

# Wireshark for Pentester: Decrypting RDP Traffic

May 5, 2021 By Raj Chandel

Over the last few years, attackers used the Remote Desktop Protocol (RDP) for accessing unsecured servers and company networks. In ransomware malware attacks since 2017, RDP has become a major vector. Security professionals have focused their attention increasingly on this protocol by writing signatures to detect and prevent attacks of RDP vulnerabilities.

*RDP supports several operating modes to encrypt network traffic, as a proprietary protocol from Microsoft. This encryption, unfortunately, makes it hard to write RDP signatures because the content of RDP is hidden.*

We can, fortunately, develop a test environment that provides the key file to decrypt the packet capture (pcap) of Wireshark's RDP traffic.

**This article shows how the environment is prepared, a decryption key is obtained, and how RDP traffic can be deciphered.**

## Table of Contents

- Prerequisites
- Trace File
- Virtual Environments Setup
- Remove Encrypted Ciphers from RDP Client
- Generate and Download RDP Server's Private Key
- Capture RDP Traffic
- Analyzing and Decrypting Wireshark Traffic
- Forensics with Captured RDP Data
- The RDP Data Protocol
- Security Protocols Used by RDP Server
- Analyze Connectivity between Client & Host
- Analyzing Credentials
- Conclusion

## Prerequisites

- A good understanding of RDP Setup and Usage.
- A virtual environment to run Two Windows host.
- First windows host act as an RDP Client.
- Second windows host act as an RDP Server.
- A Linux OS for Decryption of pfx file

