

SQL INJECTION 102

M. Fatih YILMAZ

mfth78@hotmail.com

İçerik

- Union Based SQL Injection
- Web For Pentester Çözümleri

UNION BASED SQL INJECTION

UNION ifadesiyle başka bir tablodaki bilgiler tek sorguda çekilebilir. SQL injection zafiyeti bulunan web sayfalarında bu ifade ile herhangi aynı database üzerindeki farklı bir tabloya ait olan bilgileri kendi isteğimiz üzerine çekebilmemizi sağlar. UNION bize daha fazla SELECT sorgusu çalıştırma fırsatı verdiği için dönücek response bilgisinde sorgumuzla birlikte istediğimiz verileri görebiliriz. Örnek verecek olursak;

```
SELECT a,b FROM table1 UNION SELECT c,d FROM table2
```

Yukarıdaki sorgu sayesinde tablo 1 deki a ve b değeriyle birlikte tablo 2 deki c ve d değerlerini 2 sütundan oluşan tek bir sonuç gibi bize döndürecektir. UNION keywordünün sql sorgularında çalışabilmesi için bazı ön şartlar karşılanmalıdır. Yaptığımız her select sorgusu aynı sütun sayısına eşit olmalıdır. Mesela ilk tablomuzda 3 ikincisinde 6 varsa 6 sınıda UNION ile çekemeyiz. Bunu anlamak için bir takım yöntemler yer almaktadır. Bunlardan biri UNION SELECT ifadesinden sonra NULL eklemektir. NULL eklememizin sebebi sütunlardaki veri türlerinin birbiriyle uyumlu olması lazımdır. NULL her veri türüyle (string,int vb.) uyumlu olduğundan problem oluşturmayacaktır.

```
' UNION SELECT NULL
' UNION SELECT NULL, NULL
```

Yukarıdaki gibi SELECT ' in ardından NULL eklenerek sütun sayısı kontrol edilir. NULL sayısı ile orijinal sorgunun sütun sayısı eşleştiğinde veri tabanı orijinal sorgunun cevabının sonuna, her sütunu NULL değerlerden oluşan ek bir satır eklenmiş cevap döndürür. Eğer NULL sayısı ilk sorgudaki sütun sayısı ile eşleşmezse hata meydana geleceğinden kaç tane sütun olduğunu anlarız. Uygulama her zaman hata döndürebilir. Bu durumda bu metodu kullanmak bizi bir sonuca ulaştırmayacaktır.

Diğer metod ise ORDER BY keywordü ile sütun sayısını bulmaya çalışırız. Bu ifade tablodan dönen değerlerin farklı sütunlara göre sıralanmasını sağlar. ORDER BY ifadesini kullanırken illa sütun adını bilmenize gerek yoktur. Sütunlar index no ları olarak belirtilebilirler.

```
' ORDER BY 1--
' ORDER BY 2--
' ORDER BY 3--
```

