Testing for DOM-Based Cross Site Scripting (DOM Tabanlı Çapraz Site Komut Dosyası Oluşturma Testi)

Summary (Özet)

DOM tabanlı çapraz site komut dosyası, bir sayfadaki aktif tarayıcı tarafındaki içeriğin, genellikle JavaScript'in bir kaynak üzerinden kullanılmasının ve bir lavaboda kullanılmasının sonucu olan XSS hatalarının fiili adıdır. Bu belge yalnızca XSS'ye yol açan JavaScript hatalarını tartışır.

DOM veya Belge Nesnesi Modeli, belgeleri bir tarayıcıda temsil etmek için kullanılan yapısal
formattır. DOM, JavaScript gibi dinamik komut dosyalarının belgenin bir form alanı veya oturum çerezi gibi bileşenlerine atıfta bulunmasını sağlar. DOM ayrıca tarayıcı tarafından güvenlik için kullanılır - örneğin farklı alan adı üzerindeki komut dosyalarını diğer alanlar için oturum çerezleri elde etmekten sınırlamak için. DOM tabanlı bir XSS güvenlik açığı, JavaScript işlevi gibi aktif içerik, özel olarak hazırlanmış bir istekle değiştirildiğinde ortaya çıkabilir, böylece bir saldırgan tarafından kontrol edilebilen bir DOM elemanı.

Tüm XSS hataları, saldırganın sunucudan iade edilen içeriği kontrol etmesini gerektirmez, bunun yerine aynı sonuçları elde etmek için zayıf JavaScript kodlama uygulamalarını kötüye kullanabilir. Sonuçlar tipik bir XSS kusuru ile aynıdır, sadece teslimat araçları farklıdır.

Diğer çapraz site komut dosyasılama güvenlik açıkları ile karşılaştırıldığında (sağlıksız bir parametrenin sunucu tarafından geçirildiği ve saklandığı, daha sonra kullanıcının tarayıcısı tarafından iade edildiği ve kullanıcının tarayıcısı bağlamında gerçekleştirildiği, DOM tabanlı bir

XSS güvenlik açığı, belge Nesnesi Modeli'nin (DOMU) unsurlarını kullanarak kodun akışını kontrol eder.

Doğaları nedeniyle, DOM tabanlı XSS güvenlik açıkları, sunucu gerçekte neyin yürütüldüğünü belirleyemedik. Bu, genel XSS filtreleme ve algılama tekniklerinin çoğunu bu tür saldırılara engel teşkil edebilir.

Bu varsayımsal örnek, aşağıdaki istemci tarafı kodunu kullanır:

```
<script>
document.write("Site is at: " + document.location.href + ".");
</script>
```

Bir saldırgan eklenebilir `#<script>alert('xss')</script>` Etkilenen sayfa URL'sine, uygulandığında, uyarı kutusunu görüntüler. Bu durumda, eklenen kod sunucuya her şey olarak gönderilmezdi. `#` Karakter, tarayıcı tarafından sorgunun bir parçası olarak değil, bir parça olarak
ele alınır. Bu örnekte, kod hemen yürütülür ve sayfa tarafından “karmaşık” uyarısı görüntülenir. Daha yaygın çapraz site komut dosyası türlerinin aksine (kodun sunucuya gönderildiği ve daha sonra tarayıcıya geri gönderildiği yansıtılan ve saklanan, bu doğrudan sunucu bağlantısı olmadan kullanıcının tarayıcısında gerçekleştirilir.

DOM tabanlı XSS kusurlarının sonuçları, çerez alımı, daha fazla kötü niyetli komut dosyası enjeksiyonu vb. Dahil olmak üzere XSS'nin daha iyi bilinen formlarında görülenler kadar
geniştir ve bu nedenle aynı ciddiyetle tedavi edilmelidir.

Test Objectives (Test Hedefleri)

- DOM lavabolarını tanımlayın.
- Her lavabo tipine uygun yükler oluşturun.

