SQLi nedir

SQL Mantığı:

```
+----------+-------------+--------------+
| kisi_adi | kisi_soyadi | dogum_tarihi |
+----------+-------------+--------------+
| muhammed | uzuner      |         1111 |
| selman   | çetin       |         2222 |
| ömer     | aydın       |         3333 |
| miraç    | tok         |         4444 |
| fatih    | alak        |         5555 |
| hamdi    | bulut       |         1808 |
+----------+-------------+--------------+
6 rows in set (0.000 sec)
```

SQL Injection'u anlamak için öncelikle SQL yapısını anlamamız lazım. SQL, Structured Query Language (Yapılandırılmış Sorgu Dili) olarak açılır. Ancak tarihin biraz daha öncesine baktığımızda, açılımı Structured English Query Language (Yapılandırılmış İngilizce Sorgu Dili) olarak geçen bir yazılım dili olduğunu görüyoruz. Bunun sebebi, SQL dilinin büyük bir çoğunluğunun matematik operatörlerinden ziyade İngilizce sorgu yazma üzerine tasarlanmasıyla alakalıdır. Sorgu kelimesini açıklamak gerekirse, SQL dilinde bildiğimiz satır ve sütundan oluşan bir tabloyu ifade eder. İstediğimiz satır ve sütundaki herhangi bir veriyi çekmek için karmaşık ve aşırı kodlar yazmak yerine, İngilizce cümleler kullanırız. Örneğin, 'Kullanıcılar tablosundaki kişi adlarını ve soyadlarını bana getirir misin?' cümlesinin SQL dilindeki yazılışı şu şekildedir: 'SELECT kisi_adi, kisi_soyadi FROM Kullanıcılar;'. 'SELECT' ifadesi istediğimiz sütunları getirmek için kullanılırken, 'FROM' ifadesi bu sütunların hangi tablodan geleceğini belirtir. Bu anlattığım en basit SQL sorgu mantığıdır. Daha fazlası için lütfen SQL dilini araştırın.

SQL Injection Nedir?

SQL Injection, hedef sitede çalışan SQL sorgularının içine kendi sorgularımızı karıştırarak manipülasyon yapmayı amaçlar. Bu şekilde, sitelerde, şirketlerde veya kurumlarda bulunan tüm veriler ifşa edilebilir. Örneğin, Reon Sağlık Hizmetleri şirketinin veritabanı sızdırıldı ve hastaların adı, soyadı, TC kimlik numarası gibi hassas bilgiler açığa çıktı. Veritabanlarının sızdırılması, hem kurumlar için hem de verileri bulunan kullanıcılar için büyük bir tehdit oluşturur. Bu nedenle, SQL Injection'ın iyi anlaşılması ve önlemlerin alınması önemlidir. SQL Injection'a geçmeden önce, sitelerin hangi tür sorguları çalıştırdığını anlamamız gerekmektedir çünkü kaynak kodlarına erişimimiz olmadığı için siteden gelen tepkilere göre arka planda çalışan sorguyu tahmin etmemiz gerekmektedir.

Örnek Sorgu 1 :

```
SELECT username, password FROM accounts WHERE username = 'User1' AND password = '123456' LIMIT 1
```

Bu sorgu, "accounts" adlı tablodan kullanıcı adı "user1" ve şifresi "123456" olan satırı getirir. Sonundaki "LIMIT 1" ifadesi, yalnızca tek bir satırda tek bir sorgu çalıştırmak için bir sınırlama getirir (SQL Injection'ı önlemek için). Bu tür sorgular genellikle hesap giriş işlemleri için kullanılır. Eğer kullanıcı adı ve şifre tabloda bulunursa "true" döndürülerek hesaba giriş yapılır.

Örnek Sorgu 2 :

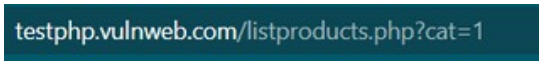
SELECT * FROM products WHERE category = 'Gifts'

Bu tarz sorgular genellikle alışveriş siteleri veya blog sitelerinde bu- lunur. ‘Gifts’ yazan kısmı kullanıcıdan input olarak alır ve kul- lanıcının aradığı kelime ile ilgili ilanlar listelenir(örnek: Alışveriş sitelerinde arama kutuları).

