XSS(Cross Site Scripting)

Masaüstü uygulaması olsun, web uygulaması olsun bir programda zafiyet bulunabilmesi için programda bir girdi(input) noktası olmalıdır.Bu girdi noktaları kimi zaman bir parametre, kimi zaman bir form, kimi zaman ise bir login sayfası olabilir.

Tabii ki bu örnekler çoğaltılabilirler. Bu noktada genel olarak bilmemiz gereken şey girdi noktalarının filtrelenme işlemi dikkatli bir şekilde yapılmalıdır. İstemci(client) tarafından gelen veriye ne olursa olsun güvenilmemelidir.Çoğu web zafiyeti istemci tarafından veri(data) beklerken istemcinin kod veya komut göndermesi ve bunların sunucu tarafında çalıştırılmasından kaynaklanmaktadır.

Cross Site Scripting ‘ te de durum farklı değildir.Bu açığın oluşmasının ana sebebi verinin encode edilmeyişi ve verinin doğru filtrelenmeyişidir.İstemci tarafını(client-side) kapsayan bir zafiyettir.Bu zafiyetin bulunuduğu web sitelerine zararlı javascript ve HTML kodları(payload) enjekte edilebilir.Sonucunda bu zararlı payload web tarayıcısında çalışır. Çoğu web zafiyetinin aksine -örneğin:sqli- hedef web uygulaması değildir. Hedef web uygulamasını kullanan kullanıcılardır.

XSS zafiyetini kullanarak gerçekleştirilebilecek saldırılar şu şekildedir:

- Kullanıcı oturumlarını çalmak.
- Tarayıcı zafiyetlerini sömürmek.
- Normal kullanıcıların zararlı işlemler gerçekleştirmesini sağlamak.

XSS zafiyeti kendi içerisinde 3 ayrı kategoride incelenirler. Bunlar:

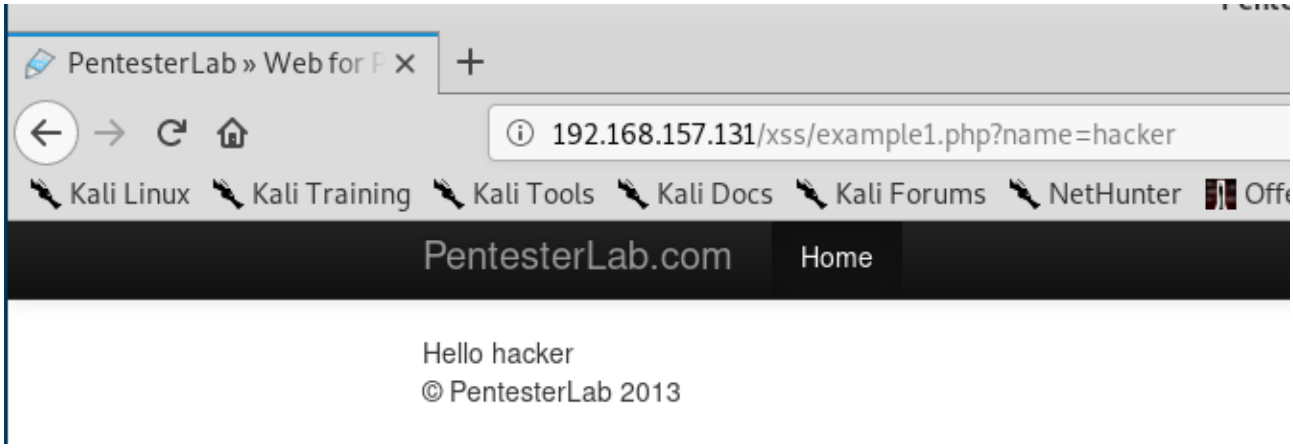
- Reflected XSS : Bu tür XSS zafiyetlerinde kodlar kalıcı olarak çalıştırılmaz. Hedefin zararlı bağlantıya gitmesi gerekmektedir.Zafiyetin sömürüleceği javascript kodlarının bulunduğu bağlantı hedefe gönderilir ve hedefin bağlantıyı açması beklenir.
- Stored XSS : Bu tür XSS zafiyetlerinde sisteme zararlı javascript kodlarının eklenilmesi mümkün olur. Örneğin bir sitede kullanıcıların içerik hakkındaki yorumlarına izin veriliyor olsun. Yorum kısmına girdi olarak zafiyetli olduğunu bildiğimiz XSS payload’umuzu gönderirsek, bu payload sunucu tarafında kaydolacağı için web uygulamasına giren her kullanıcıda zararlı javascript kodu çalıştırılacaktır.
- DOM based XSS : DOM kısaca HTML etiketlerinin(tag) genel adıdır. Bu tür bir XSS zafiyetinde javascript ile HTML taglarının özellikleri değerleri vb. değiştirilebilmektedir. Reflected XSS ‘ te olduğu gibi saldırıdan sadece zafiyetli bağlantıyı açan kullanıcılar etkilenir.