How To Test (Nasıl Test Edilir)

JavaScript uygulamaları diğer uygulama türlerinden önemli ölçüde farklıdır, çünkü genellikle sunucu tarafından dinamik olarak oluşturulurlar. Hangi kodun uygulandığını anlamak için, test edilen web
sitesinin, JavaScript'in yürütüldüğü tüm örnekleri ve kullanıcı girişinin kabul

edildiği yerleri belirlemek için süründürülmelidir. Birçok web sitesi, genellikle yüz binlerce kod satırına uzanan ve şirket içinde geliştirilmeyen büyük işlev kitaplıklarına güvenir. Bu durumlarda, yukarıdan aşağıya test genellikle tek geçerli seçenek haline gelir, çünkü birçok alt seviye işlevi asla kullanılmaz ve hangi lavaboların sıklıkla mevcut olandan daha fazla zaman kullanacağını belirlemek için bunları analiz etmek. Aynı şey, girişlerin veya eksikliğinin başlangıçta tanımlanmaması durumunda yukarıdan aşağıya test için de söylenebilir.

Kullanıcı girişi iki ana biçimde gelir:

- sunucu tarafından sayfaya doğrudan XSS'ye izin vermeyecek şekilde yazılmış giriş ve
- Müşteri tarafı JavaScript nesnelerinden elde edilen giriş.

İşte sunucunun JavaScript'e nasıl veri ekleyebileceğinin iki örneği:

```
var data = "<escaped data from the server>";
var result = someFunction("<escaped data from the server>");
```

İşte istemci tarafı JavaScript nesnelerinden iki giriş örneği:

```
var data = window.location;
var result = someFunction(window.referrer);
```

JavaScript kodunda nasıl alındıkları konusunda çok az fark olsa da, giriş sunucu aracılığıyla alındığında, sunucunun istediği verilere herhangi bir permütasyon uygulayabileceğini belirtmek önemlidir. Öte yandan, JavaScript nesneleri tarafından gerçekleştirilen permütasyonlar oldukça iyi anlaşılır ve belgelenmiştir. Eğer `someFunction` Yukarıdaki örnekte bir lavabo vardı, daha sonra eski durumdaki istismar, sunucu tarafından yapılan filtrelemeye bağlı olurdu, oysa ikinci durumda tarayıcı tarafından yapılan kodlamaya bağlı olacaktır.
`window.referrer` Nesne. Stefano Di Paulo, belgeyi ve konum özelliklerini kullanarak bir URL'nin çeşitli öğelerini istendiğinde tarayıcıların hangi tarayıcıların geri döndüğüne dair mükemmel bir makale yazdı.

Ek olarak, JavaScript genellikle dışında yürütülür. `<script>` Bloklar, geçmişte XSS filtre baypaslarına yol açan birçok vektör tarafından kanıtlandığı gibi. Uygulamayı sürünürken, ifade özniteliklerine sahip etkinlik işleyicileri ve CSS blokları gibi

yerlerde komut dosyalarının kullanımını not etmek önemlidir. Ayrıca, hangi kodun yürütüldüğünü belirlemek için herhangi bir yerinde CSS veya komut dosyası nesnelerinin değerlendirilmesi gerekeceğini unutmayın.

Otomatik test, DOM tabanlı XSS'yi tanımlamada ve doğrulamada, genellikle belirli bir yük göndererek ve sunucu yanıtında gözlemleme girişimleri yaparak XSS'yi tanımlamada çok sınırlı bir başarıya sahiptir. Bu, mesaj parametresinin kullanıcıya geri yansıtıldığı aşağıda belirtilen basit örnek için iyi çalışabilir:

```
<script>
var pos=document.URL.indexOf("message=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</script>
```

Ancak, aşağıdaki hileli durumda tespit edilmeyebilir:

```
<script>
var navAgt = navigator.userAgent;

if (navAgt.indexOf("MSIE")!=-1) {
      document.write("You are using IE as a browser and visiting site: " + doc
      ument.location.href + ".");
}
else
{
   document.write("You are using an unknown browser.");
}
</script>
```

Bu nedenle, otomatik test, test aracı istemci tarafı kodunun ek analizini yerine getiremediği sürece, DOM tabanlı XSS'ye duyarlı olabilecek alanları tespit etmeyecektir.

Bu nedenle manuel test yapılmalı ve bir saldırgan için yararlı olabilecek parametrelerin atıfta bulunulduğu koddaki alanları inceleyerek yapılabilir. Bu tür alanların örnekleri, kodun sayfaya

dinamik olarak yazıldığı yerleri ve DOM'un değiştirildiği veya hatta senaryoların doğrudan yürütüldüğü yerleri içerir.

Remediation (Düzeltme)

DOM tabanlı XSS'yi önlemeye yönelik önlemler için, DOM tabanlı XSS Önleme Hile Levhasına bakın.

Referance (Referanslar)

- DomXSSWiki
- DOM XSS article by Amit Klein