# SQL Server – Link… Link… Link… and Shell: How to Hack Database Links in SQL Server!

Microsoft SQL Server allows links to be created to external data sources such as other SQL servers, Oracle databases, excel spreadsheets, and so on. Due to common misconfigurations the links, or "Linked Servers", can often be exploited to traverse database link networks, gain unauthorized access to data, and deploy shells…

## Introduction to SQL Server Links

Creating a SQL Server link is pretty trivial. Depending on your preference, it can be done with the "sp_addlinkedserver" stored procedure or SQL Server Management Studio (SSMS). For more information about creating database links visit http://technet.microsoft.com/en-us/library/ms190479.aspx. Typically attackers don't create links. However, they do view and exploit them.

Existing links can be viewed from the "Server Objects->Link Servers" menu in SSMS. Alternatively, they can be listed with the "sp_linkedservers" stored procedure, or by issuing the query "select * from master..sysservers". Selecting directly from the "sysservers" table is the preferred method as it discloses a little more information about the links.

For existing links, there are a few key settings to pay attention to. Obviously the link destination (srvname), type of data source (providername), and whether the link is accessible (dataaccess), are important for exploiting the links. Additionally, outgoing RPC connections (rpcout) need to be enabled on links in order to enable xp_cmdshell on remote linked servers.

There are two major configurations that attackers focus on when hacking database links. Those include the data source (providername) and the way that links are configured to authenticate. In this blog I'll only be focusing on SQL Server data sources for links that connect to other SQL Servers.

Each of those SQL Server links can be configured to authenticate in a number of different ways. It is possible to disable the links by not providing any connection credentials, it's possible to use the current security context, or it's possible to preset a SQL account and password that will be used for all connections that use the link. Our experience during penetration testing is that after crawling all of the links there is always one or more configured with sysadmin permissions; this allows for privilege escalation from the initial public access to sysadmin access without ever leaving the database layer.

## Selecting Data From SQL Server links

Although only sysadmins can create links, any database user can attempt to access them. However, there are two very important things to understand regarding the usage of links:

1. If a link is enabled (dataaccess set to 1), every user on the database server can use the link regardless of the user's permissions (public, sysadmin, doesn't matter)
2. If the link is configured to use a SQL account, every connection is made with the privileges of that

account (privileges on the link destination). In other words, public user on server A can potentially execute SQL queries on server B as sysadmin.

SQL Server links are very easy to use. Openquery()can be used in T-SQL to select data from database links. For example, the following query enumerates the server version on the remote server:

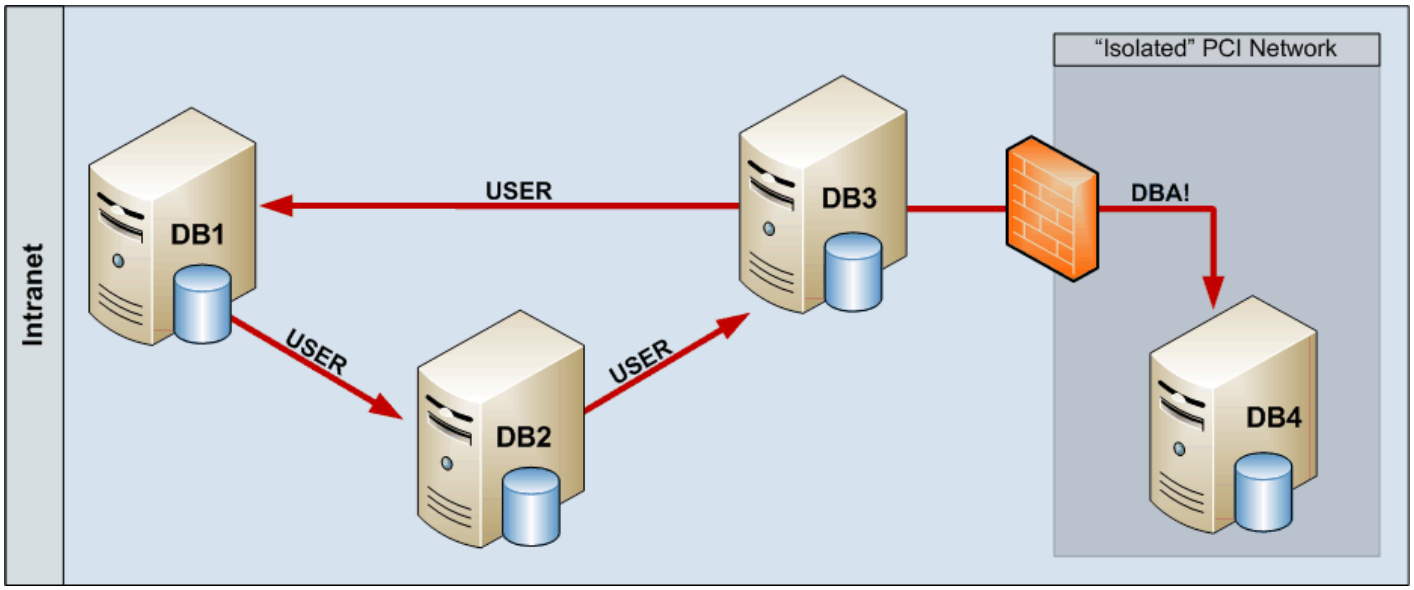
```
select version from openquery("linkedserver", 'select @@version as version');
```