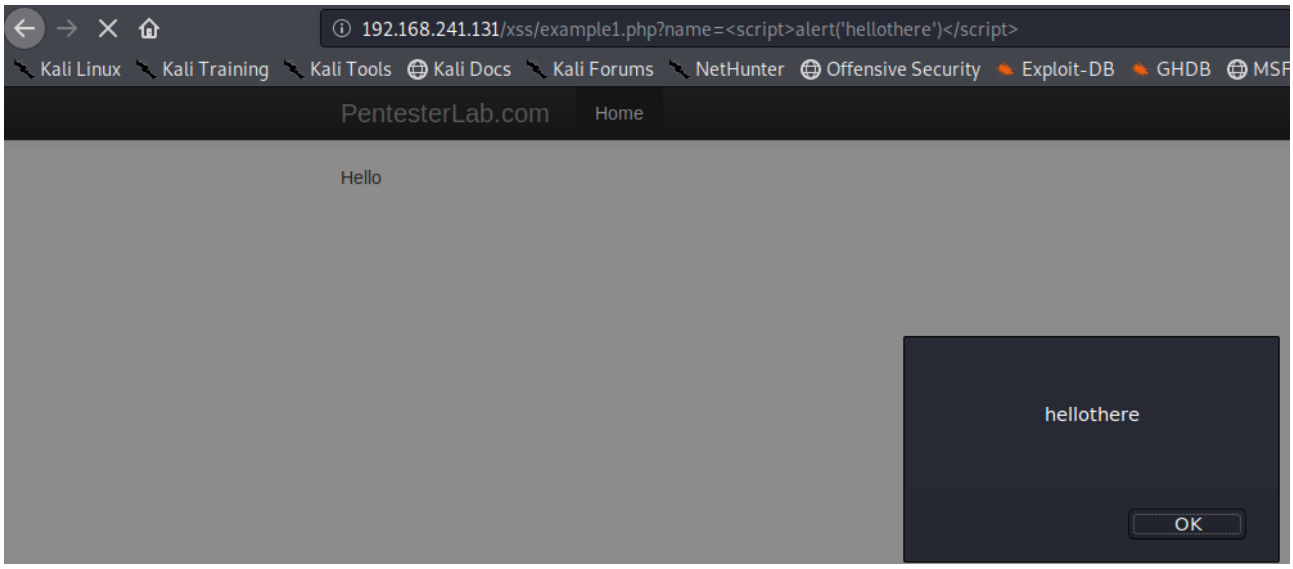
Uygulama

Şimdi zafiyetin nasıl sömürüleceğini görmek için bir lab ortamı oluşturalım. Ben bunun için Pentesterlab'ın sunmuş olduğu web for pentester's sanal makinasını kullanacağım.

