSS	Scurgerea de date, atacuri DDoS, phishing, inginerie socială
Metode de protecție	Secure coding, audit de securitate, patch-uri, testare de penetrare	Criptare, autentificare multi-factor, firewall-uri, monitorizare activă
Exemplu de atac	Un atacator exploatează o vulnerabilitate de buffer overflow într-o aplicație pentru a obține acces la sistem	O bază de date nesecurizată expune datele utilizatorilor pe internet

De ce este importantă securitatea software?

Securitatea software este **o componentă a securității informaționale**. Dacă software-ul este vulnerabil, poate compromite securitatea întregului sistem.

Exemplu: O aplicație web care nu validează corect datele de intrare poate permite unui atacator să execute **SQL Injection**, obținând acces la baza de date. Aceasta afectează atât **securitatea software** (cod nesigur), cât și **securitatea informațională** (date compromise).

- **Securitatea software** - *protejează aplicațiile* împotriva exploatării vulnerabilităților.
- **Securitatea informațională** - *protejează datele și infrastructura IT.*
- **O securitate software slabă poate compromite securitatea informațională a unei organizații.**

- Consecințele atacurilor: pierderi de date, pierderi financiare, afectarea reputației.
- Exemple de atacuri majore:
    - **Equifax (2017)** – scurgerea datelor a 147 milioane de utilizatori
    - **Yahoo (2013–2014)** – compromiterea a 3 miliarde de conturi
    - **SolarWinds (2020)** – atac asupra lanțului de aprovizionare cu software
    - În 2021, un atac asupra companiei **Colonial Pipeline** a blocat livrarea de combustibil în SUA
    - În 2017, virusul ***WannaCry*** a criptat fișierele a peste 230.000 de computere.
- Tendințe actuale: atacurile devin tot mai sofisticate, impunând măsuri de protecție avansate.

2. Principalele amenințări și vulnerabilități

- Clasificarea amenințărilor:
    - Amenințări interne vs. externe
    - Atacuri active vs. pasive

2.1 Tipuri de amenințări

- **Interne** (atacuri din interiorul organizației, erori umane).
- **Externe** (hackeri, malware (softul malițios), atacuri asupra infrastructurii).
- **Atacuri de tip DoS** (Denial-of-Service) **și DdoS** (Distributed Denial-of-Service) - serverele sunt suprasolicitate și devin inaccesibile .
- **Phishing și inginerie socială** – manipularea psihologică a utilizatorilor cu scopul obținerii de informație confedențială:
    - Primești un email fals care îți cere să îți schimbi parola
    - Un „tehnician IT" te sună și îți cere datele de autentificare

2.2 Vulnerabilități comune

- **Erori de programare** (Vulnerabilități de implementare)
    - Depășirea bufferului (Buffer Overflow) - o eroare care permite hackerilor să preia controlul asupra memoriei sistemului.
    - Gestionarea incorectă a excepțiilor.
- **Probleme de configurare** (Vulnerabilități de configurare)
    - Conturi implicite nesigure.
    - Drepturi de acces incorecte.
- **Vulnerabilități logice**
    - Probleme de autentificare și autorizare.
    - Procesarea incorectă a datelor utilizatorilor.

3. Tipuri comune de atacuri asupra software-ului

- **Inginerie socială**
    - Phishing, vishing, smishing
- **Exploatarea vulnerabilităților aplicațiilor** - Atacuri asupra aplicațiilor web

• SQL Injection – injectarea de interogări SQL malițioase (atacatorii obțin acces la baza de date).

• Cross-Site Scripting (XSS) – injectarea de scripturi periculoase (atacatorii execută cod JavaScript malițios în browserul utilizatorului).

• Cross-Site Request Forgery (CSRF) – efectuarea unor acțiuni în numele victimei.

- **Atacuri asupra sistemelor și rețelelor**
    - Denial-of-Service (DoS) și Distributed Denial-of-Service (DDoS)
    - Man-in-the-Middle (MitM) – interceptarea și modificarea datelor transmise.
    - DNS Spoofing – redirecționarea utilizatorilor către site-uri false.