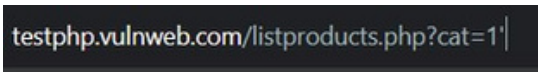
Sql İnjection Tespiti:

Error Based Sqli:

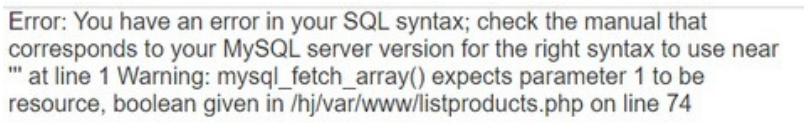
Bu sql injection modelinde sitede çalışan sql sorgusunu bozarak sitenin bize hata mesajı göstermesi amaçlanmaktadır. Bu sayede hacker sql sorgusuna müdahele edebildiğini ve yazdığı payloadın hiç bir filtrelemeye takılmadan çalıştığını anlar. Örneklendirmek gerekirse:



PhpVuln sitesinde kategori listelemek için bir parametre alıyor. Bunu ”?cat=1” ibaresinden anlayabiliriz. cat kelimesi de category kelimesinin kısaltırması olduğunu tahmin ediyorum.



Sitedeki url nin sonuna bir tane tek tırnak atarak sitede çalışan sorguyu bozmayı deneyeceğim.



ve site bize sql hatası verdi. Bu sonuca bakarsak bu sitede sql injection var diyebiliriz.
sonuç olarak arkadaki sorguyu aklımda şekillendirmeye çalışırsam şöyle olurdu

SELECT Productname, Description, Authors, ... FROM Listing WHERE category = '1'

yukarıda yazdığım kolon adları ve tablo ismi tamamen tahmindir.
Tablo adını ve kolonları öğrenmek için başka metodlar kullanmalıyız.
Error Based Sql İnjection için bazı wildcardlar:

'
"
') ")
`)
'))
"))
`))
--

Blind Sqli:

Kör injection, yazdığımız sorguların bize herhangi bir dönüşünü göremediğimiz sorgulardır.

genellikle site bize doğru sorgu yazarsak True yanlış kod yazarsak False döndürür.

Hacker ise ” % ” operantını kullanarak kelimeleri tahmin etmeye çalışır. Örneklendirmek gerekirse bir tabloda Kullanıcı adlarını çekmeye çalışalım. Bize true döndüren sorgunun sonuna “WHERE username LIKE ‘a%’ ” deriz ve a ile başlayan kullanıcı adı varsa true döndürür eğer yoksa a yerine 29 harfi teker teker dener ve doğru harfi bulmaya çalışır. Bir kullanıcı adının tamamı bu şekilde bulunur. Biraz yorucu gibi gözüksede otomatik toollarla birlikte bir hayli kolaylaşıyor.

Time Based Sqli:

Zaman tabanlı sql injection, Girdiğimiz sorgu sonucunda ne bir hata çıktısı ne de bir true, false döndürdüğü durumları kapsar. Hacker yazdığı kodun çalışıp çalışmadığını anlamak için kodunun sonuna bekleme komutu koyar ve ona dönen cevabın kaç saniyede döndüğüne göre sql açığını tespit eder. Örneklendirmek gerekirse

select sleep(10) from users where....

ifadesini var olan sorgunun sonuna yazar ve normalde dönecek olan cevabın 10 saniye daha geç gelmesini bekler. Eğer başarılı olursa sql injection vardır.

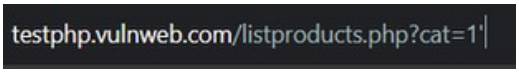
Örnek SQL Injection:

Hedef Sitemiz : vulnweb

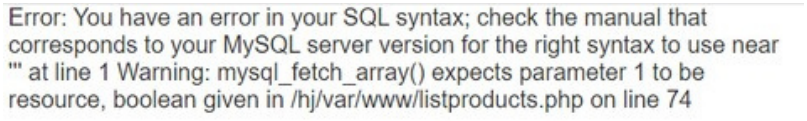
Aşamaları sıra ile sizde deneyerek sql injection yapmayı deneyebilirsiniz.

1) Test alanını belirlemek: “?cat=1” parametresi bana şüpheli geldi ve burada sql injection testleri yapmaya karar verdim.

2) sql injection zaafiyetinin tespiti için tek tırnak atarak sorguyu bozmaya çalışıyorum ve başarılı oluyorum.



Sonuç



3) Şimdi kolon sayısını bulmamız lazım. Bunun Sebebi Sql dili C dilinden üretilmiştir ve c dili matrix hesabı yaparken sutün sayıları eşit olmak zorundadır. Bu yüzden sırayla sutün sayısı arttırıyorum.

not: ile 1,2,... şeklinde arttırmak yerine NULL,NULL,... şeklinde de arttırablirsiniz.