Iso linki : https://pentesterlab.com/exercises/web_for_pentester/attachments

Örnek1-



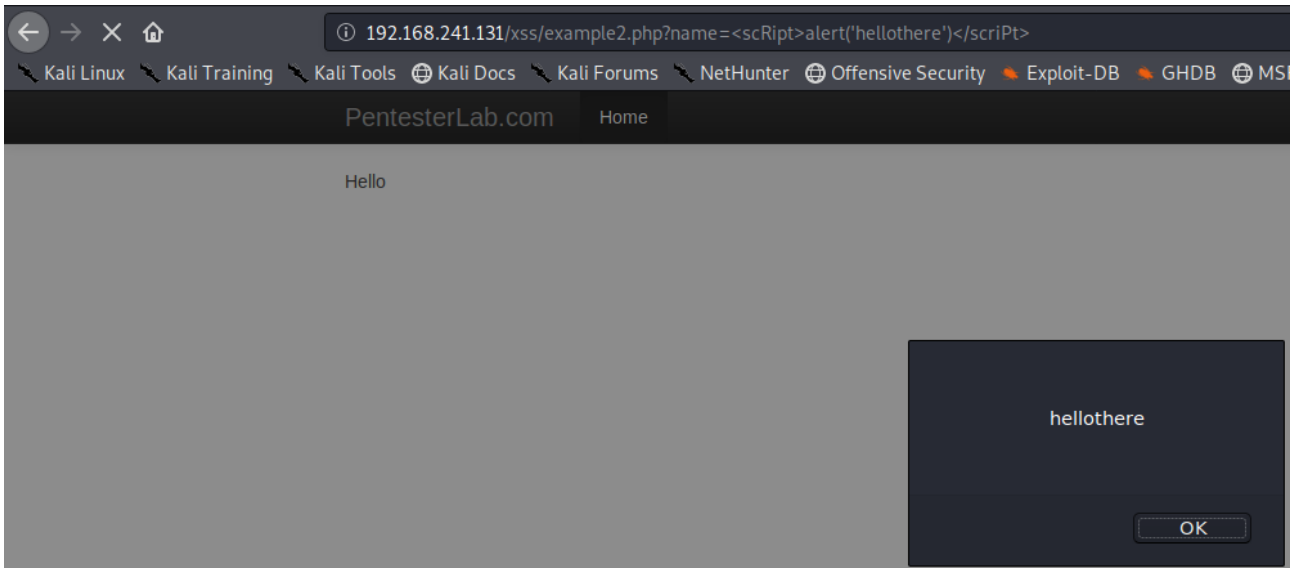


Birinci örnekte bizi yukarıdaki gibi bir ekran karşılıyor.Name adındaki parametrede bir girdi noktamız olduğunu görüyoruz.Uygulamamız boyunca amacımız javascript'in tetiklenip tetiklenemediğini görmemiz için ekrana bir uyarı(alert) kutusu bastırmak olacak.Javascript kodunun çalıştırılması için birden fazla yöntem olmakla birlikte biz kodumuzu çalıştırmak için <script> tag'ini kullanacağız.

Birinci örnekte herhangi bir filtreleme ile karşılaşmadığımız için kodumuzu aşağıdaki payload ile çalıştırabildik.Yani veri girmemiz gereken kısıma kod girdik.