4. Standardele, fundamentele și bunele practici de securitate software

- **OWASP Top 10: Cele mai comune vulnerabilități ale aplicațiilor web**

OWASP (Open Web Application Security Project) publică periodic lista **Top 10 vulnerabilități ale aplicațiilor web**, oferind o perspectivă asupra celor mai grave riscuri de securitate întâlnite în practică. Această listă este actualizată la fiecare câțiva ani, pe baza analizelor comunității de securitate cibernetică.

---

1. A01:2021 – Broken Access Control (Control de acces deficitar)

Prevalență: Foarte ridicată (94% dintre aplicațiile testate au avut probleme de control al accesului).

Descriere:

- Utilizatorii neautorizați pot accesa date sau funcționalități restricționate.
- Lipsa validării corecte a permisiunilor.

Exemplu:

- Un utilizator obișnuit poate accesa panoul de administrare prin manipularea URL-ului:

```
html

https://site.com/admin/dashboard
```

- Aplicația nu verifică dacă utilizatorul are rol de administrator.

Măsuri de protecție:

- Implementarea verificărilor stricte pentru permisiuni.
- Utilizarea listelor de control al accesului (ACL).
- Testarea regulată a mecanismelor de acces.

---

2. A02:2021 – Cryptographic Failures (Erori criptografice)

Prevalență: 60% dintre aplicațiile web testate au avut probleme de criptare.

Descriere:

- Datele sensibile nu sunt criptate corespunzător.
- Folosirea algoritmilor de criptare slabi sau a cheilor nesigure.

Exemplu:

- Stocarea parolelor în format **plaintext** în baza de date.

Măsuri de protecție:

- Folosirea algoritmilor puternici (AES-256, bcrypt, Argon2).
- Implementarea TLS pentru comunicațiile securizate.
- Evitarea algoritmilor depășiți (MD5, SHA-1).

---

3. A03:2021 – Injection (Injecții de cod – SQL, NoSQL, OS, LDAP, etc.)

Prevalență: 94% dintre aplicații sunt vulnerabile la cel puțin un tip de atac prin injecție.
Descriere:

- Atacatorii injectează cod malițios într-o aplicație pentru a compromite sistemul.

Exemplu (SQL Injection):
Utilizatorul introduce în formularul de login:

```
Sql

' OR 1=1 --
```

Aplicația execută:

```
Sql

SELECT * FROM users WHERE username = '' OR 1=1 --' AND password = '';
```

Atacatorul se autentifică fără a avea o parolă validă.

Măsuri de protecție:

- Folosirea **Prepared Statements** și ORM-uri pentru accesul la baze de date.
- Validarea strictă a datelor de intrare.

---

4. A04:2021 – Insecure Design (Design nesigur)

Prevalență: Crește pe măsură ce aplicațiile devin mai complexe.

Descriere:

- Arhitectura software permite exploatarea vulnerabilităților.
- Lipsa unei abordări „security by design”.

Exemplu:

- O aplicație permite resetarea parolei fără validarea identității utilizatorului.

Măsuri de protecție:

- Utilizarea principiului **"least privilege"**.
- Modele de securitate clar definite în faza de proiectare.

---

5. A05:2021 – Security Misconfiguration (Configurare incorectă a securității)

Prevalență: Foarte comună, peste 80% dintre aplicații au probleme de configurare.

Descriere:

- Servere web configurate greșit.
- Expunerea setărilor de debug în producție.

Exemplu:

- Păstrarea setărilor implicite în serverul Apache/Nginx.

Măsuri de protecție:

- Dezactivarea modului **debug** pe serverele live.
- Revizuirea periodică a configurațiilor de securitate.

---

6. A06:2021 – Vulnerable and Outdated Components (Componente vulnerabile și învechite)

Prevalență: 90% dintre aplicații folosesc componente depășite.

Descriere:

- Folosirea versiunilor vechi ale librăriilor/framework-urilor.

Exemplu:

- Aplicații care rulează pe **Apache Struts 2.3.x**, vulnerabil la Remote Code Execution (CVE-2017-5638).

Măsuri de protecție:

- Actualizarea constantă a bibliotecilor și framework-urilor.

