

# **Bezpečnost webových aplikací**

**4IZ228 – tvorba webových stránek a aplikací**

Jirka Kosek

Poslední modifikace: \$Date: 2011/05/06 08:41:40 \$

Copyright © 2000-2011 Jiří Kosek

# Obsah

<b>Úvod</b> .....	<b>3</b>
Co nás čeká .....	4
<b>Šifrování přenášených dat</b> .....	<b>5</b>
Proč šifrovat .....	6
Jak se dnes šifruje .....	7
Certifikáty a CA .....	8
<b>Autentizace a autorizace uživatelů</b> .....	<b>9</b>
Základní pojmy .....	10
HTTP autentizace .....	11
Vlastní autentizace .....	12
Klientské certifikáty .....	13
OpenID .....	14
Ukládání hesel .....	15
Autorizace .....	16
<b>Nejčastější bezpečnostní slabiny aplikací</b> .....	<b>17</b>
Nekontrolování vstupu od uživatele .....	18
Použití neinicializovaných proměnných .....	20
Ochrana session .....	21
SQL injection .....	22
<b>Pár slov závěrem</b> .....	<b>23</b>
Pár slov závěrem .....	24
<b>Zdroje informací</b> .....	<b>25</b>
Povinná četba .....	26
Další zajímavé zdroje .....	27

# Úvod

Co nás čeká .....	4
-------------------	---

# Co nás čeká

- šifrování přenášených dat
- autentizace a autorizace uživatelů
- přehled nejčastějších slabých míst v aplikacích

# Šifrování přenášených dat

Proč šifrovat .....	6
Jak se dnes šifruje .....	7
Certifikáty a CA .....	8

# Proč šifrovat

- některá data jsou skutečně citlivá
  - on-line bankovníctví
  - komunikace mezi obchodními partnery
  - vzdálený přístup do podnikového IS
- kromě zašifrování přenosu většina technologií nabízí ochranu před záměnou komunikujících stran

# Jak se dnes šifruje

- hybridní systémy – kombinace asymetrických a symetrických šifer
- pro zabezpečení přenosu se používá SSL (Secure Sockets Layer) nebo TLS (Transport Layer Security) – protokoly umožňující zašifrovat cokoliv na bázi protokolu TCP
- prohlížeč si o zabezpečené připojení řekne pomocí speciálního URL ve tvaru

https://...

- server se s klientem dohodne na kvalitě šifrování (možnost snížení kvality šifrování počítačem mezi klientem a serverem)

# Certifikáty a CA

- server posílá klientovi certifikát
- certifikát – spojuje dohromady počítač s reálně existující osobou (fyzickou či právníkou)
- certifikát slouží pro ověření totožnosti serveru
- certifikát může mít i klient, ale na webu se to zatím moc nepoužívá
- certifikát vydává certifikační autorita (CA) – ta by měla ověřit skutečnou identitu žadatele o certifikát
- prohlížeč automaticky věří certifikátům od CA, které zná (umí ověřit podpis na certifikátu)
- ostatní certifikáty je potřeba ručně doinstalovat nebo doinstalovat CA, která je vystavila

# Autentizace a autorizace uživatelů

Základní pojmy .....	10
HTTP autentizace .....	11
Vlastní autentizace .....	12
Klientské certifikáty .....	13
OpenID .....	14
Ukládání hesel .....	15
Autorizace .....	16

# Základní pojmy

autentizace

ověření totožnosti

autorizace

ověření práv pro vykonání určité činnosti

# HTTP autentizace

- standardní součást protokolu HTTP
- nelze změnit podobu přihlašovacího okna
- obtížně se řeší odhlášení a automatické odhlášení po určité době
- bývá implementována na úrovni webového serveru
- hesla jsou přenášena v nekódované podobě
  - bezpečnější metoda Digest se začíná rozšiřovat až v poslední době

# Vlastní autentizace

- využívá HTML formuláře a session proměnné
- mnohem větší flexibilita oproti HTTP – vlastní přihlašovací stránka, hesla uložená na libovolném místě
- v session proměnné se uchovávají informace o přihlášeném uživateli a o době jeho posledního přístupu
- odhlášení – stačí zrušit session proměnnou
- automatické odhlášení – při každém požadavku se porovnává aktuální čas s časem posledního přístupu (ten je uložen v session proměnné)
- pokud klient podporuje JavaScript, lze použít challenge-response mechanismus (heslo není přenášeno v odkrytém tvaru)

