AUTENTIZACE
AUTORIZACE

PV179

ONDŘEJ PAVLICA

OPAKOVÁNÍ Z BC STUDIA

- **Autentizace** – ověřuji, kdo jsem, za koho se vydávám
- **Autorizace** – kontroluji, uděluji či zamítám přístup k určitému zdroji podle mé identity (která by měla být autentizována)

- Autentikace, autentifikace, apod. neexistuje, prof. Matyáš za toto rád roastuje na státnicích :D

PRINCIPY UKLÁDÁNÍ CITLIVÝCH DAT

- Posílání privátního klíče = **velký špatný**
- Pokud si data nevyměňujeme na šifrovaném spojení, tak je jedno, jak moc bezpečně data ukládáme
- Méně je někdy více – aplikovat „need to know“ princip
- Ukládat hesla je bezpečnostní riziko
  - Stačí jeho hash
  - Ještě lépe před hashováním saltovat (= přidání nějakých dat navíc – „soli“ – k heslu)
  - Každý uživatel nejlépe vlastní salt
  - MD5 ani SHA-1 nejsou bezpečné hashe na hesla

IDENTITY FRAMEWORK (CORE)

- Nejpoužívanější balíček na autentizaci a autorizaci v ASP.NETu
- Téma zbytku prezentace
- Principy z něj ale platí i v ostatních řešeních
- Jde velmi rozsáhle ohnout podle potřeby, proto informace dále budou primárně ohledně výchozí konfigurace

- Základní objekty:
  - **Uživatel**
  - **Role** – vztah M:N k uživateli
  - **Deklarace identity (claim)** – jeden konkrétní údaj, co se váže k mé osobě (např. rodné číslo)
  - **Politika (policy)** – Komplexnější způsob přístupu ke zdroji (např. politika „registrovaný plnoletý“ -> má potvrzený uživ. účet, hodnota claimu „datum narození“ > DateTime.Now – 18 let)

NASAZENÍ DO PROJEKTU

1. Potřeba nainstalovat balíčky `Microsoft.AspNetCore.Identity.EntityFrameworkCore` a `Microsoft.AspNetCore.Identity.UI`
2. V builderu (nejčastěji v Program.cs) pak zavolat `Services.AddDefaultIdentity<TUser>(options).AddEntityFrameworkStores<TDbContext>()`
   - V základním řešení by `TUser` měl dědit z `IdentityUser` a `TDbContext` z `IdentityDbContext`
3. Dále je potřeba nastavit způsob autentizace, např. `builder.Services.ConfigureApplicationCookie(options)` pro autentizaci pomocí cookies
4. Následně je potřeba autentizaci a autorizaci zapnout:
   - `app.UseAuthentication()` + `app.UseAuthorization()`
5. Přidat novou migraci a aplikovat ji
6. Přidat login stránku využívající služby `UserManager` / `SignInManager`
7. Pro více viz demo

AUTORIZACE V KÓDU

- Pomocí atributu `[Authorize]` na controlleru či akci
- Jde selektivně vypnout na konkrétní akci či controlleru pomocí `[AllowAnonymous]`
- V rámci views jde provést kontrolu pomocí property `User`

- Co když ale potřebujeme složitější logiku než, že je uživatel přihlášený?
  - Autorizace pomocí rolí a nebo politik

DEMO

ZÁKLADNÍ AUTENTIZACE A AUTORIZACE

ROLE

- Potřeba explicitně přidat definice rolí v rámci migrace
- Musí se v builderu zapnout:
  - `builder.Services`
    `.AddDefaultIdentity<TUser>(options)`
    `.AddRoles<TRole>()`
- Přidat uživatele do role lze pomocí služby `UserManager`
- Oveření rolí se opět provádí pomocí atributu `[Authorize(Roles = „FirstRole, SecondRole“)]`
  - Role oddělené čárkou v rámci atributu jsou kontrolovány v módu **OR**
  - Pokud chceme **AND**, stačí přidat dva `[Authorize]` atributy pod sebe

DEMO

AUTORIZACE POMOCÍ ROLÍ

POLITIKY

- Politiky se definují v nastavování autorizace, například:

```
services.AddAuthorization(options =>
    options.AddPolicy("VipGuest", policy =>
        policy.RequireClaim("BadgeGuestType", "VIP")));
```

- Ověření opět přes `[Authorize(Policy = „VipGuest“)]`
- Přidání claimů opět přes službu `UserManager`

