Cross-Site Scripting (XSS) Rehberi

Web'in En Yaygın Zafiyetini Anlamak, Tespit Etmek ve Önlemek

XSS Nedir? Güven Sınırının İhlali

Cross-Site Scripting (XSS), saldırganların güvenilir web sayfalarına kötü amaçlı istemci tarafı (client-side) betikler enjekte etmesine olanak tanıyan bir zafiyettir.

Hedef doğrudan web sunucusu değil, o sunucuya güvenen kullanıcının tarayıcısıdır.

Siber Güvenlik Açısından Önemi ve Etkileri

Oturum Çalma

Kullanıcıların session (oturum) verilerinin ele geçirilmesi.

Veri Hırsızlığı

Hassas kişisel bilgilerin sızdırılması.

Yönlendirme

Kullanıcının haberi olmadan zararlı sitelere yönlendirilmesi.

Site Tahrifatı

Tarayıcıdaki görsel sunumun manipüle edilmesi (Vandalism).

XSS doğrudan kullanıcıya saldırır, ancak tüm yasal sorumluluk ve itibar kaybı siteye aittir.

XSS Açıklıkları Nelerdir? (3 Temel Kategori)

1 Yansıyan (Reflected) XSS

Kalıcı değildir (Non-persistent).

İstekteki zararlı payload (örneğin URL'de), sunucu tarafından anında sayfaya yansıtılır.

2 Kalıcı (Stored) XSS

Sunucuda depolanır (Persistent).

Payload veritabanına veya dosya sistemine kaydedilir, sayfayı ziyaret eden herkese çalışır.

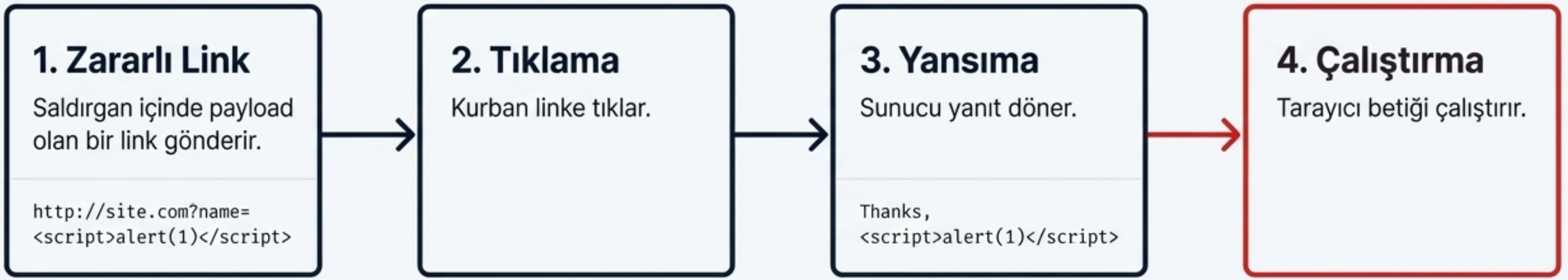
3 DOM Tabanlı XSS

İstemci tarafında gerçekleşir.

Zafiyet tamamen tarayıcıdaki DOM (Document Object Model) manipülasyonundan kaynaklanır.

1. Yansıyan (Reflected) XSS

Parametreler veya kullanıcı girdileri doğrulanmadan anında ekrana basılırsa meydana gelir. Saldırgan, kurbanı özel hazırlanmış bir linke tıklamaya ikna etmelidir (Oltalama/Phishing).



Sık Görülen Yerler: Arama kutuları (Search fields) veya hata mesajı sayfaları.

2. Kalıcı (Stored) XSS

Zararlı kod, uygulamanın veritabanına bir kez enjekte edilir. Sayfayı açan her masum kullanıcı bu koda maruz kalır.

1. Enjeksiyon

Saldırgan bir foruma veya yoruma zararlı script yazar ve kaydeder.

2. Depolama

Sunucu bunu veritabanına normal bir veri gibi kalıcı olarak kaydeder.

3. Toplu Etkilenme

Sayfayı görüntüleyen tüm ziyaretçilerin tarayıcısında bu script otomatik olarak çalışır.

Sık Görülen Yerler: Blog yorumları, sosyal ağ mesaj panoları, kullanıcı profil sayfaları.

3. DOM Tabanlı XSS

Kullanıcı verisi, güvenlik süzgecinden geçirilmeden doğrudan DOM nesnelerine aktarıldığında oluşur.

```
let payload = window.location.hash;

document.getElementById('default').innerHTML =
payload;
```

- ✓ Zararlı veri URL'deki # (hash) işaretinden sonra gelir.
- ✓ Sunucu # sonrasını görmez, sadece tarayıcı işler.
- ✓ Doğrudan sayfa kaynağında (client-side) manipülasyon yapılır.