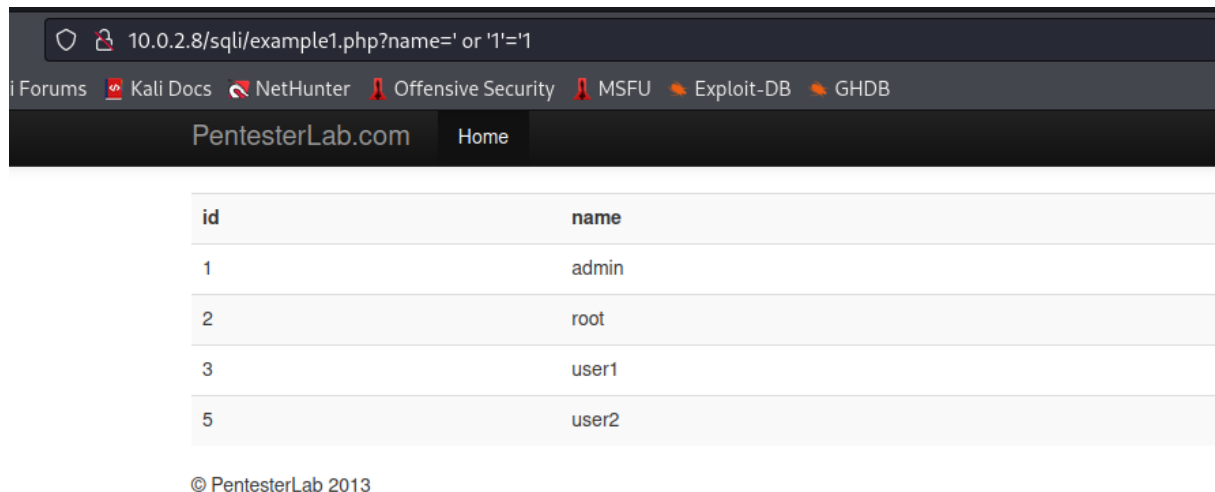
Eğer gerçek sütun sayısından fazla bir sayı belirtirsek veritabanı hata döndürecektir. Bu sayede sütun sayısını anlamış olacağız. Uygulama hatayı response içerisinde gösterebilir, genel bir hata gösterebilir ya da basitçe herhangi bir sonuç göstermeyebilir.

WEB FOR PENTESTER SQL INJECTION

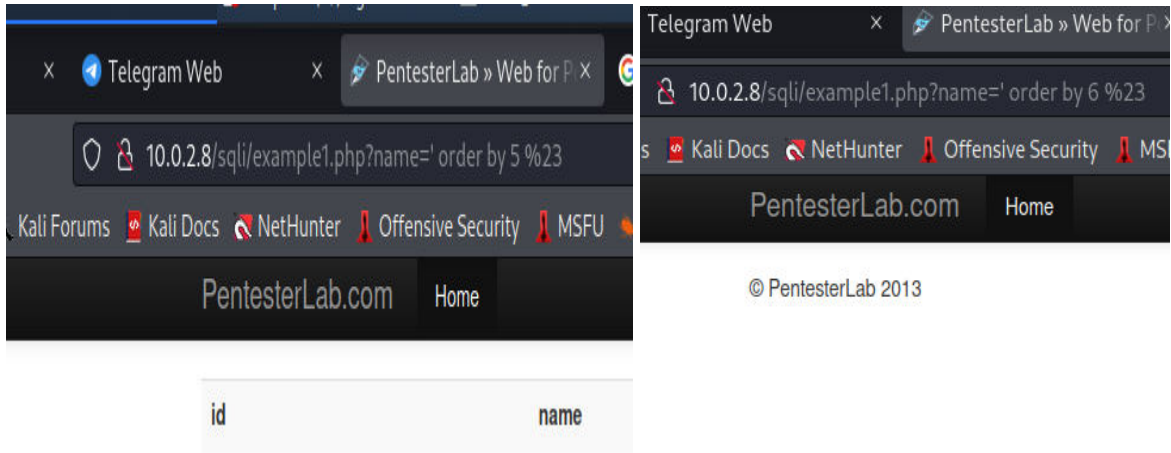
Sqli yaparken arkada çalışan kodları göz önüne alarak önce basit ' or 1=1 -- ifadesini denedim fakat bu şekil sonuç dönmedi. Farklı payloadlar denedim.

Ardından SQLi payloadlardan bildiğim ' or '1'='1 şeklinde name kısmına yazdığımda bütün kullanıcılar karşıma çıktı. Büyük ihtimalle sonradan koyduğum tireler yorum satırı yapmıyordu. Çünkü bu yazdığım payloadı düşününce SELECT * from tablaname WHERE name=" or '1'='1' olunca sorgu kapandığından dolayı kabul etti. Şu payloadı denediğimde de sql sorgusu çalıştı ' or 1=1 or ' yani yorum satırı yapmak istediğim alan çalışmıyor. İnternette baktığımda farklı yorum satırı işaretlerini gördüm onları denedim yine olmadı . Biraz daha baktığımda encode edilmiş halini kullanmam gerektiğini anladım. Tirelerin encode halini yazdım kabul etmedi # ifadesinin encode edilmiş hali %23 ile kabul etti.