---

7. A07:2021 – Identification and Authentication Failures (Eșecuri de autentificare și identificare)

Prevalență: 75% dintre aplicații au probleme de autentificare.

Descriere:

- Autentificare slabă sau lipsa protecției conturilor utilizatorilor.

Exemplu:

- Sistemele care permit parole slabe (ex: "123456").

Măsuri de protecție:

- Implementarea autentificării multi-factor (MFA).
- Stocarea securizată a parolelor (bcrypt, Argon2).

---

8. A08:2021 – Software and Data Integrity Failures (Probleme de integritate a software-ului și datelor)

Prevalență: În creștere odată cu atacurile de tip supply chain.

Descriere:

- Lipsa verificării integrității software-ului descărcat.

Exemplu:

- Descărcarea unui fișier de instalare compromis.

Măsuri de protecție:

- Utilizarea semnăturilor digitale și a verificării hash-urilor.

---

9. A09:2021 – Security Logging and Monitoring Failures (Eșecuri în jurnalizare și monitorizare)

Prevalență: 80% dintre aplicații nu monitorizează corect activitatea utilizatorilor.

Descriere:

- Lipsa log-urilor de securitate sau monitorizare insuficientă.

Exemplu:

- Un atacator încearcă 1000 de parole pe un cont, dar sistemul nu detectează atacul.

Măsuri de protecție:

- Implementarea alertelor de securitate.

---

10. A10:2021 – Server-Side Request Forgery (SSRF – Falsificarea cererilor de pe server)

Prevalență: În creștere, în special la serviciile cloud.

Descriere:

- Atacatorii trimit cereri HTTP malițioase către servere interne.

Exemplu:

- Exploatarea unei aplicații pentru a accesa resurse interne (ex: baze de date private).

Măsuri de protecție:

- Restricționarea accesului la resursele interne.

Concluzie

- **OWASP Top 10** reprezintă cele mai grave și comune probleme de securitate ale aplicațiilor web.
- Fiecare dintre aceste vulnerabilități poate duce la **compromiterea completă a unei aplicații**.
- Implementarea **practicilor de codare securizată și testarea continuă** sunt esențiale pentru prevenirea atacurilor.

- Modelul CIA (Confidențialitate, Integritate, Disponibilitate) în Securitatea Software

Modelul **CIA** (Confidentiality, Integrity, Availability) este un **principiu fundamental în securitatea software**, fiind utilizat pentru a asigura protecția aplicațiilor, datelor și sistemelor informatice. În contextul **securității software**, acest model se aplică pentru a preveni atacurile, a gestiona accesul utilizatorilor și a proteja datele aplicațiilor împotriva modificărilor neautorizate.

---

1. Confidențialitate (Confidentiality) în Securitatea Software

Ce înseamnă?

Confidențialitatea asigură că **doar utilizatorii autorizați** pot accesa anumite date sau funcționalități ale aplicației. Protejează informațiile sensibile împotriva accesului neautorizat.

Amenințări comune în software:

- **Expunerea datelor sensibile** (ex: parole, informații financiare, date personale).
- **Atacuri de tip SQL Injection** (ex: atacatorii extrag date prin interogări malițioase).
- **Interceptarea comunicațiilor (Man-in-the-Middle)** (ex: atacatorii interceptează traficul între client și server).
- **Stocarea parolelor în format neprotejat** (ex: parole salvate în format plaintext).

Măsuri de protecție în software:

- **Criptarea datelor sensibile** (ex: AES-256 pentru date, hashing cu bcrypt pentru parole).
- **Autentificare multifactor (MFA)** (ex: SMS, aplicații de autentificare, token-uri hardware).
- **Control strict al accesului** (ex: RBAC – Role-Based Access Control).
- **Utilizarea TLS/SSL pentru comunicațiile între client și server**.

Exemplu practic:

Aplicație de banking online

- Clienții trebuie să se autentifice cu **parolă + cod SMS (MFA)**.
- Tranzacțiile sunt criptate cu **TLS 1.3**.
- Datele utilizatorilor sunt criptate la stocare cu **AES-256**.

---

2. Integritate (Integrity) în Securitatea Software

Ce înseamnă?