# Klientské certifikáty

- využívá mechanismus SSL/TLS, ale certifikátem se neproказuje jen server, ale i uživatel
- uživatelsky méně přívětivé – uživatel musí mít na počítači k dispozici svůj certifikát
- využívá se v aplikacích, které potřebují vzbudit zdání větší bezpečnosti, např. internetové bankovníctví

# OpenID

- decentralizovaný mechanismus pro jednotné přihlašování k webovým aplikacím (single sign on)
- uživatel používá jednotný OpenID identifikátor v mnoha aplikacích
- autentizaci neprovádějí jednotlivé aplikace, ale poskytovatel identity

# Ukládání hesel

- aplikace by v žádném případě neměla ukládat hesla v odkryté podobě
- ukládání otisku (hashe) hesla nestačí, kvůli předgenerovaným slovníkům otisků
- doporučené je ukládat otisk hesla a „soli“

# Autorizace

- obvykle se řeší až na aplikační úrovni
- přístupová práva bývají nejčastěji uložena v nějaké databázi

# Nejčastější bezpečnostní slabiny aplikací

Nekontrolování vstupu od uživatele .....	18
Použití neinicializovaných proměnných .....	20
Ochrana session .....	21
SQL injection .....	22

# Nekontrolování vstupu od uživatele

- veškerá data získaná od uživatele by měla být před použitím ověřena
- musíme počítat s tím, že uživatel omylem udělá chybu nebo se někdo záměrně snaží nabourat do aplikace
- data je potřeba vždy validovat na straně serveru, protože kód běžící na klientovi může uživatel měnit (např. AJAXové aplikace)
- data pocházející od uživatele (může je měnit)
  - obsah formulářových polí
  - URL adresa požadavku
  - cookies
  - HTTP hlavičky

## Příklad 1. Získání libovolného souboru ze serveru

Předpokládejme, že máme skript, který generuje webové stránky. Obsah stránky získá ze zvoleného souboru a k němu doplní standardní hlavičku a patičku. Jednotlivé stránky se tak volají pomocí adresy

`http://example.org/index.php?page=uvodni.inc.`

... standardní hlavička v HTML ...

```
<?php
```

```
    include $_GET["page"];
```

```
?>
```

... standardní patička ...

Co se stane, když zlý uživatel zadá URL ve tvaru

`http://example.org/index.php?page=/etc/passwd?`

## Příklad 2. Správné řešení s kontrolou dovolených vstupů

... standardní hlavička v HTML ...

```
<?php
```

```
    if (in_array($_GET["page"], array("uvod.inc", "cenik.inc", ►  
"kontakt.inc"))
```

```
    {  
        include $_GET["page"];
```

```
    }
```

```
    else
```

```
    {
```

```
        echo "Požadovaná stránka neexistuje. Pokračujte na
```

```
        <a href='index.php?page=uvod.inc'>úvodní stránce</a>.";
```

```
    }
```

# Nekontrolování vstupu od uživatele (Pokračování)

?>

... standardní patička ...

# Použití neinicializovaných proměnných

- některé jazyky (např. starší verze PHP) automaticky načítají data z požadavku do proměnných
- data zvnějšku mohou změnit obsah neinicializované proměnné

## Příklad 3. Špatný kód

```
<?php
    if (over_prihlaseni()) $prihlasen = true;

    if ($prihlasen)
    {
        // proved' veřejně nepřístupné operace
    }
?>
```

Lze snadno napadnou přidáním `?prihlasen=1` na konec URL adresy požadavku.

## Příklad 4. Správný kód

```
<?php
    $prihlasen = false;
    if (over_prihlaseni()) $prihlasen = true;

    if ($prihlasen)
    {
        // proved' veřejně nepřístupné operace
    }
?>
```

- pro snížení rizika obdobných chyb nové verze PHP nenačítají data z vnějšku (metody GET a POST, cookies) do proměnných, ale jsou dostupná ve speciálních polích `$_GET`, `$_POST`, `$_COOKIE` apod.