Arkada çalışan kodun şu tarz bir sql sorgusu olduğunu düşündüm. SELECT * from tablename WHERE name='\$name'



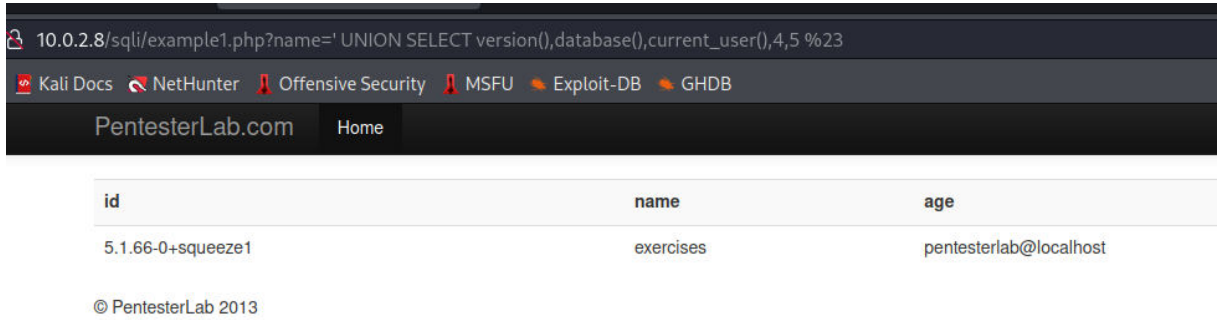
Buraya union attack yapabileceğimi düşündüm. UNION sorgusu için column sayılarının aynı olması gerektiğinden order by sql fonksiyonunu kullandım.



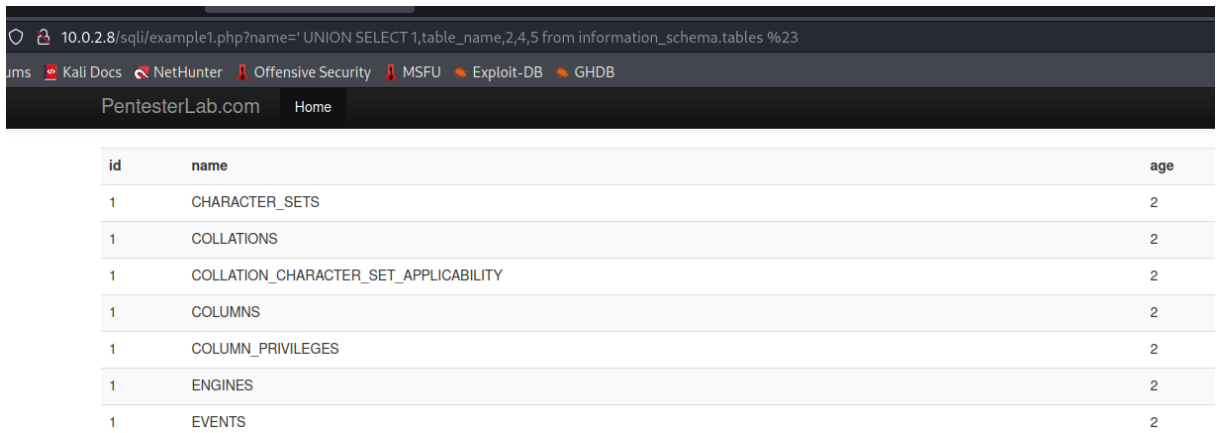
Yukarıda yaptığım order by ile 6 yapınca id name ve age sütunları dönmüyordu yani 6 tane sütun yoktu. Bu sayede sütun sayısının 5 olduğunu anladım sonrasında UNION SELECT kodunu çalıştırarak sonuçları nereye bastırdığını gördüm.

```
' UNION SELECT 1,2,3,4,5 %23
```