Integritatea asigură că **datele și codul sursă al aplicației nu sunt alterate neautorizat**. Se referă la menținerea corectitudinii informațiilor și prevenirea modificărilor malițioase sau accidentale.

Amenințări comune în software:

- **Modificarea codului sursă prin atacuri de tip Code Injection**.
- **Manipularea datelor în tranzit (Man-in-the-Middle Attack)**.
- **SQL Injection** – atacatorii modifică sau șterg date din baza de date.
- **Modificarea fișierelor de configurare sau a datelor logate**.

Măsuri de protecție în software:

- **Utilizarea semnăturilor digitale** pentru verificarea autenticității codului și a datelor.
- **Implementarea mecanismelor de versionare și backup** pentru prevenirea pierderii sau alterării datelor.
- **Verificarea integrității fișierelor** cu hash-uri (SHA-256, SHA-3).
- **Utilizarea ORM (Object-Relational Mapping)** pentru prevenirea SQL Injection.

Exemplu practic:

Platformă de comerț electronic

- Fiecare tranzacție este semnată digital pentru a preveni modificarea sumelor de plată.
- Sistemul loghează și verifică toate modificările asupra bazei de date.
- Un atac SQL Injection este prevenit prin utilizarea **prepared statements** în interogările SQL.

---

3. Disponibilitate (Availability) în Securitatea Software

Ce înseamnă?

Disponibilitatea garantează că **aplicația și serviciile software sunt accesibile utilizatorilor atunci când sunt necesare**, fără întreruperi cauzate de atacuri, erori sau defecțiuni.

Amenințări comune în software:

- **Atacuri DDoS (Distributed Denial of Service)** – serverele sunt suprasolicitate și devin inaccesibile.
- **Eroare de infrastructură** (ex: căderea serverelor, defecte hardware).
- **Bug-uri critice în software** care provoacă crash-uri sau downtime.
- **Blocarea accidentală a utilizatorilor legitimi din cauza unei erori în sistemul de autentificare**.

Măsuri de protecție în software:

- **Utilizarea infrastructurilor de tip load balancing** pentru distribuirea traficului.
- **Soluții anti-DDoS (ex: Cloudflare, AWS Shield, Arbor Networks)**.
- **Backup-uri regulate și planuri de recuperare în caz de dezastru (DRP – Disaster Recovery Plan)**.
- **Monitorizare activă și alertare în timp real pentru downtime**.

Exemplu practic:

Serviciu de streaming video (ex: Netflix, YouTube)

- Serverele sunt distribuite global pentru a preveni suprasolicitarea unei singure locații.
- Se folosesc rețele de distribuție a conținutului (CDN – Content Delivery Networks).
- În caz de atac DDoS, sistemele de protecție blochează cererile malițioase fără a afecta utilizatorii legitimi.

Importanța echilibrului CIA în securitatea software

Un sistem software trebuie să mențină un **echilibru între Confidențialitate, Integritate și Disponibilitate**. Dacă un principiu este prioritizat excesiv, poate compromite celelalte două.

Aspect	Compromis posibil
Confidențialitate vs. Disponibilitate	Un sistem extrem de securizat poate bloca accesul utilizatorilor legitimi.
Disponibilitate vs. Integritate	Un sistem care prioritizează disponibilitatea poate permite modificări neautorizate ale datelor.
Confidențialitate vs. Integritate	Un sistem care pune accent pe confidențialitate poate întârzia detectarea modificărilor neautorizate.

Concluzie

- **Modelul CIA este esențial în securitatea software** pentru protejarea aplicațiilor și datelor.
- **Confidențialitatea** protejează datele împotriva accesului neautorizat.
- **Integritatea** asigură corectitudinea datelor și a codului software.
- **Disponibilitatea** garantează accesul neîntrerupt la servicii și aplicații.
- **Un echilibru între aceste trei principii este necesar pentru un software securizat.**

- **Principiul privilegiului minim (conceptul de least privilege) în Securitatea Software**

Principiul **privilegiului minim** (Least Privilege) este un concept fundamental în **securitatea software**, care stipulează că **un utilizator, proces sau sistem ar trebui să aibă doar permisiunile strict necesare pentru a îndeplini o anumită sarcină și nimic mai mult**.

