Zadanie 3 UPB

1. **Kontrola zložitosti hesla pri registrácii**

   Pri registrácii používateľov je pre nás dôležité zabezpečiť, aby zadané heslo spĺňalo bezpečnostné štandardy. Tieto štandardy pomáhajú znížiť riziko, že heslo bude ľahko uhádnuté alebo prelomené rôznymi typmi útokov. Funkcia `validate_password_complexity` obsahuje niekoľko kritérií, ktoré kontrolujú zložitosť hesla, a tým zvyšujú celkovú bezpečnosť.

   V rámci funkcie `validate_password_complexity` využívame nasledujúce kritériá, aby sme zaistili, že heslo je dostatočne silné:

   1. **Minimálna dĺžka hesla:**
      - Heslo musí mať minimálne 8 znakov.
      - Toto pravidlo pomáha zvyšovať bezpečnosť, pretože dlhšie heslá sú ťažšie prelomiteľné pomocou tzv. útokov „brute force,“ kde útočník skúša všetky možné kombinácie znakov.
   2. **Veľké písmená:**
      - Heslo musí obsahovať aspoň jedno veľké písmeno.
      - Kombinácia veľkých a malých písmen zvyšuje variabilitu hesla a sťažuje útoky založené na hádaní jednoduchých a bežne používaných slov.
   3. **Malé písmená:**
      - Heslo musí obsahovať aspoň jedno malé písmeno.
      - Táto požiadavka zvyšuje variabilitu hesla a znižuje riziko, že heslo bude ľahko uhádnuté alebo prelomené.
   4. **Čísla:**
      - Heslo musí obsahovať aspoň jedno číslo.
      - Pridanie čísel do hesla zvyšuje jeho zložitosť a odolnosť voči tzv. slovníkovým útokom, ktoré často skúšajú bežné kombinácie písmen bez čísel.
   5. **Špeciálne znaky:**
      - Heslo musí obsahovať aspoň jeden špeciálny znak (napr. !, @, #, $, %, ^, &).
      - Prítomnosť špeciálneho znaku zvyšuje variabilnosť hesla a zvyšuje jeho bezpečnosť, pretože tieto znaky sa bežne nenachádzajú v jednoduchých heslách alebo bežných slovách.

```python
# Funkcia na validáciu hesla
def validate_password_complexity(form, field):  matus kuma
    password = field.data
    if len(password) < 8:
        raise ValidationError('Password must be at least 8 characters long.')
    if not re.search( pattern: r'[A-Z]', password):
        raise ValidationError('Password must contain at least one uppercase letter.')
    if not re.search( pattern: r'[a-z]', password):
        raise ValidationError('Password must contain at least one lowercase letter.')
    if not re.search( pattern: r'[0-9]', password):
        raise ValidationError('Password must contain at least one number.')
    if not re.search( pattern: r'[!@#$%^&*(),.?":{}|<>]', password):
        raise ValidationError('Password must contain at least one special character.')
```

2. Bezpečné ukladanie hesiel do databázy

Pri implementácii bezpečného ukladania hesiel do databázy sme použili manuálne kroky na generovanie a ukladanie zahashovaného hesla so soľou (salt), ktorá zvyšuje bezpečnosť používateľských údajov. Keďže je zakázané používať knižnice, ktoré priamo ponúkajú hotovú funkcionalitu, použili sme základné kryptografické primitívy a postupy na zabezpečenie hesiel. Nasleduje podrobný popis systému bezpečného ukladania hesiel.

Postup a bezpečnostné opatrenia

1. **Generovanie soli (salt) pre každé heslo:**
    - Pre každé nové heslo generujeme náhodný salt, ktorý sa pripojí k heslu pred zahashovaním.
    - Salt slúži na ochranu proti tzv. „rainbow table“ útokom, pri ktorých útočníci využívajú predpočítané tabuľky s hashmi bežne používaných hesiel. Použitím rôznych saltov pre každého používateľa zabezpečíme, že rovnaké heslá budú mať rozdielne hashe, čím sa zamedzí ich jednoduchému uhádnutiu.
    - Salt je uložený v databáze spolu s hashom v hexadecimálnom formáte.
2. **Hashovanie hesla so saltom:**
    - Hashovanie sa vykonáva spojením hesla a saltu pomocou silného hashovacieho algoritmu, ktorý vytvára unikátny a nepoužiteľný reťazec z pôvodného hesla.
    - Hash je uložený v databáze namiesto pôvodného hesla, čo zabezpečuje, že ani v prípade prístupu do databázy útočníci nezískajú heslo v čitateľnej forme.
    - Použitý hashovací algoritmus musí byť odolný voči kolíziám a mal by mať dostatočnú výpočtovú náročnosť (napr. SHA-256 alebo SHA-512), aby bolo náročné rekonštruovať pôvodné heslo.
3. **Ukladanie hashu a saltu v hexadecimálnom formáte:**
    - Oba údaje (hash aj salt) sa ukladajú v hexadecimálnom formáte, čo zabezpečuje konzistentnosť pri ukladaní binárnych údajov do databázy.
    - Tento formát umožňuje jednoduché spracovanie a správu dát bez straty informácií, ktoré sú potrebné na validáciu hesla pri prihlasovaní používateľa.

4. **Validácia hesla pri prihlasovaní:**
    - Pri prihlasovaní používateľa vyberieme z databázy uložený salt, ktorý bol použitý pri registrácii.
    - Salt pripojíme k heslu zadanému pri prihlásení a následne vytvoríme nový hash.
    - Novovytvorený hash porovnáme s hashom uloženým v databáze – ak sa hashe zhodujú, heslo je správne.