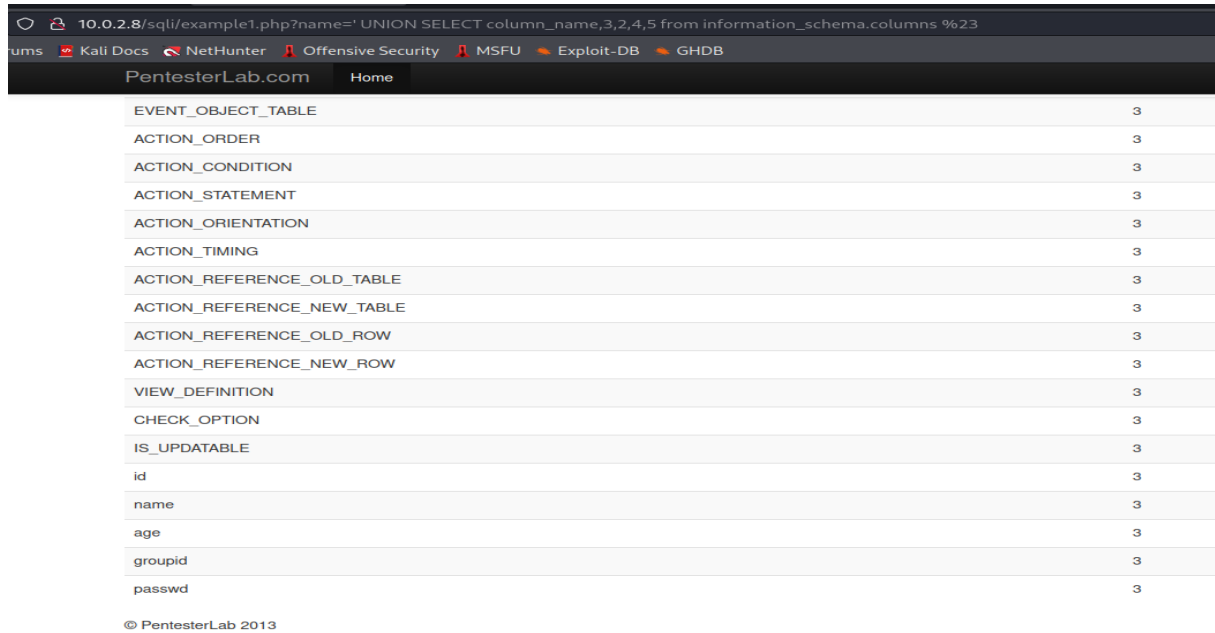
Sonra SQL injection cheat sheetine bakarak version tablo ve databaseyi nasıl getireceğimi öğrendim.



Resimde görüldüğü üzere version ,database ve mevcut kullanıcı bilgisini UNION SQL INJECTION ile bastırabildim.



Burada da veritabanında kullanılan tabloları bastırdım.



EVENT_OBJECT_TABLE	3
ACTION_ORDER	3
ACTION_CONDITION	3
ACTION_STATEMENT	3
ACTION_ORIENTATION	3
ACTION_TIMING	3
ACTION_REFERENCE_OLD_TABLE	3
ACTION_REFERENCE_NEW_TABLE	3
ACTION_REFERENCE_OLD_ROW	3
ACTION_REFERENCE_NEW_ROW	3
VIEW_DEFINITION	3
CHECK_OPTION	3
IS_UPDATABLE	3
id	3
name	3
age	3
groupid	3
passwd	3

© PentesterLab 2013

Column Names

Yukarıdaki resimde kolon adlarının bastırılmasını görüyorsunuz.

Example 2

Bu örnekte url kısmındaki girdi yerine boşluk bırakmak engellenmişti. Burada yukarıda yorum satırı encoding edildiği için bununla alakalı olabileceğini düşündüm HTML url encoding yazıp bana boşluk bırakmamı sağlayacak karakterlere göz attım. Tab karakteri ve yeni satır karakterleri vardı. İkisinin encode edilmiş halini girince yine zafiyet gerçekleşti.