# Ochrana session

- jediná 100% spolehlivá ochrana je SSL a vypnuté posílání HTTP hlavičky `Referer`
- útok spočívá ve špatné kontrole vstupu a v cross-site skriptování

## Příklad 5. Získání session-id přenášeného v URL

1. na serveru, kde je uživatel přihlášen, je diskusní fórum
2. útočník do diskusního fóra přidá příspěvek s odkazem vedoucím na jeho server (odkaz musí být zajímavý, aby zaujal)
3. skript na útočnickově serveru z HTTP hlavičky `Referer` získá kompletní URL předchozí stránky včetně session-id
4. pomocí získaného session-id se útočník může přihlásit na server pod jménem uživatele a číst jeho data, změnit heslo, ...
5. snížení rizika:
  - všechny odkazy ve vložených příspěvcích přesměrovávat přes pomocnou stránku, která již v URL nemá session-id
  - dodatečná kontrola session-id (kontrola shody IP adresy, session-id se mění pro každou stránku)

## Příklad 6. Ochrana session-id přenášeného v cookie

1. na serveru, kde je uživatel přihlášen je diskusní fórum
2. útočník do diskusního fóra přidá příspěvek s kusem JS kódu, který čte cookie

```
<script>document.write('')</script>
```

3. útočníkův skript získá obsah cookie a může ji zneužít
4. ochrana:
  - jako v předchozím případě
  - důsledná kontrola vstupních dat, nedovolit zadání HTML a JS kódu do příspěvku

# SQL injection

- skripty často konstruuji SQL dotaz dynamicky na základě vstupů
- vstupy se musí pečlivě kontrolovat, aby chybný vstup neumožnil spuštění libovolného SQL příkazu

## Příklad 7. Chybný skript umožňující SQL injection

Formulář obsahuje vstupní pole `jmeno` pro zadání hledaného jména

```
<?php
...
$jmeno = $_GET["jmeno"];
$spojeni = ODBC_Connect("test", "user", "password");
$vysledek = ODBC_Exec($spojeni,
    "SELECT * FROM Zamestnanci
    WHERE Jmeno LIKE '$jmeno%'
    ORDER BY Jmeno");
...
?>
```

## Příklad 8. Správné řešení s kontrolou vstupu

Před předáním dat dotazu se testuje, zda řetězec obsahuje jen povolené znaky

```
<?php
...
$jmeno = $_GET["jmeno"];
if (!eregI("^-[áčďěěíóöřšťúůýž]+$" , $jmeno))
{
    echo "Hledaný text může obsahovat jen písmena.";
    exit;
}
$spojeni = ODBC_Connect("test", "user", "password");
$vysledek = ODBC_Exec($spojeni,
    "SELECT * FROM Zamestnanci
    WHERE Jmeno LIKE '$jmeno%'
    ORDER BY Jmeno");
...
?>
```

# Pár slov závěrem

Pár slov závěrem ..... 24

# Pár slov závěrem

*Nepodceňovat!!!!!!!!!!!!*

# Zdroje informací

Povinná četba .....	26
Další zajímavé zdroje .....	27

# Povinná četba

- OWASP Guide to Building Secure Web Applications<sup>1</sup>
- OpenID<sup>2</sup> – distribuovaný systém pro autentizaci uživatelů

<sup>1</sup> <http://prdownloads.sourceforge.net/owasp/OWASPGuide2.0.1.pdf?download>

<sup>2</sup> <http://openid.net/>

## Další zajímavé zdroje

- WWW Authentication<sup>3</sup> – rozsáhlý článek o možnostech autentizace na webu
- Open Web Application Security Project<sup>4</sup> – popis nejčastějších druhů bezpečnostních chyb v aplikacích
- Pěkná ukázka využití SQL injection pro nabourání do špatně zabezpečeného serveru<sup>5</sup>

<sup>3</sup> <http://www.seifried.org/security/www-auth/>

<sup>4</sup> <http://www.owasp.org>

<sup>5</sup> <http://web.archive.org/web/20051026171811/http://www.underground.cz/969>