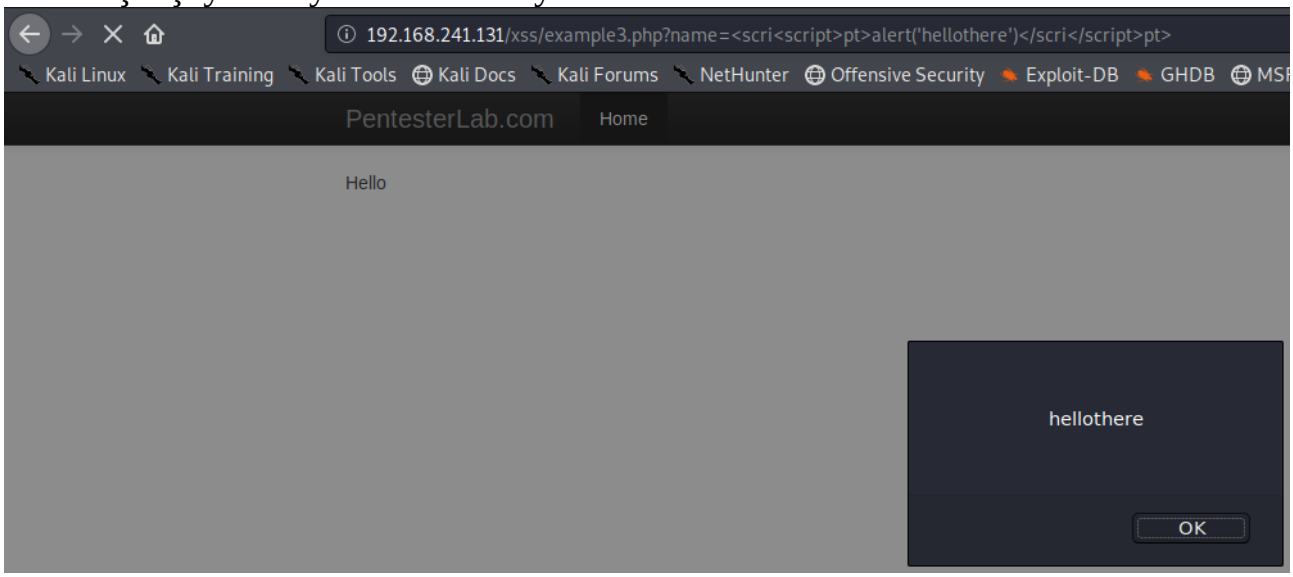
Payload : <script>alert(‘hello-there’)</script>

Örnek2-
Bu örnekte yukarıdaki payloadımızı denediğimiz zaman kodumuzu çalıştıramadığımızı göreceğiz. Yani bir filtreleme söz konusu. Bu filtrelemeyi aşmak için farklı yöntemler kullanacağız. HTML ‘ de taglerin küçük-büyük harf duyarlı (case sensitive) olmadığı aklımıza geliyor ve şu şekilde bir payload kullanıyoruz.



Payload : <scRipt>alert(‘hellothere’)</scriPt>

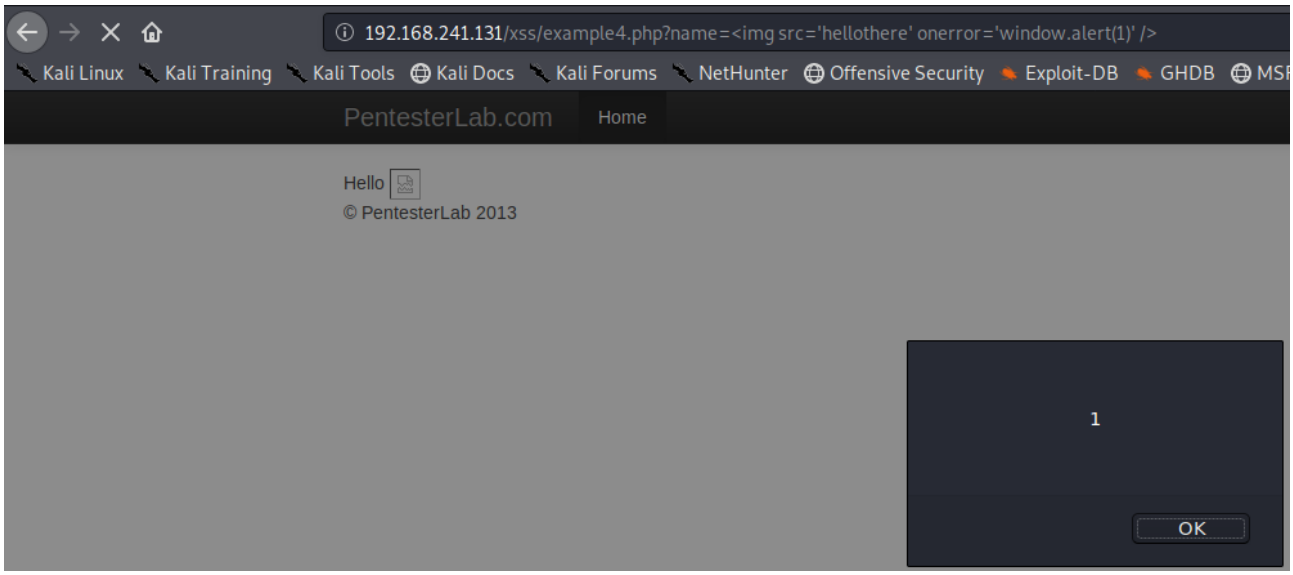
Örnek3-Bu örnektede önceki örneklerde olduğu gibi name parametresini ekrana basıyor.Denemelerimiz sonucu <script> tag’inin engellendiğini görüyoruz. Bypass etmek için şöyle bir yöntem kullanıyoruz.