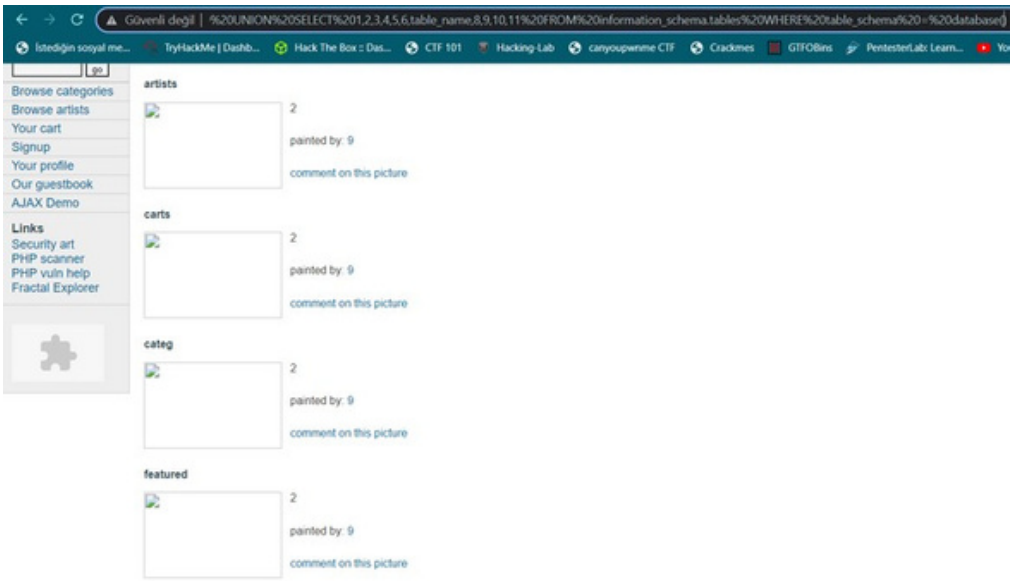
not2: sitedeki diğer ilanları görmemek ve rahat çalışabilmek için sorgunun başındaki “?cat=1” ifadesindeki 1 değerini 99999 yaptım(yani olmayan bir id talep ettim). Böylelikle sitedeki diğer ilanlar karşıma çıkmayacak.
http://testphp.vulnweb.com/listproducts.php?cat=999999 UNION SELECT 1,2,3,4,5,6,7,8,9,10,11

Bu sayede olmayan bir ilan geldi ve 7, 2, 9, 11 sayıları geldi bu sayılar bu sitedeki içeriklerin hangi kolondan geldiğini beliritiyor. Url'de belirtilen sayılar yerine artık sql fonksiyonları yazabiliriz.

4) Kolon sayısını bulduktan sonra information_schema içindeki diğer tabloların bilgilerine erişeceğiz.

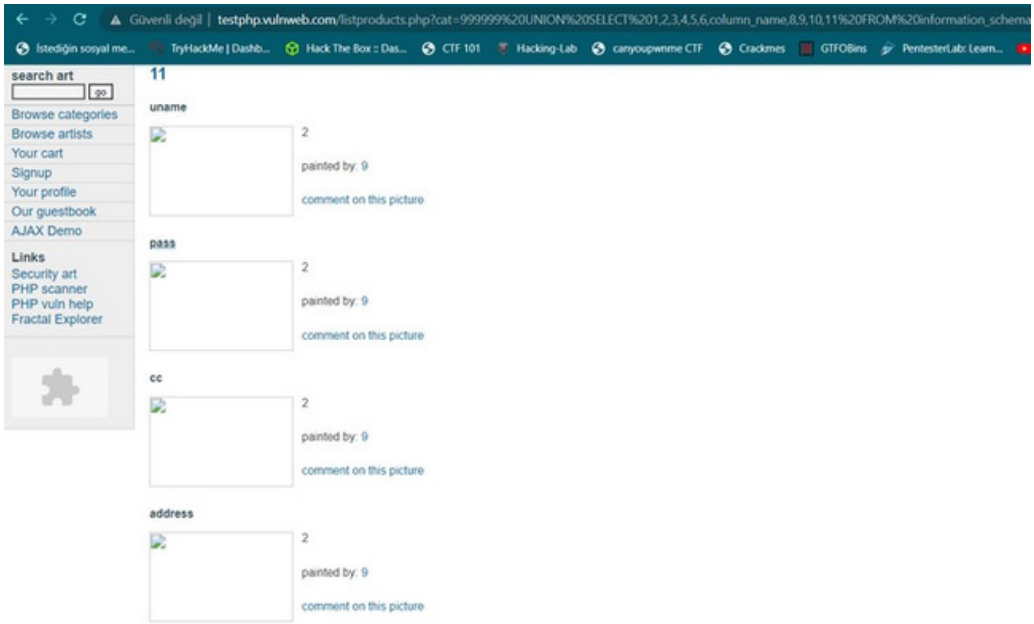
information_schema'ı basitçe anlatmak gerekirse sistemdeki databaselerin bilgilerini tutan genel bir databasedir. bizde bu database sömüreceğiz.