Zdôvodnenie zvoleného prístupu

Zvolený postup využíva základné kryptografické prvky na efektívne a bezpečné ukladanie hesiel bez použitia hotových knižníc s priamym generovaním hashov. Tento prístup zabezpečuje:

- **Odolnosť voči útokom:** Použitie náhodného saltu a silného hashovacieho algoritmu zabezpečuje vysokú odolnosť voči „rainbow table“ útokom a slovníkovým útokom.
- **Bezpečnosť údajov v databáze:** Uloženie hesla vo forme hashu namiesto čitateľnej podoby zabraňuje tomu, aby útočník, ktorý získa prístup do databázy, mohol získať priamo použiteľné heslá.
- **Flexibilita a bezpečnosť:** Tento systém umožňuje kedykoľvek zmeniť hashovací algoritmus alebo zvýšiť počet iterácií hashovania, čím zvyšujeme odolnosť proti narastajúcej výpočtovej sile potenciálnych útočníkov.

3. Ochrana proti Brute-Force

Na ochranu prihlasovacieho systému pred brute-force útokmi, kde útočník opakovane skúša rôzne kombinácie hesiel, sme implementovali Google reCAPTCHA. Táto ochrana zabezpečuje, že prístup k prihlasovaniu je umožnený iba skutočným používateľom, a bráni automatizovaným systémom v hromadnom testovaní hesiel. ReCAPTCHA pridáva overovaciu vrstvu, ktorá zastaví akýkoľvek programatický alebo automatizovaný pokus o prihlásenie.

Implementácia ochrany proti brute-force útokom pomocou reCAPTCHA

1. **Validácia reCAPTCHA pri prihlasovaní:**
    - Po zadaní prihlasovacích údajov používateľom kontrolujeme jeho odpoveď na reCAPTCHA pomocou parametra `g-recaptcha-response`, ktorý sa získava z formulára.
    - Tento parameter je odoslaný na overenie reCAPTCHA serveru prostredníctvom POST požiadavky s naším súkromným kľúčom `secret_key` na bezpečné overenie.
2. **Overenie reCAPTCHA na strane servera:**
    - Odpoveď reCAPTCHA (buď úspech, alebo neúspech) je spracovaná v rámci serverovej odpovede. Ak odpoveď označí používateľa ako skutočného, pokračujeme v overovaní používateľského hesla.
    - V prípade, že reCAPTCHA neprejde, používateľ dostane chybovú správu s výzvou na dokončenie reCAPTCHA.
3. **Význam implementácie reCAPTCHA proti brute-force útokom:**

- reCAPTCHA je efektívna ochrana pred útokmi brute-force, pretože obmedzuje rýchlosť a množstvo pokusov o prihlásenie – automatizovaní útočníci musia riešiť reCAPTCHA na každé prihlásenie, čo prakticky znemožňuje programatické testovanie hesiel.
- Táto technológia tiež rozpoznáva bežné vzorce správania robotov, čím zvyšuje bezpečnosť bez zbytočného narúšania prihlásenia legitímnych používateľov.

Výhody použitia reCAPTCHA proti brute-force útokom

Použitím reCAPTCHA pri prihlasovaní pridávame do systému bezpečnostnú vrstvu, ktorá odoláva programatickým útokom, no zároveň zbytočne nezaťažuje skutočných používateľov. Systém si zachováva funkčnosť, pričom poskytuje ochranu pred útokmi na heslá a potenciálnym zneužitím prihlasovacích údajov.

Pri použití reCAPTCHA na ochranu proti brute-force útokom je nutné zabezpečiť pripojenie na internet, pretože reCAPTCHA overuje používateľovu odpoveď na vzdialenom serveri Google.

4. Ochrana voči slovníkovým heslám

Na ochranu používateľských účtov sme pri registrácii implementovali kontrolu proti používaniu slovníkových hesiel – teda bežne používaných a jednoduchých hesiel, ktoré sú častým cieľom útokov. Tento krok pomáha zvýšiť bezpečnosť, pretože útočníci bežne používajú slovníkové útoky, pri ktorých testujú známe alebo ľahko uhádnuteľné heslá.

Implementácia ochrany proti slovníkovým heslám

1. **Definícia slovníka bežne používaných hesiel:**
   - Vytvorili sme zoznam bežných hesiel (`common_passwords`), ktoré často používajú neskúsení používatelia a ktoré sú jednoduché na uhádnutie (napr. „Password123!“, „Welcome@2023“, „Summer2023!”).
   - Tento zoznam obsahuje heslá, ktoré sú známe ako málo bezpečné a sú ľahko odhaliteľné pri slovníkových útokoch.
2. **Kontrola hesla pri registrácii:**
   - V rámci validačnej funkcie `validate_password_complexity` kontrolujeme, či zadané heslo nie je súčasťou zoznamu `common_passwords`.
   - Ak používateľ zadá heslo, ktoré je na zozname, registrácia je zamietnutá s chybovou hláškou („Your password is a common password“), ktorá ich informuje, že zadané heslo nie je dostatočne bezpečné.
3. **Výhody ochrany proti slovníkovým heslám:**
   - **Zvýšenie bezpečnosti:** Zamedzením používania bežne používaných hesiel odrádzame útočníkov od slovníkových útokov, kde sa pokúšajú o jednoduché, často používané kombinácie hesiel.
   - **Povzbudenie k tvorbe silných hesiel:** Používatelia sú motivovaní vytvárať heslá, ktoré sú dlhšie, variabilnejšie a ťažšie uhádnuteľné.