Payload : <scri<script>pt>alert('hellothere')</scri</script>pt>

Örnek4-

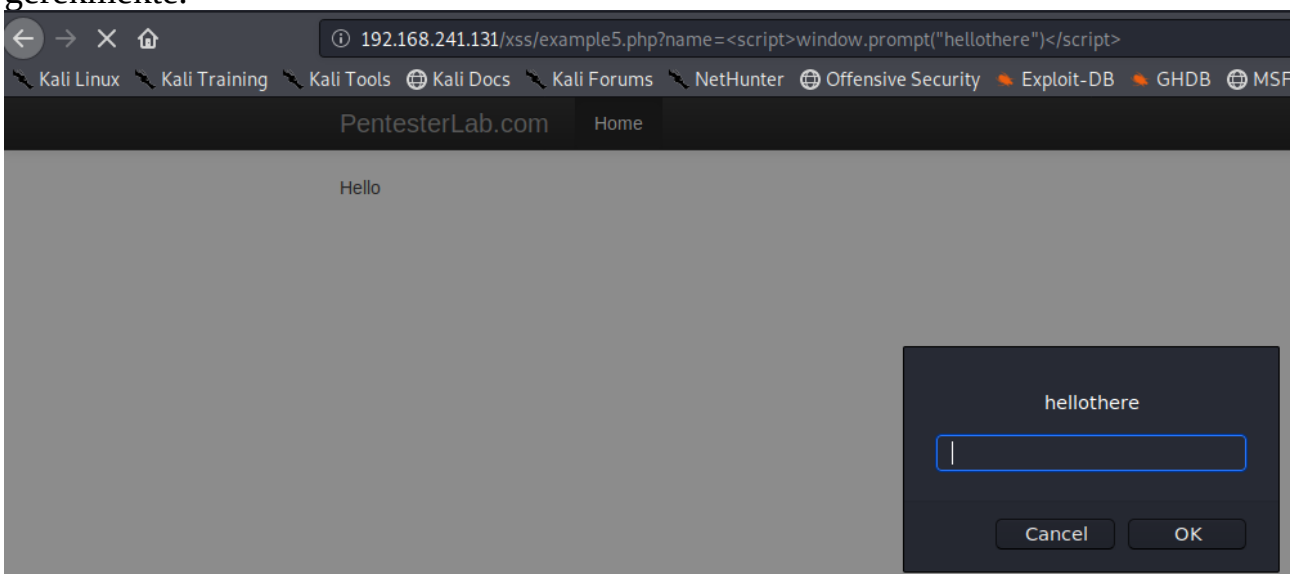
Önceki payloadlarımızı deneyince görüyoruzki bu sefer geliştiriciler <scirpt> tag'ini tamamen engellemişler. Daha öncede bahsettiğimiz gibi javascript kodu çalıştırmak için birden fazla yolumuz mevcut. Bu yöntemde bir resim dosyasının bulunmaması halinde javascript kodumuzu çalıştıracak bir script yazıyoruz. Resim dosyasının bulunmadığından emin olmak için ise rastgele bir source veriyoruz.