- Politiky jsou mnohem komplexnější téma, doporučuji pročíst dokumentaci

AUTORIZACE POMOCÍ POLITIK A CLAIMŮ

DEMO NENÍ (SORRY JAKO)

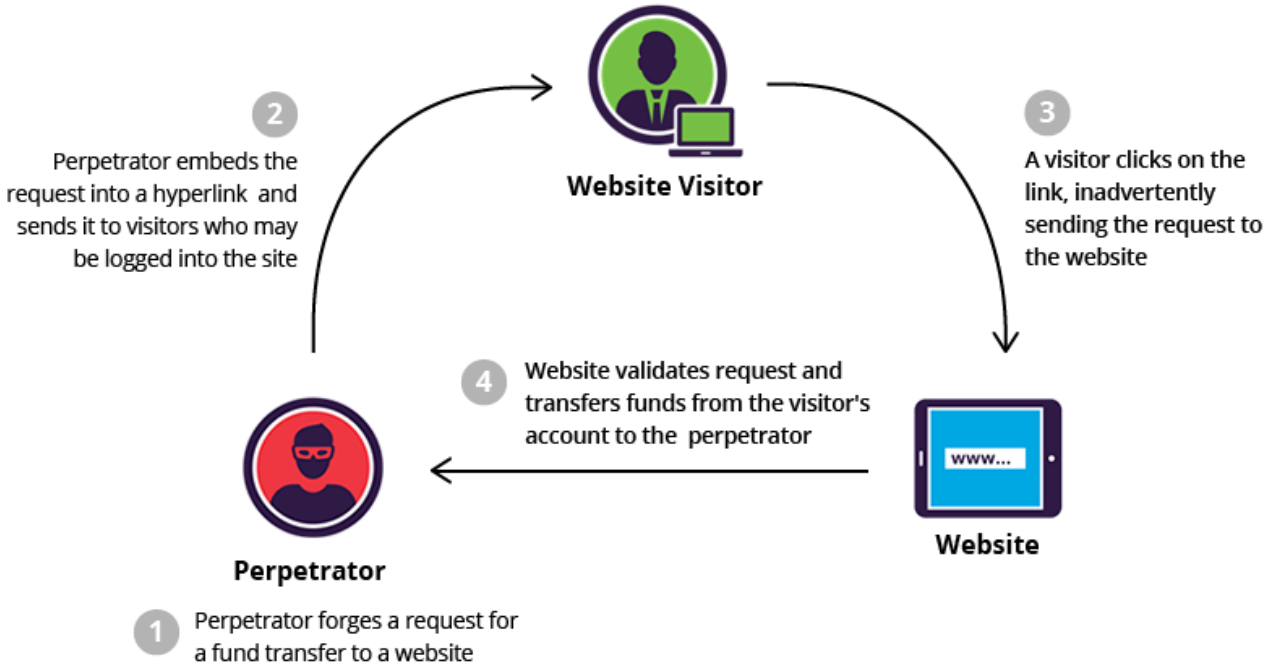
ČASTÉ ÚTOKY A JAK SE JIM BRÁNIT

SQL INJEKCE

- Nesanitizujeme textové vstupy od uživatele při dotazování do databáze
- Příklad z obrázku: `$“INSERT INTO TABLICE(PlateNo, TicketSent, TicketPaid) VALUES(‘{userInput}‘, 0, 0)“`
  - Zkuste si doplnit text SPZ jako input
- Problém má jednoduché řešení – nebastlit SQL ručně, ale použít nějaký existující framework na komunikaci s DB
- Pokud ale opravdu děláme na staříčkém ADO.NET, tak jde vstup jednoduše sanitizovat tak, že jej vložíme do query jako parametr

CSRF

- Průběh útoku viz obrázek
- Příčina:
  - Neověřujeme původ dotazu
- Obrana:
  - Tvořit v Razoru formuláře přes `<form>` **tag helper**
  - Tvořit v Razoru formuláře přes `using (@Html.BeginForm) {}`
  - Přidat do formuáře položku `@Html.AddAntiForgeryToken()`
- Validace na straně controlleru se pak provádí přidáním atributu `[ValidateAntiForgeryToken]`



XSS

- Průběh útoku viz obrázek

- Příčina
  - neescapujeme uživatelské vstupy při renderování HTML

- Obrana:
  - Nepoužívat `HTMLString` / `@Html.Raw()`
  - Nevkládat uživatelský vstup do DOMu pomocí JS konkatenace stringů nebo `document.write()`