Yani name=root'%09or%091=1'%23 ifadesiyle zafiyet gerçekleşti.

```
<?php
require_once('../header.php');
require_once('db.php');

if (preg_match('/ /', $_GET["name"])) {
    die("ERROR NO SPACE");
}

$sql = "SELECT * FROM users where name='";
$sql .= $_GET["name"]."'";

$result = mysql_query($sql);
if ($result) {
```

Example 3

Bu örneği çözmeye çalışırken internette githubta bunların php kodlarının bulunduğunu öğrendim. Nasıl çalıştığını görmek için kodu inceledim. Resimde yer alan regex ifadesinde herhangi white space karakteri bulunursa kod hata döndürüyor. Burayı çözerken internetten yardım aldım çünkü herşeyi denedim boşluk yerine ne yapabilirim diye fakat php'nin yorum satırını denemek hiç aklıma gelmedi. `/**/` yaparak white space oluşmuyor fakat kodda o aralarda boşluk oluşuyor. Bu sayede zafiyet oluştu. Bunu engellemek için direkt sql sorgusunun içine kullanıcıdan aldığı değerleri girmek yerine php'nin pdo kütüphanesini kullanarak `prepare` ve `bind_param`la güvenli hale getirebilir. Aynı zamanda yaptığı regex işlemine `/**/` olmamasını sağlarsa da güvenlik açığı giderilmiş olur. `Escape_string` fonksiyonunu kullansa da boşluklar sayılmayacağından giderilmiş olur.

`name=root'/**/or/**/1=1%23`

```
<?php
require_once('../header.php');
require_once('db.php');
if (preg_match('/\s+/', $_GET["name"])) {
    die("ERROR NO SPACE");
}
$sql = "SELECT * FROM users where name='";
$sql .= $_GET["name"]."'";

$result = mysql_query($sql);
if ($result) {
```

Example 4

Bu örnekte `mysql_real_escape_string` ile tırnak işareti karşı önlem alınmış fakat burada tırnak işaretini kullanmadığı için zafiyet ortaya çıkıyor. Tırnak olmadığından dolayı sorgumuzu `id=2 or 1=1%23` şeklinde yazdığımızda kullanıcılar önümüze geliyor. Bunu engellemek için tırnak ifadesi getirilebilirdi. En genel çözüm dediğim gibi pdo sınıfından `prepare` ile `bind_param`ı kullanmak olabilir.

```
<?php
require_once('../header.php');
require_once('db.php');
$sql="SELECT * FROM users where id=";
    $sql.=mysql_real_escape_string($_GET["id"])." ";
    $result = mysql_query($sql);

    if ($result) {
```

Example 5

Burada integer kontrolü sağlanmış regex ifadesiyle id parametresine sadece integer girebilmemiz istenmiş fakat ^ işareti ile başlangıçta olması istenmiş yani herhangi bir sayı ile başlayıp devamında ne girdiğimiz çok önemli değil. Bu yüzden aynı şekil id=2 or 1=1%23 ifadesiyle yine istediğimiz sonucu çekebiliyoruz.

```
<?php

require_once('../header.php');
require_once('db.php');
if (!preg_match('/^[0-9]+/', $_GET["id"])) {
    die("ERROR INTEGER REQUIRED");
}
$sql = "SELECT * FROM users where id=";
$sql .= $_GET["id"] ;

$result = mysql_query($sql);

if ($result) {
```

