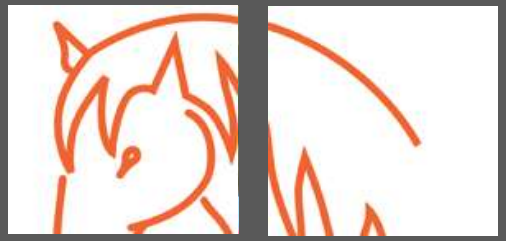
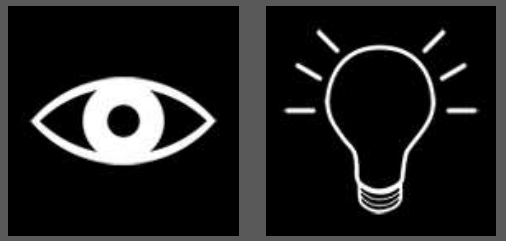


Správa tajemství
v ASP.NET Core aplikacích

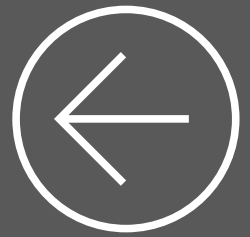


Michal Altair Valášek

Microsoft Most Valuable Professional

Development & Security Consultant, Altairis

michal.valasek@altairis.cz | www.aspnet.cz | www.secpublica.cz



Obsah této přednášky

Budeme se zabývat:

- Krátkodobou ochranou dat
- Typicky data roundtripovaná přes klienta
  - Autentizační tickety, CSRF tokeny
  - Chráněné cookies
  - Vlastní využití, např. reset hesla nebo ověřené e-mailu

Nebudeme se zabývat

- Dlouhodobou ochranou dat
- Obecnými kryptografickými principy

Co znamená „ochrana dat“

- Šifrování
  - Utajení obsahu zprávy
  - Pomocí symetrického algoritmu (AES)
- Elektronický podpis
  - Možnost odhalit změnu zprávy
  - Pomocí HMAC (Hash Message Authentication Code)

- **Authenticated Encryption**
  - **Téměř ve všech aplikacích potřebujete obojí!**
  - **Samotné šifrování (bez podpisu) představuje bezpečnostní riziko!**






Machine Keys

Ochrana dat v ASP.NET 4.x

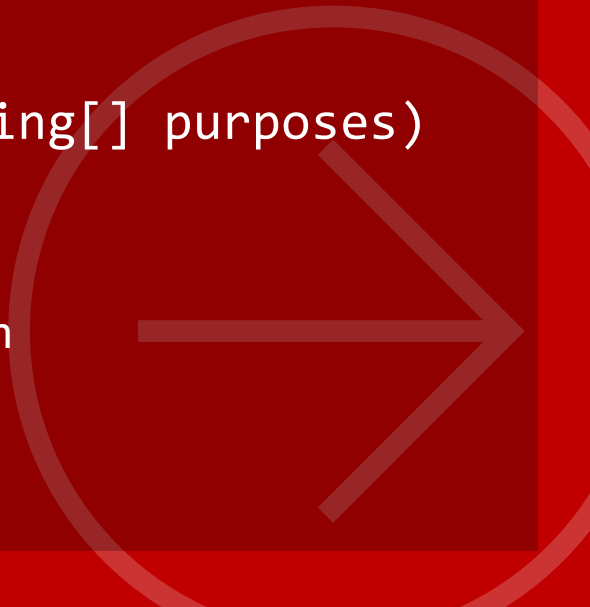
altairis

Co je `MachineKey`

- Dvojice klíčů určená ve web.configu pro
  - **`validationKey`** – elektronický podpis (HMAC)
  - **`decryptionKey`** – šifrování a dešifrování dat
- Lze je generovat automaticky
  - **`AutoGenerate`** – pro všechny aplikace tentýž
  - **`AutoGenerate,IsolateApps`** – různé klíče
- Doporučuji je nicméně specifikovat explicitně
  - Generátor: **`http://chaos.aspnet.cz`**
- Třída **`System.Web.Security.MachineKey`**

Použití třídy `MachineKey`

- `byte[] Protect(byte[] data, params string[] purposes)`
  - Zašifruje a podepíše **`data`**
  - Předává se data „účelů“, která zabraňuje cross-attacku
- `byte[] Unprotect(byte[] data, params string[] purposes)`
  - Dešifruje **`data`** a ověří jejich podpis
  - Musí dostat stejnou sadu „účelů“, jako při **`Protect`**
  - Pokud volání selže, vyhodí **`CryptographicException`**
- „Účely“ slouží jako další „sůl“ při vytváření podpisu