It is also possible to use openquery to execute SQL queries over multiple nested links; this makes link chaining possible and thus allows the exploitation of link trees:

```
select version from openquery("link1",'select version from
openquery("link2",''select @@version as version'')')
```

The same way it's possible to nest as many openquery statements as necessary to access all the linked servers. The only "problem" is that every nested query has to use twice as many single quotes as the outer query; writing queries gets quite cumbersome when you have to use 32 single quotes around every string.

The following image shows an example of a typical linked database network. A user with public privilege access to DB1 can follow the database link to DB2 (user level permissions), and from DB2 to DB3 (user level permissions). Now, it's possible to follow the link from DB3 back to DB1 (user level permissions) or the link to DB4. As this link is configured with elevated privileges, following the link chain from DB1 to DB2 to DB3 to DB4 gives the (originally non-privileged) user sysadmin permissions on DB4 which is located in an "Isolated" network zone.




Database links can be queried using alternative syntax as well but it doesn't allow queries over multiple links; also, actual exploitation requires rpcout to be enabled for the links and as rpcout is disabled by default this is somewhat unlikely to work that often. An example of the alternative four part naming syntax is below.

```
select name FROM [linkedserver].master.sys.databases
```

## Boring… Give Me a Shell Already!

One last think to know for the exploitation of linked servers is that even though Microsoft states "OPENQUERY cannot be used to execute extended stored procedures on a linked server" it is possible. The trick is to return some data, end SQL statement, and then execute the desired stored procedure. Below is a basic example of how to execute xp_cmdshell over a database link using openquery():

```
select 1 from openquery("linkedserver",'select 1;exec master..xp_cmdshell ''dir c:''')
```

The query doesn't return the results of xp_cmdshell, but if xp_cmdshell is enabled and the user has the privileges to execute it, it will execute the dir command on the operating system. During penetration tests this approach has allowed us to compromise the underlying operating system and further compromise the targeted IT infrastructure. One easy way to get a shell on the target system is to call PowerShell (if the OS has it installed), inject reverse meterpreter stager into memory, and wait for it to call back home. The general process works as follows:

1. Create a PowerShell script to execute your Metasploit payload using the techniques from http://www.exploit-monday.com/2011_10_16_archive.html.
2. Next, Unicode encode the script with the programming language of your choice.
3. Then, base64 encode the string with the programming language of your choice.
4. Finally, execute the "powershell –noexit –noprofile –EncodedCommand " command via xp_cmdshell.

If xp_cmdshell is not enabled on a linked server, it may not be possible to enable it even if the link is configured with sysadmin privileges. Any queries executed via openquery are considered user transactions which don't allow reconfigure to be run. Enabling xp_cmdshell using sp_configure does not change the server state without reconfigure and thus xp_cmdshell will stay disabled. If rpcout is enabled for all links on the link path, it is possible to enable xp_cmdshell using the following syntax:

```
EXECUTE('sp_configure ''xp_cmdshell'',1;reconfigure;') AT LinkedServer
```