Payload :

Örnek5-

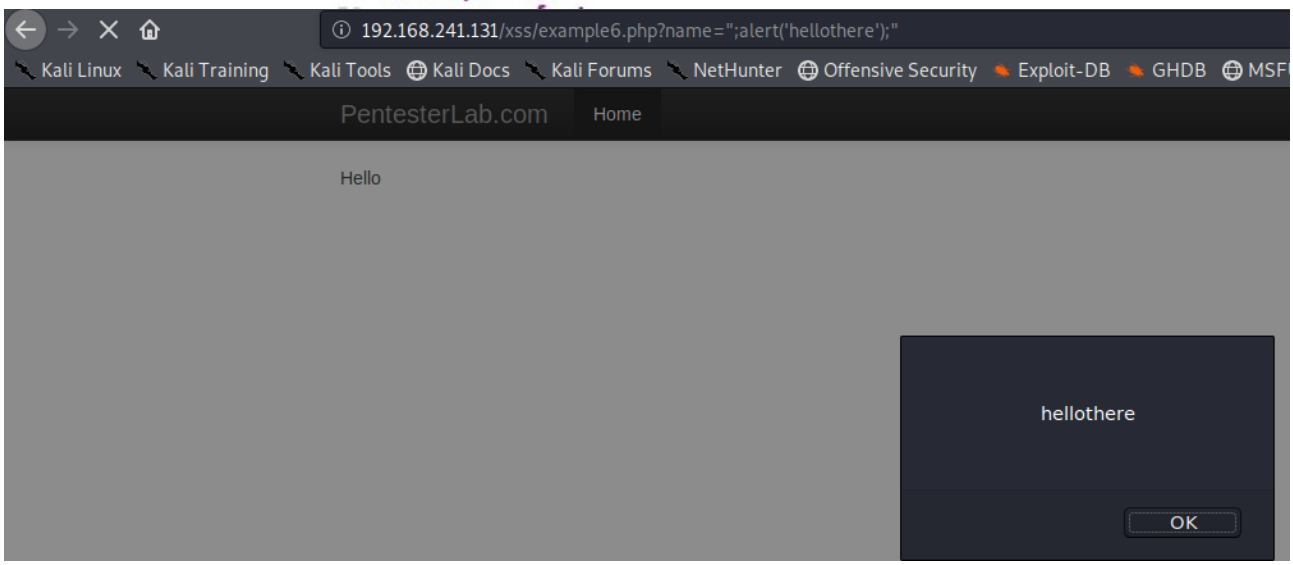
Bu örneği incelediğimizde ise alert fonksiyonu ve <script> tag'inin birlikte filtrelendiğini görüyoruz. Bu yüzden bunları kullanmadan kodumuzu çalıştırmamız gerekmekte.



Payload: <script>window.prompt("hellothere")</script>

Örnek6-
Bu örnekte önceki payloadlarımızı deneyip hata mesajı aldıktan sonra, kaynak kodunu inceliyoruz. Buradanda görüldüğü gibi parametremiz halihazırda script etiketinin içerisine alınmış halde.

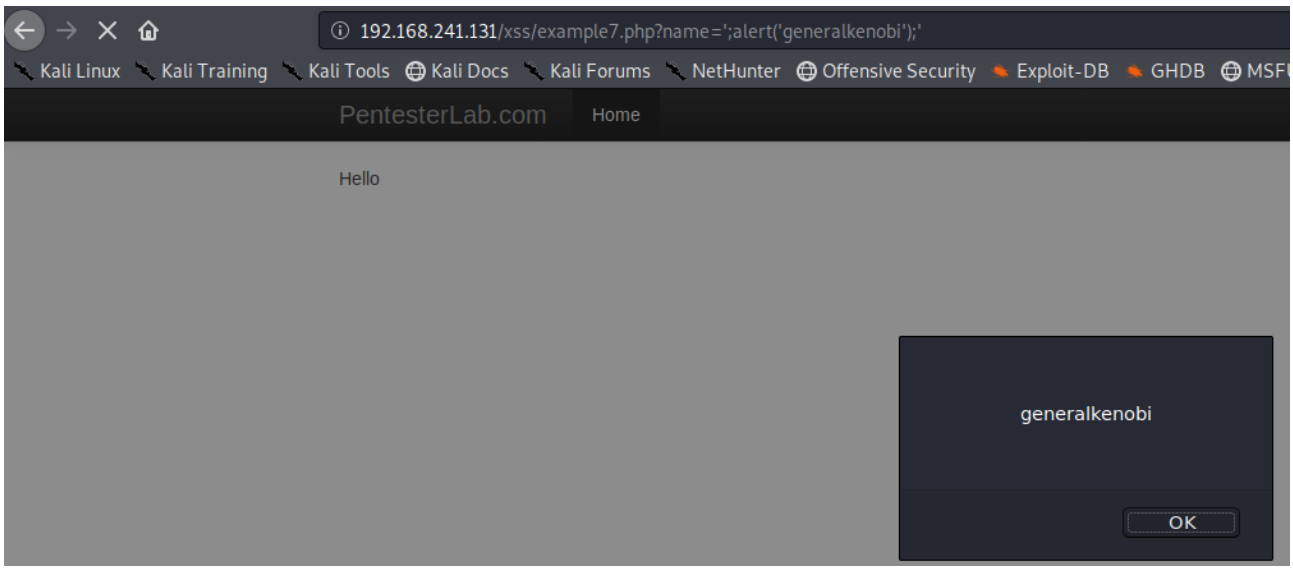
```
47 Hello
48 <script>
49     var $a= "hacker";
50 </script>
```