ASP.NET Data Protection API

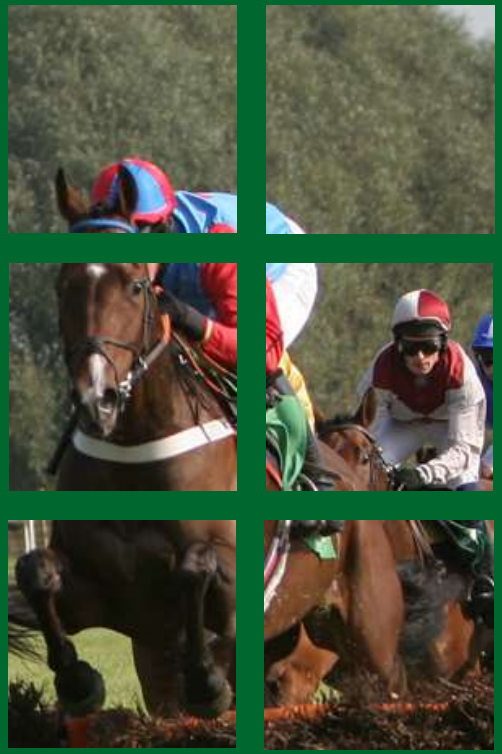
Nezaměňovat s Data Protection API (DPAPI) ve Windows!

Ochrana dat v ASP.NET Core

altairis

Principy ASP.NET Data Protection API

- Z hlediska konzumenta je API velice podobné jako machine keys
  - Metody **`Protect`** a **`Unprotect`**
    - Mají **`byte[]`** i **`string`** variantu
    - Nabízejí šifrování a podpis s pomocí *purposes*
  - Přibyly nové možnosti
    - Automatická rotace klíčů
    - Automatické omezení životnosti payloadu
- Implementace je ale jiná
  - Klíče jsou vesměs uchovávány ve file systému
  - Zůstává zachována možnost zero konfigurace
  - Budete se to toho rýpat zejména na webových farmách



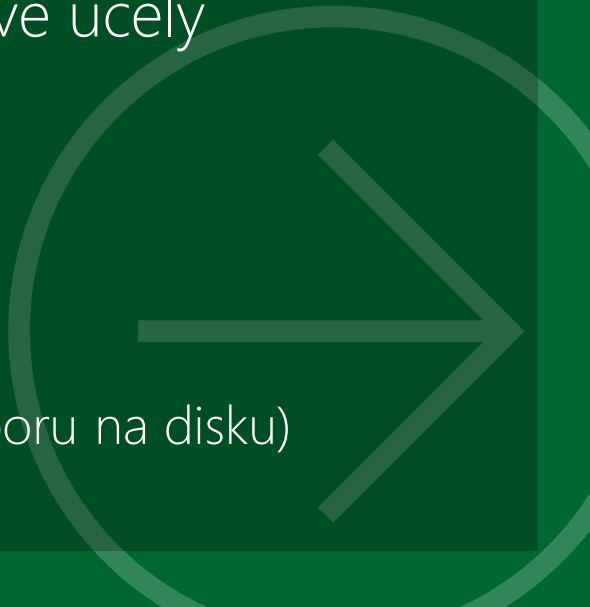
demo

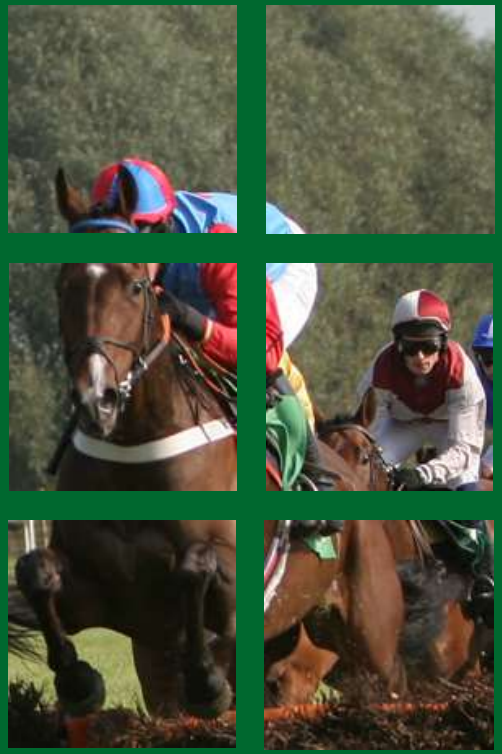
Typické použití ASP.NET
Data Protection API
ve webové aplikaci