- Această strategie reduce suprafața de atac a unei aplicații și limitează **impactul unei breșe de securitate** sau al unei erori accidentale.

Importanța Principiului Privilegiului Minim în Securitatea Software

🔹 **Minimizează riscurile de securitate** – restricționarea accesului reduce posibilitatea ca un atacator să exploateze o vulnerabilitate.

🔹 **Previne escaladarea privilegiilor** – chiar dacă un atacator compromite un cont sau un proces, acesta va avea acces doar la resursele minime necesare.

🔹 **Îmbunătățește controlul și auditul** – accesul limitat face mai ușoară monitorizarea activităților suspecte.

🔹 **Reduce impactul erorilor umane** – utilizatorii nu pot modifica sau șterge accidental date critice dacă nu au permisiuni administrative.

Aplicarea Principiului Privilegiului Minim în Securitatea Software

1. Controlul Accesului pentru Utilizatori

- **Exemplu:** Într-un sistem de gestionare a bazelor de date, utilizatorii obișnuiți ar trebui să aibă doar **acces de citire**, iar doar administratorii să poată **modifica sau șterge date**.
- **Greșeală comună:** Acordarea tuturor utilizatorilor acces de tip „Administrator” pentru a evita probleme de permisiuni.

---

2. Permisiuni Minime pentru Procese și Aplicații

- **Exemplu:** Un server web (ex: Apache, Nginx) ar trebui să ruleze cu un **utilizator cu privilegii restricționate**, nu ca **root**, pentru a preveni compromiterea întregului sistem.
- **Greșeală comună:** Rularea aplicațiilor critice cu **privilegii administrative**, ceea ce poate permite atacatorilor să preia controlul complet asupra sistemului în caz de exploatare.

---

3. Restricționarea Accesului la Resursele Critice

- **Exemplu:** O aplicație web care accesează baza de date ar trebui să aibă doar **permisiuni de citire/scriere** pentru tabelele necesare și să nu poată **șterge datele critice**.
- **Greșeală comună:** Crearea unui cont de bază de date cu **toate privilegiile activitate** („GRANT ALL”) pentru fiecare componentă a aplicației.

---

4. Separarea Conturilor de Utilizator și a Mediilor

- **Exemplu:** Un dezvoltator ar trebui să aibă un cont standard pentru sarcinile zilnice și un cont separat cu **privilegii administrative** utilizat doar atunci când este necesar.
- **Greșeală comună:** Folosirea unui **singur cont de administrator** pentru toate activitățile, inclusiv navigarea pe internet, ceea ce poate expune sistemul la malware.
-

---

5. Minimizarea Utilizării Serviciilor și Bibliotecilor Externe

- **Exemplu:** O aplicație ar trebui să **încarce doar modulele și bibliotecile strict necesare** pentru a funcționa, evitând dependențele inutile care pot introduce vulnerabilități.
- **Greșeală comună:** Importarea și utilizarea unor pachete externe fără a verifica dacă sunt actualizate și sigure.

Implementarea Practică a Principiului Privilegiului Minim

Măsură	Descriere
RBAC (Role-Based Access Control)	Acordarea permisiunilor bazate pe rolurile utilizatorilor.
Principiul separării sarcinilor (Separation of Duties - SoD)	Oferirea accesului doar la resursele esențiale pentru îndeplinirea unei sarcini.
Utilizarea listelor de control al accesului (ACLs)	Definirea regulilor specifice pentru ce utilizatori/procese pot accesa resursele.

Măsură	Descriere
Audit și logare	Monitorizarea accesului și a modificărilor pentru a detecta încercările de acces neautorizat.
Principiul "Need-to-Know"	Limitarea accesului utilizatorilor doar la datele necesare sarcinilor lor.
Reducerea privilegiilor aplicațiilor	Aplicațiile trebuie să ruleze cu permisiuni minime, evitând rularea ca administrator/root.