Payload : “;alert(‘hellothere’);”

Örnek7-
Kaynak kodunu incelediğimizde az önceki soruda ki gibi değişkenimizin <script> tag’i içerisinde olduğunu görüyoruz. Tek fark değişken tanımında tek tırnak yerine çift tırnak kullanılmış olması.

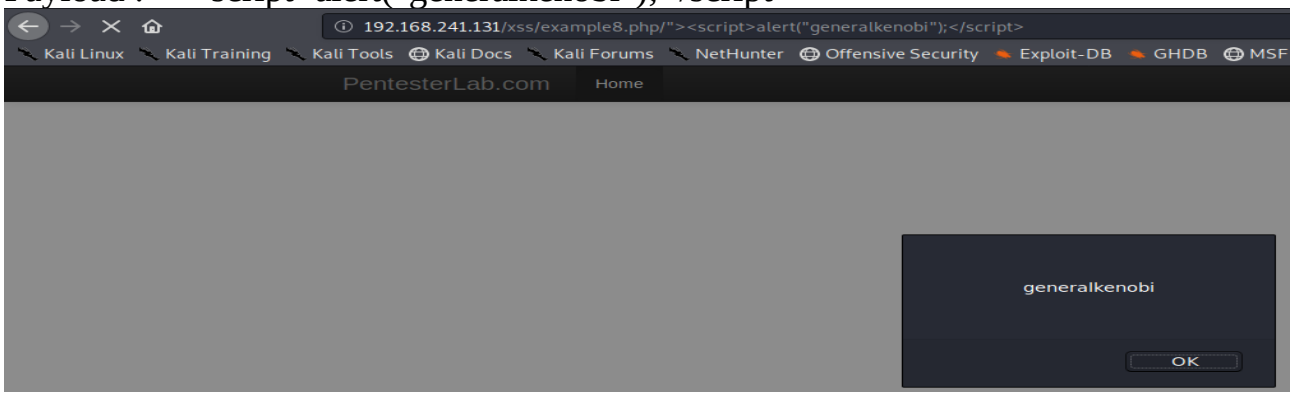
```
48 Hello
49 <script>
50     var $a= 'hacker';
51 </script>
52
```



Payload : ';alert('generalkenobi');'

Örnek8-
Bu örnekte karşımıza çıkan sayfa farklılaşıyor. Parametreyi URL üzerinden almak yerine parametre bir form aracılığıyla alınıyor ve bu form POST metodu ile yollanıyor.

Dikkatimizi çeken bir başka nokta ise girdi alındıktan sonra formun tekrar kullanıcıya döndürülmesi.
Payload : "><script>alert("generalkenobi");</script>



Görüldüğü gibi amacımız genel olarak alışılmışın dışında bir yaklaşım ile zafiyetleri tespit etmek oldu. Tabii ki bu zafiyet özelinde Javascript bilgiside oldukça önemli olacaktır.