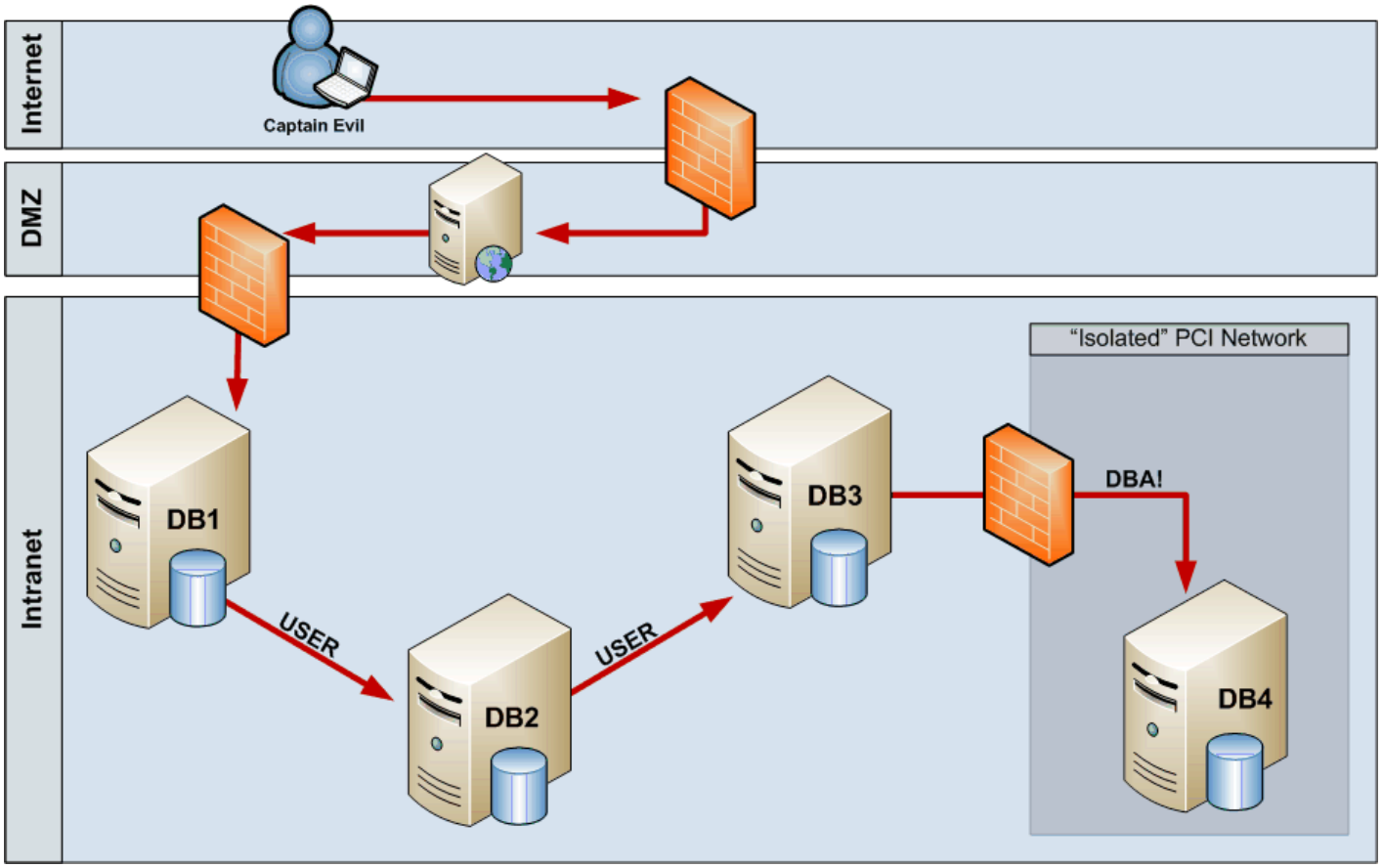
But like I mentioned before, rpcout is disabled by default so it's pretty unlikely to work over long link chains.

## Exploiting SQL Server Links Externally

While database links can be a way to escalate privileges after getting authenticated access to a database internally, a more serious risk is introduced when linked servers are available externally. SQL injection vulnerabilities are still very common and successful injection attack allows arbitrary SQL query execution on the database server. If the web application 's database connection is configured with least privilege (pretty common) it is not trivial to escalate permissions to the internal network where the database server is likely located. However, like mentioned before, any user regardless of their privilege level is allowed to use the preconfigured database links. We have found during penetration tests that links configured with sysadmin privileges typically lead to a complete compromise of the internal network environment.

The following image shows an attack path for an external compromise. After identifying a SQL injection

on the DMZ web application server, Captain Evil can start following the links from DB1 to DB2 to DB3 to DB4. And after getting sysadmin permissions on DB4, Captain Evil can execute xp_cmdshell to execute Powershell and shoot back a reverse shell. So by compromising the web application it's possible to gain access to a secure network without any "hacking" within the Intranet. Just by following database links using legitimate (i.e. not blocked by internal ACL etc.) database connections Captain Evil got access to the most critical system (and maybe more).




## Automating the Whole Thing

Scott Sutherland and I wrote two Metasploit exploits to automate the whole process described in this blog: one for direct database connections (and this one could be used for legitimate link auditing as well) and one for SQL injection using error, union, or time based blind injections. The modules crawl all the database links between SQL Servers, identify sysadmin connections, and if xp_cmdshell is enabled, deploy a chosen Metasploit payload (has be a reverse payload as the attacker won't know where the vulnerable database server actually is). The direct exploit (mssql_linkcrawler) has been merged into Metasploit and the other one is waiting for approval (currently available at https://github.com/nullbind/Metasploit-Modules/blob/master/mssql_linkcrawler_sqli.rb).

## Prevention

If you do not need database links, remove them all. It's pretty common to see links going from production environment to development environment. This shouldn't be done in the first place, but even worse is the "who cares about the dev databases, we don't care if the links have sysadmin privileges" attitude. You should care as escalating an attack in the dev environment will eventually give full access

to prod environment too. So keep in mind, all the database links should be configured with the least privilege; restrict access to those databases / tables that are really needed.

## References


- http://technet.microsoft.com/en-us/library/ms188279.aspx
- http://www.databasejournal.com/features/mssql/article.php/3085211/Linked-Servers-on-MS-SQL-Part-1.htm
- http://www.databasejournal.com/features/mssql/article.php/3085211/Linked-Servers-on-MS-SQL-Part-2.htm
- http://www.netspi.com/blog/2012/11/05/owasp-appsec-2012-presentation-sql-server-exploitation-escalation-and-pilfering/
- https://www.owasp.org/index.php/Category:OWASP_SQLiX_Project