Exemplu Practic de Configurare a Privilegiilor Minime în MySQL

- **Scenariu:** Un cont de utilizator al aplicației trebuie să aibă acces doar la citirea și scrierea datelor într-o anumită tabelă, fără drept de ștergere sau modificare a structurii bazei de date.
- **Greșit:**

```
sql
GRANT ALL PRIVILEGES ON my_database.* TO 'app_user'@'localhost';
```

- Aceasta acordă **toate privilegiile**, inclusiv ștergerea tabelelor și modificarea structurii bazei de date.
- **Corect:**

```
sql
GRANT SELECT, INSERT, UPDATE ON my_database.orders TO
'app_user'@'localhost';
```

- Contul `app_user` poate doar **citi, insera și actualiza** date în tabelul `orders`, fără a putea șterge date sau baza de date.

Beneficiile Principiului Privilegiului Minim

- **Reducerea impactului atacurilor** – în cazul unui atac, un cont cu permisiuni limitate va reduce accesul atacatorului.
- **Protecția împotriva erorilor umane** – un utilizator nu poate șterge accidental date critice.
- **Îmbunătățirea securității generale** – sistemele devin mai rezistente la breșe de securitate.
- **Conformitate cu reglementările** – standarde precum **ISO 27001, GDPR, NIST** impun implementarea accesului minim necesar.

Concluzie

🔷 Principiul **Privilegiului Minim** este un concept esențial în **securitatea software**, reducând riscurile de atac și protejând datele critice.
🔷 Aplicarea corectă a acestui principiu limitează accesul utilizatorilor și proceselor doar la **resursele strict necesare**, prevenind exploatarea vulnerabilităților.
🔷 Implementarea acestui principiu implică **gestionarea corectă a permisiunilor**,

separarea conturilor, **auditarea accesului** și **configurarea atentă a resurselor software**.

- **Principiul apărării în profunzime (**conceptul de defense-in-depth) în Securitatea Software

Apărarea în profunzime (**Defense-in-Depth**) este o strategie de securitate care presupune implementarea **mai multor niveluri de protecție** pentru a asigura securitatea unui sistem software. Dacă un strat de securitate este compromis, **există alte niveluri de protecție care pot preveni sau limita daunele**.

◈ Este o metodă folosită în securitatea cibernetică pentru a **crea redundanță** și pentru a minimiza riscurile de atac.
◈ Se bazează pe ideea că **nu există o soluție unică care să asigure securitatea completă a unui sistem**.
◈ Este inspirat din strategiile militare, unde protecția nu se bazează pe un singur punct de apărare, ci pe **mai multe bariere succesive**.

De ce este importantă Apărarea în Profunzime?

- **Reducerea suprafeței de atac** – prin aplicarea mai multor straturi de securitate, atacatorii întâmpină obstacole multiple.
- **Limitarea impactului unui atac** – dacă un strat de securitate e compromis, celelalte pot limita efectele.
-**Creșterea timpului de detecție și reacție** – sistemele pot detecta atacurile în faze incipiente.
- **Conformitate cu standardele de securitate** – multe reglementări (ex: GDPR, ISO 27001, NIST) impun măsuri multiple de protecție.

Principalele straturi ale Apărării în Profunzime în Securitatea Software

Un sistem securizat folosește mai multe **niveluri de protecție**, care pot fi grupate astfel:

1. Protecția Perimetrală (Network Security)

Obiectiv: Prevenirea accesului neautorizat la rețeaua sistemului.

Măsuri de protecție:
- **Firewall-uri** – blochează traficul malițios.
- **Sisteme de prevenire a intruziunilor (IPS/IDS)** – monitorizează traficul și detectează atacurile.
- **Segregarea rețelei (Network Segmentation)** – separarea rețelei interne de cea publică.
- **VPN și criptare pentru comunicații** – protejează traficul împotriva interceptării.

2. Securitatea Aplicației (Application Security)

Obiectiv: Prevenirea exploatării vulnerabilităților software.

Măsuri de protecție:
- **Validarea și filtrarea input-urilor** – prevenirea atacurilor de tip SQL Injection, XSS.
- **Utilizarea unui framework de securitate** – protecția codului împotriva vulnerabilităților comune.
- **Autentificare și autorizare strictă** – implementarea MFA (Multi-Factor Authentication).
- **Politici de actualizare și patching** – remedierea rapidă a vulnerabilităților cunoscute.