http://testphp.vulnweb.com/listproducts.php?cat=999999 UNION SELECT 1,2,3,4,5,6,7,8,9,10,11



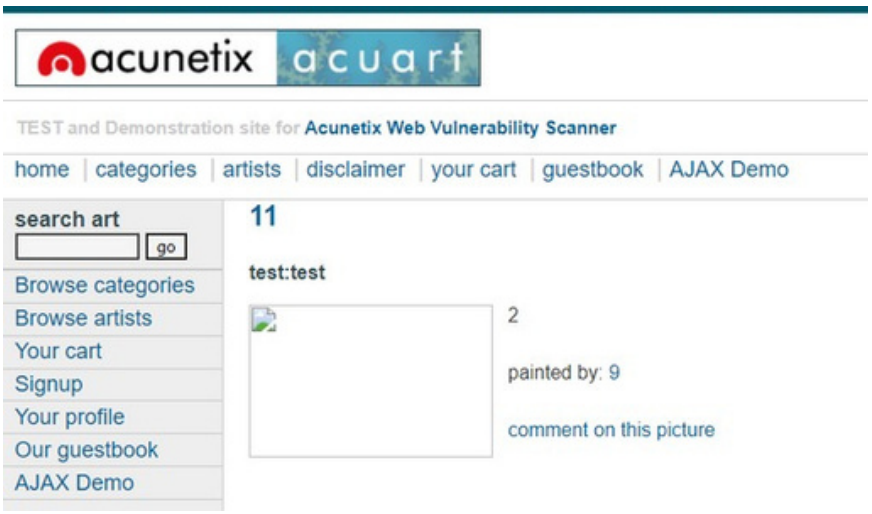
Böylelikle sistemde bulunan tablo adlarını çekmiş olduk.

5) Tablo ismini çektikten sonra son önemli noktamız kolon isimlerini çekmek-
tir. Bunun içinde tablo isminin users olan kolonları listeliyorum
http://testphp.vulnweb.com/listproducts.php?cat=999999 UNION SELECT
1,2,3,4,5,6,column_name,8,9,10,11 FROM information_schema.columns WHERE column_name = 'users'



6) Artık tablo isimini ve kolonları bildiğimize göre istediğimiz veriyi rahatça çekebiliriz.

http://testphp.vulnweb.com/listproducts.php?cat=999999 UNION SELECT 1,2,3,4,5,6,concat(uname, ":", pass),8,9,10,11 FROM users



Sonuç olarak kullanıcı_adı = test, şifresi = test.

Sql injection'u Önlemek :

White List Yaklaşımı:
White List yaklaşımı, kullanıcı girişlerini kontrol etmek için güvenilen ve izin verilen değerlerin bir listesini oluşturmayı içerir. Yani, web uygulamanıza gelen tüm kullanıcı girişleri, bu belirlenmiş güvenli değerlerle eşleştirilerek filtrelenir. Sadece bu listedeki değerlerin kabul edildiği bir doğrulama süreci uygulanır.

Blacklist Yaklaşımı:
Blacklist yaklaşımı, yasaklanmış ve kabul edilemez değerlerin bir listesini oluşturmayı içerir. Bu değerler, SQL injection saldırılarında yaygın olarak kullanılan karakterler, kelimeler veya kod parçacıklarını içerebilir. Kullanıcı girişleri bu yasaklı değerlerle karşılaştırılır ve eşleşme tespit edilirse reddedilir.

Filtrelizasyon Yaklaşımı:
Filtrelizasyon yaklaşımı, kullanıcıdan gelen input değerlerini SQL sorgusuna doğrudan yerleştirmek yerine, SQL sorgusunun parametreleri olarak kullanılır. Bu parametreler, veritabanı sorgusu çalıştırıldığında, veri tabanı yönetim sistemi tarafından güvenli bir şekilde işlenir. Bu yöntem, SQL enjeksiyon saldırılarına karşı daha güvenli bir ortam sağlar, çünkü kullanıcı girişi doğrudan SQL kodunun bir parçası olarak yorumlanmaz.