Example 6

Bu ifadeye baktığımızda yukarıdakiyle aynı bir regex ifadesi girilmiş fakat bu sefer başlangıçta değil de integer değerın sonda olması isteniyor bunu ne yapabilirim diye düşündüm normalde yaptığımız sorguda 'or 1=1' ifadesinde zaten sondaki değer integer bir ifade sql sorgusunun ekstra bir şey yapmadan çalışabileceğini düşündüm. Sonuçta burada sql' db sini atlatmaya değil php'nin regex ifadesinden başarılı bir şekilde geçmeye çalışıyoruz. Bu yüzden sonunda eklediğimiz 1 değeri regexten başarıyla geçti. Bunu engellemek için mysql_real_escape_string ifadesi kullanılabilirdi. Regexin başınıda kontrol edip boşlukları bypass edemeyecek şekilde ayarladıktan sonra.

```
<?php

require_once('../header.php');
require_once('db.php');
if (!preg_match('/^[0-9]+$/', $_GET["id"])) {
    die("ERROR INTEGER REQUIRED");
}
$sql = "SELECT * FROM users where id=";
$sql .= $_GET["id"] ;

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


Example 7

Kodu incelediğimiz zaman yine integer gerektiren bir regex ifadesi ve ayrıca m ifadesi var m'nin anlamı multiline search yani bütün satırda istenilen ifade varmı diye inceleniyor. Eğer herhangi bir yerinde integerdan farklı bir ifade yer alıyorsa ERROR kısmına giriyor. Bunu atlatmak için çok uğraştım burada da internetten yardım aldım en son. İşin içinden çıkamadım. Çünkü regex php yazdığımda m ifadesinin her bir satırı gözden geçirdiği yazıyordu yanlış anlamam sonucunda baya uğraştım. Buradaki regexi bypass etmek için tek tek satırları kontrol ettiğinden biz zaten tek satır veri yolluyoruz bunu aşmak için \n ifadesinin HTML encode edilmiş halini id=2 %0A or 1=1 diyerek atlatabiliyoruz. Sonunda integer değerle bitmesini istediği için 1=1 'deki 1 buna izin veriyor. Burada ilk satırda 2 ifadesini verdiğimiz varsayarak doğru kabul edip 2. Satırdaki komutu işletiyor.

```
require_once('../header.php');
require_once('db.php');
if (!preg_match('/^-[0-9]+$/m', $_GET["id"])) {
    die("ERROR INTEGER REQUIRED");
}
$sql = "SELECT * FROM users where id=";
$sql .= $_GET["id"];

$result = mysql_query($sql);

if ($result) {
```

Example 8

Burada gördüğünüz üzere SQL sorgusunda order by ifadesi kullanılmış buradan direkt olarak union veya select sorgusu ile data çekemeyiz. Bunun için araştırma yaptığımda ORDER BY için 2 tane özellik karşılaştım. Bunlardan biri ASC diğeri DESC, bunlar sayesinde sıralanmak isteyen sütun artan veya azalan sıraya göre sıralanabiliyor. Fakat normal tırnak işareti kullanılmamış. ` bu tırnak işaretine yer verilmiş dikkat etmek gereken önemli bir husus çünkü kodu görmeden çözmeye çalıştığımda baya uğraştırmıştı. Bunu ekledikten sonra sorgudan çıkıyor ASC %23 ifadesini eklemekle birlikte sql sorgusundaki zafiyet gerçekleşiyor. Burada escape_string ve regex ile pdo suz koruma sağlanabilir.

```
require_once('../header.php');
require_once('db.php');
$sql = "SELECT * FROM users ORDER BY `";
$sql .= mysql_real_escape_string($_GET["order"])."`";
$result = mysql_query($sql);
```

Example 9

Burada tırnak işareti kullanılmamış bundan dolayı `my_real_escape_string` fonksiyonu tırnak işareti gibi işlemler eklemediğimiz için girdiğimiz sorguyu yakalayamıyor. Direkt sorguyu koda dahil edince de zafiyet ortaya çıkıyor. Bunu engellemek için tırnak işareti parametre girdiğimiz yere eklenebilir.

```
require_once('../header.php');
require_once('db.php');
$sql = "SELECT * FROM users ORDER BY ";
$sql .= mysql_real_escape_string($_GET["order"]);
$result = mysql_query($sql);
```