---

3. Protecția Datelor (Data Security)

Obiectiv: Protejarea informațiilor stocate și transferate împotriva accesului neautorizat.

Măsuri de protecție:
- **Criptarea datelor la stocare și în tranzit** – AES-256, TLS.
- **Controlul accesului bazat pe roluri (RBAC – Role-Based Access Control)**.
- **Tokenizare și hashing pentru parole** – bcrypt, Argon2.
- **Back-up regulat și plan de recuperare** – prevenirea pierderii datelor.

---

4. Protecția la Nivelul Utilizatorului (User Security)

Obiectiv: Prevenirea atacurilor care exploatează utilizatorii umani.

Măsuri de protecție:
- **Autentificare multifactorială (MFA)** – protejează conturile.
- **Limitarea privilegiilor utilizatorilor (Principiul Privilegiului Minim)**.
- **Sesiuni cu expirare automată și mecanisme anti-brute force**.
- **Training de securitate pentru utilizatori** – conștientizarea atacurilor de tip phishing, inginerie socială.

---

5. Monitorizare și Detectare (Security Monitoring)

Obiectiv: Identificarea și reacția rapidă la incidente de securitate.

Măsuri de protecție:
- **Sisteme de logare și audit** – monitorizarea accesului și a activităților suspecte.
- **Monitorizarea activității utilizatorilor și a traficului de rețea**.

- **Răspuns la incidente și echipe de securitate SOC (Security Operations Center)**.
- **Detectarea anomaliilor prin AI și machine learning**.

Exemplu Practic de Apărare în Profunzime

Scenariu: O companie dezvoltă o aplicație web pentru gestionarea datelor financiare ale clienților.

Fără Apărare în Profunzime:

- Aplicația nu folosește HTTPS.
- Oricine poate accesa baza de date direct.
- Conturile utilizatorilor nu au protecție MFA.
- Lipsa unui sistem de detecție a atacurilor.

Cu Apărare în Profunzime:

◈ **Protecția rețelei:** Serverele sunt protejate cu firewall-uri și sisteme de prevenire a atacurilor.
◈ **Protecția aplicației:** Aplicația filtrează input-urile pentru a preveni atacurile SQL Injection și XSS.
◈ **Protecția datelor:** Datele financiare sunt criptate cu AES-256, iar parolele utilizatorilor sunt protejate cu hashing.
◈ **Securitatea utilizatorilor:** Autentificarea multifactorială (MFA) este obligatorie.
◈ **Monitorizare:** Toate evenimentele suspecte sunt înregistrate și analizate.

Diferența dintre Apărarea în Profunzime și alte strategii de securitate

Strategie	Descriere
Apărare în Profunzime	Folosește mai multe niveluri de protecție pentru a limita atacurile.
Securitate perimetrală	Se concentrează doar pe protecția inițială (ex: firewall).
Zero Trust Security	Presupune că niciun utilizator sau sistem nu este de încredere implicit.
Security by Design	Securitatea este integrată încă din faza de dezvoltare a software-ului.

Concluzie

- **Apărarea în Profunzime** este o strategie de securitate esențială care implementează **mai multe niveluri de protecție** pentru a preveni atacurile și a limita daunele.
- Un sistem sigur nu se bazează pe **o singură măsură de protecție**, ci combină **securitatea rețelei, aplicației, datelor și utilizatorilor**.
- **Fiecare strat de securitate acționează ca o barieră suplimentară**, reducând riscul ca o breșă să compromită întregul sistem.

- Aplicarea principiului **Defense-in-Depth** este necesară pentru **asigurarea confidențialității, integrității și disponibilității datelor**.

5. Concluzii și recomandări generale

- Necesitatea adoptării unei mentalități de securitate în dezvoltarea software
- Importanța implementării măsurilor de securitate
- Integrarea securității în toate etapele dezvoltării (conceptul de Secure SDLC)
- Utilizarea instrumentelor de analiză a securității (analiza statică și dinamică codului, testare de penetrare)
- Educație și formare continuă a echipei
- Respectarea reglementărilor și standardelor internaționale (GDPR, OWASP, ISO 27001)

GDPR (General Data Protection Regulation) este **Regulamentul general privind protecția datelor** adoptat de Uniunea Europeană (UE) în **2016** și aplicabil din **25 mai 2018**. Acesta stabilește un **cadru legal unificat pentru protecția datelor personale** ale cetățenilor UE și SEE (Spațiul Economic European).