Zafiyet Nasıl Tespit Edilir? (Manuel Analiz)

1

Girdi Yansımalarını İzlemek

Formlara (örneğin ‘Geeks’ kelimesi) zararsız veri girip, HTML kaynağında nereye ve nasıl yansıdığını kontrol etmek.

2

HTML Olaylarını (Events) Test Etmek

Tıklama, yükleme gibi özellikleri manipüle etmeyi denemek.

`onload` `onerror` `onclick`

3

Özel Karakter Taraması

Sunucunun karakterleri filtreleyip filtrelemediğini (Blacklist/Whitelist) deneme yanılma ile bulmak. `<` `>` `"` `'` `&`

Güvenlik Filtrelerini Aşmak (Bypass Teknikleri)

Geliştiriciler genellikle sadece <script> etiketini engeller. Ancak HTML'in esnek yapısı, saldırganlara sayısız alternatif sunar.

Engellenen (Blocked)

```
HELLO
<script>alert('XSS')</script>
```

(Tarayıcı etiketi siler)

Çalışan (Bypassed) - Alternatif Vektörler

```
<img src='no_such_file.jpg'
onerror=alert('123');>
```

```
<body onload=alert('attack')>
```

HTML o kadar esnektir ki, çift kodlama (double encoding) veya eksik karakterler bile betiğin çalışmasını durdurmayabilir.

Otomatik Tespit Araçları

Dinamik Analizörler (Dynamic Scanners)

Uygulama çalışırken otomatik zafiyet taraması yapar ve payload'lar göndererek uygulamanın tepkisini ölçer.

Fuzzing Araçları (Fuzz Testers)

Beklenmedik, geçersiz veya rastgele veri yığınları (özel karakterler içeren stringler) göndererek sistemin hatalarını tespit eder.

Statik Kod Analizi (Static Analyzers)

Kaynak kodunu çalıştırmadan tarayarak, girdilerin filtrelenmeden ekrana basldığı değişkenleri (echo, print) tespit eder.

Temel Savunma: Girdi Doğrulama (Input Validation)

Karaliste (Blacklisting)

Neden Başarısız Olur?

- Sadece bilinen zararlı girdileri (örneğin <script>) engellemeye çalışır.
- Zararlı string kombinasyonları sonsuzdur.

```
"jav a sc ript"
"%25%35%63"
```

Beyaz Liste (Whitelisting)

En İyi Uygulama

- Sadece beklenen, tanımlı ve güvenli veri formatlarına (uzunluk, tip, içerik) izin verir.
- Beklenmeyen tüm karakterler varsayılan olarak reddedilir.

Karakter Seti Etkisizleştirme (Output Escaping)

İstemciye gönderilen tüm veriler, tarayıcının bunları kod değil, düz metin olarak okuyacağı şekilde dönüştürülmelidir.

Raw Characters		Escaped Entities
<	→	<
>	→	>
&	→	&
"	→	"
'	→	'

Sadece etiketleri değil, HTML özniteliklerini (attributes) de koruyun. Tüm öznitelikler tırnak içine alınmalı ve veriler standartlarda kodlanmalıdır.

Modern Tarayıcı Savunması: CSP

Content Security Policy (CSP), yeni nesil tarayıcıların XSS etkilerini en aza indirmek için kullandığı güçlü bir teknolojidir.

- **Kaynağı Sınırla:** Betiklerin (script) sadece güvenilen, belirli alan adlarından (domain) yüklenmesine izin verir.
- **Inline JavaScript'i Engelle:** Sayfa içine doğrudan yazılan betiklerin (<script>...</script> veya <body onload=...>) çalışmasını varsayılan olarak durdurur.
- **Derinlemesine Savunma (Defense in Depth):** Girdi doğrulama ve kodlamanın atlatıldığı senaryolarda son güvenlik bariyerini oluşturur.

Özet: Güvenli Geliştirme Kültürü

1

Tasarım (Design)

- HTML ve JavaScript için güvenli kodlama standartları belirleyin.
- Girdilerin yansıtıldığı tüm noktaları haritalandırın.

2

Geliştirme (Implementation)

- Veri girdisi ve çıktısı için standart, ortak modüller kullanın.
- İşletim sistemi, veritabanı ve dilin yerleşik güvenlik fonksiyonlarını tercih edin.

3

Test (Testing)

- XSS için özel birim testleri (Unit tests) yazın.
- Geliştirme sürecine dinamik tarayıcıları entegre edin.