altairis

Konfigurace ASP.NET Data Protection API

- V drtivé většině případů jej nebudete sami konfigurovat
  - Využívá ji infrastruktura ASP.NET
  - Vy jenom využijete tuto konfiguraci pro své účely
  - Injectujete si jej pomocí IoC/DI
- Pokud je to nutné, máte možnost
  - Konfigurovat stávající providery
    - Místo ukládání dat (typicky na webové farmě)
    - Dodatečná ochrana dat v klidu (šifrování souboru na disku)
  - Psát vlastní providery





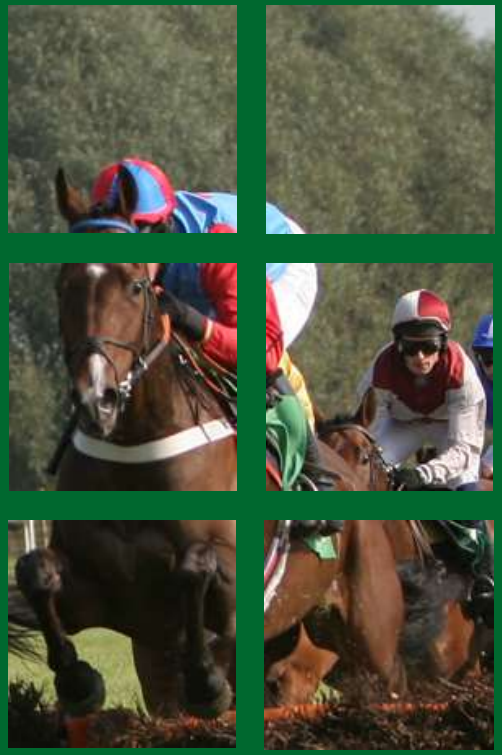
demo

Vlastní konfigurace
a použití mimo DI scénáře

altairis

Omezení životnosti payloadu

- Interně se děje uložením času expirace do obsahu
- Lze použít s jakýmkoliv providerem
  - Jeho timebombed verzi vytvoříte voláním metody `ToTimeLimitedDataProtector`
- Pokud platnost dat vypršela, vyhodí metoda `Unprotect` výjimku `CryptographicException`
- Používejte hojně, leč s rozumem
  - Lepší je vázat platnost na nějakou nativní událost
  - Časové omezení používejte jako záchrannou brzdu



demo

Omezení životnosti payloadu

altairis

Nový systém ve starém .NETu

- Nové Data Protection API lze využívat i ve starém .NETu
  - Podporována verze ≥4.5.1
  - NuGet balíček `Microsoft.AspNet.DataProtection.SystemWeb`
- Typické scénáře
  - Sdílení dat mezi legacy a Core aplikacemi
  - Chcete využít výhod typu automatické rotace klíčů

Podrobnější informace:
https://docs.asp.net/en/latest/security/data-protection/compatibility/replacing-machinekey.html






Jak je to s těmi „účely“

Derivace klíčů

altairis

Derivace klíčů pomocí purpose strings

- **Pro každý účel chceme použít samostatný klíč**
  - Aby nedošlo ke křížové kontaminaci
  - Např. volným uživatelským vstupem
- To by ale byla maintenance nightmare
  - Pročež klíče místo toho derivujeme z jednoho master klíče
    - Potřebujeme deterministický postup…
    - …který ale zachová dostatečnou entropii
  - K tomu se používají purpose strings

Jak zvolit správný purpose string

- Tak, aby ho žádná jiná dobře vychovaná část aplikace (subsystém nebo komponenta) **neměla stejný**
- **Není nutné jej tajit!** Může být ve všech instancích aplikace stejný (natvrdo zadaný v kódu)
- Nesmí v úplnosti pocházet z **uživatelem zadaných údajů** (uživatel by ho mohl podvrhnout)
- Snažíme se o **maximální fragmentaci**
  - Proto máme purposes a sub-purposes
  - Klasičtí kandidáti na sub-purposes: uživatelské jméno, tennant ID, argument potvrzované operace…

dotazy ?

www.**aspnet.cz**

www.**rider.cz**

facebook.com/rider.cz

twitter.com/ridercz

ask.fm/ridercz

youtube.com/altairiscz

michal.valasek@altairis.cz