Scopul GDPR este de a **proteja drepturile și libertățile persoanelor** în ceea ce privește **prelucrarea și circulația datelor lor personale**, oferindu-le mai mult control asupra modului în care sunt utilizate informațiile lor.

GDPR și Securitatea Software

GDPR impune **cerințe stricte de securitate** pentru protecția datelor personale, ceea ce afectează direct dezvoltarea software:

- **Criptarea datelor** – Stocarea și transmiterea datelor trebuie securizate.
- **Autentificare și autorizare** – Implementarea controalelor stricte de acces (ex: MFA).
- **Audit și logare** – Monitorizarea accesului și a modificărilor datelor.
- **Evaluări periodice de securitate** – Teste de penetrare, verificări de conformitate.
- **Principiul "Privacy by Design"** – Aplicațiile trebuie să fie proiectate de la început cu măsuri de protecție a datelor.

ISO/IEC 27001 este **standardul internațional pentru managementul securității informațiilor** (ISMS – Information Security Management System). Publicat de **Organizația Internațională pentru Standardizare (ISO) și Comisia Electrotehnică Internațională (IEC)**, ISO 27001 definește **cerințele pentru implementarea unui sistem de management al securității informațiilor** într-o organizație.

Scopul său este de a proteja **confidențialitatea, integritatea și disponibilitatea** informațiilor prin **măsuri tehnice și administrative**.

ISO 27001 și securitatea software

- **Protecția datelor utilizatorilor** – Respectarea cerințelor ISO 27001 ajută la securizarea **aplicațiilor software** împotriva atacurilor cibernetice.
- **Criptare și autentificare** – Implementarea unor **măsuri stricte** pentru protecția **datelor stocate și transmise**.
- **Monitorizare și logare** – Utilizarea sistemelor de **detectare a amenințărilor** și auditarea activității utilizatorilor.
- **Testare și actualizare** – Aplicațiile trebuie să fie testate și actualizate constant pentru a **remedia vulnerabilitățile**.

TLS (Transport Layer Security) este un protocol criptografic utilizat pentru **securizarea comunicațiilor** pe internet. Acesta oferă **confidențialitate, integritate și autentificare** pentru datele transmise între clienți și servere.

TLS este succesorul protocolului **SSL (Secure Sockets Layer)**, care nu mai este considerat sigur. Versiunea actuală recomandată este **TLS 1.3**.

bcrypt este un **algoritm de hashing** folosit pentru protejarea parolelor. A fost dezvoltat în 1999 de **Niels Provos și David Mazières** și este conceput pentru a fi **rezistent la atacuri brute-force și rainbow table**.

Spre deosebire de alte algoritmi de hashing (ex: **MD5, SHA-1, SHA-256**), **bcrypt include un mecanism de „încetinire"** pentru a face atacurile mai dificile odată cu creșterea puterii de calcul.

Argon2 este un **algoritm modern de hashing al parolelor**, conceput pentru a oferi **maximă securitate împotriva atacurilor brute-force și GPU**. A fost dezvoltat de **Alex Biryukov, Daniel Dinu și Dmitry Khovratovich** și a câștigat competiția **Password Hashing Competition (PHC) în 2015**, devenind standardul recomandat pentru protecția parolelor.

Argon2 este mai sigur și mai eficient decât bcrypt

CERT (Computer Emergency Response Team) este o echipă specializată în **detectarea, prevenirea și răspunsul la incidentele de securitate cibernetică**.

- **Scopul principal**: Protejarea infrastructurilor IT și a utilizatorilor împotriva atacurilor cibernetice prin monitorizare, investigație și remediere.
- **CERT acționează ca un centru de coordonare** între guverne, companii, instituții și utilizatori pentru a gestiona amenințările informatice.