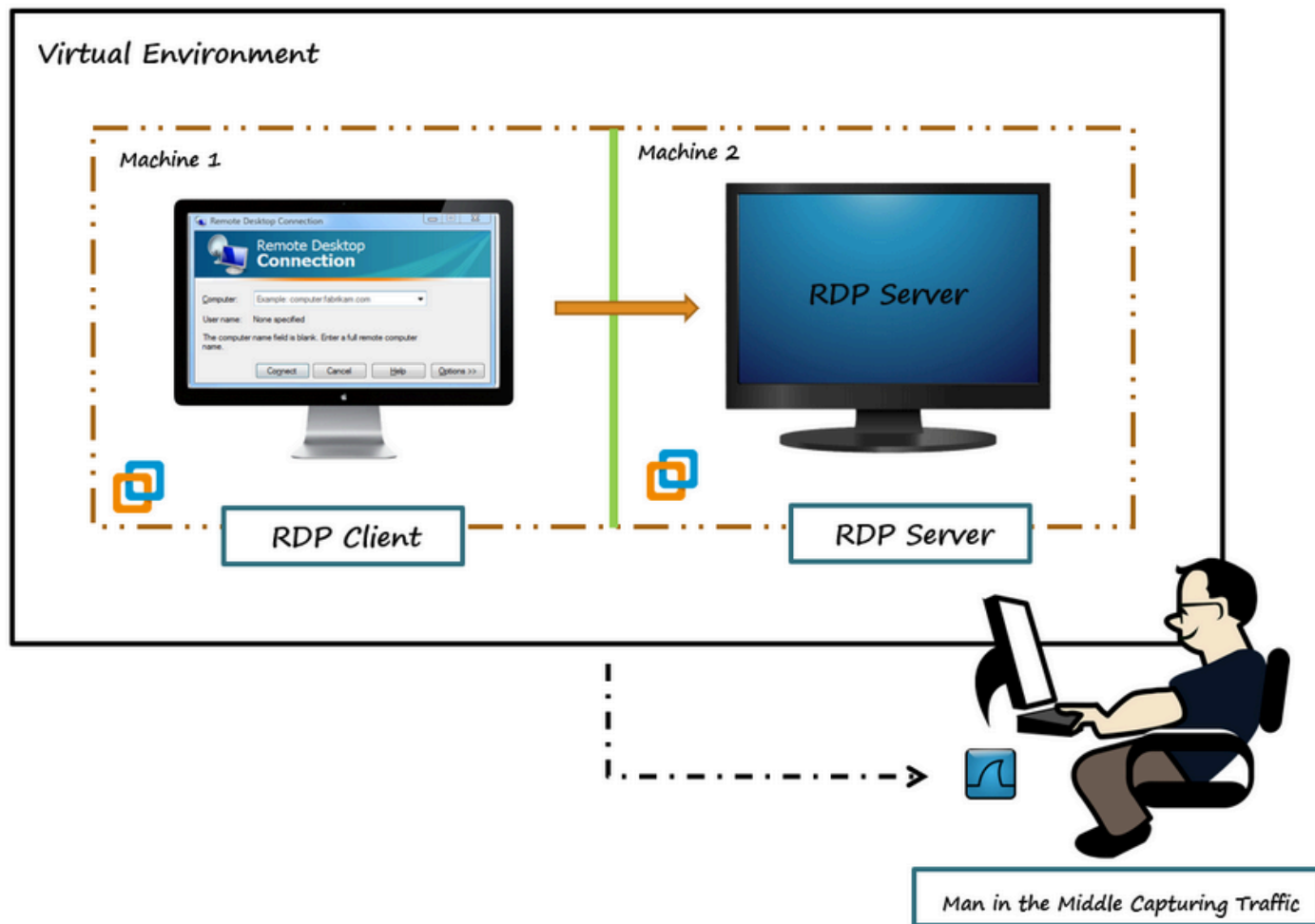
- Wireshark Running in another host

## Trace file

You can download the trace file from [here](#)

## Virtual Environments Setup

VirtualBox or VMware Workstation for Windows and Linux are the two most common virtual environments for this sort of testing. For macOS, VMWare Fusion is used.



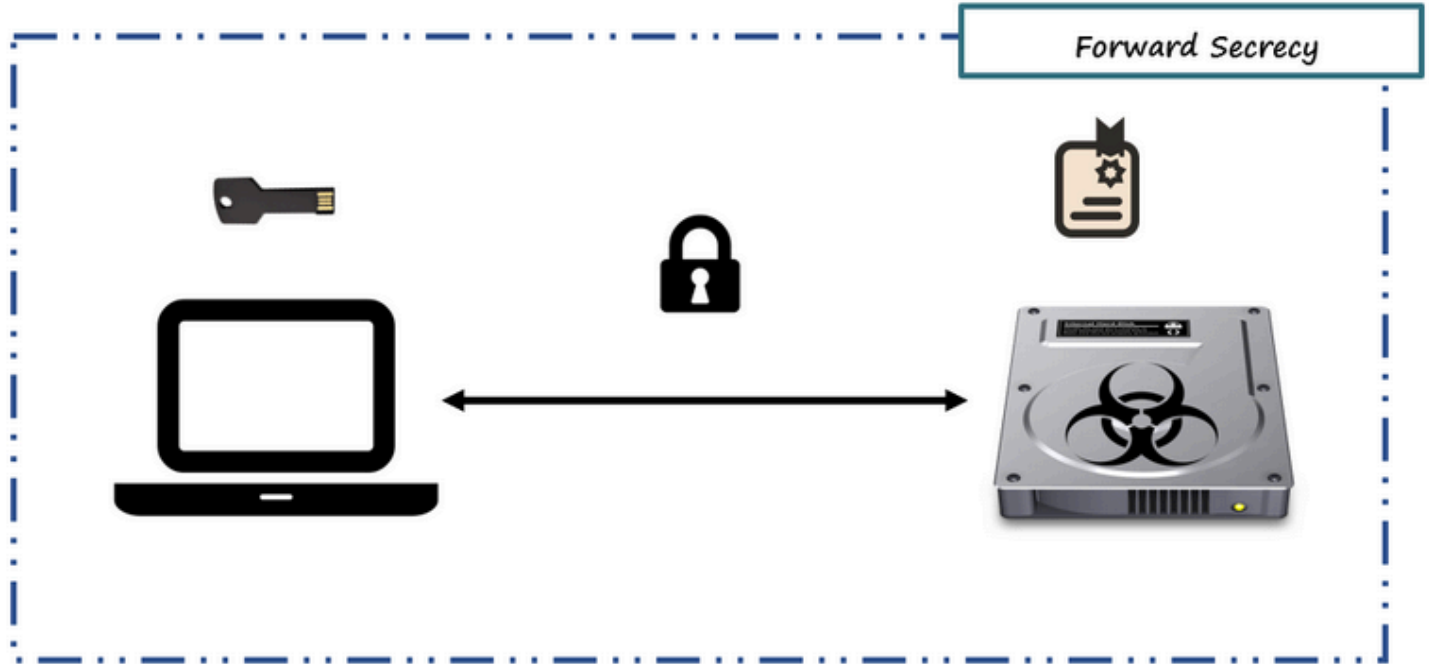
Two Windows 10 hosts were included in our test lab setup. One of the hosts was an RDP server, while the other was an RDP client. We have recorded RDP traffic with Wireshark as a man in the middle between these two hosts.

## Remove Encrypted Ciphers from RDP Client

### Machine 1

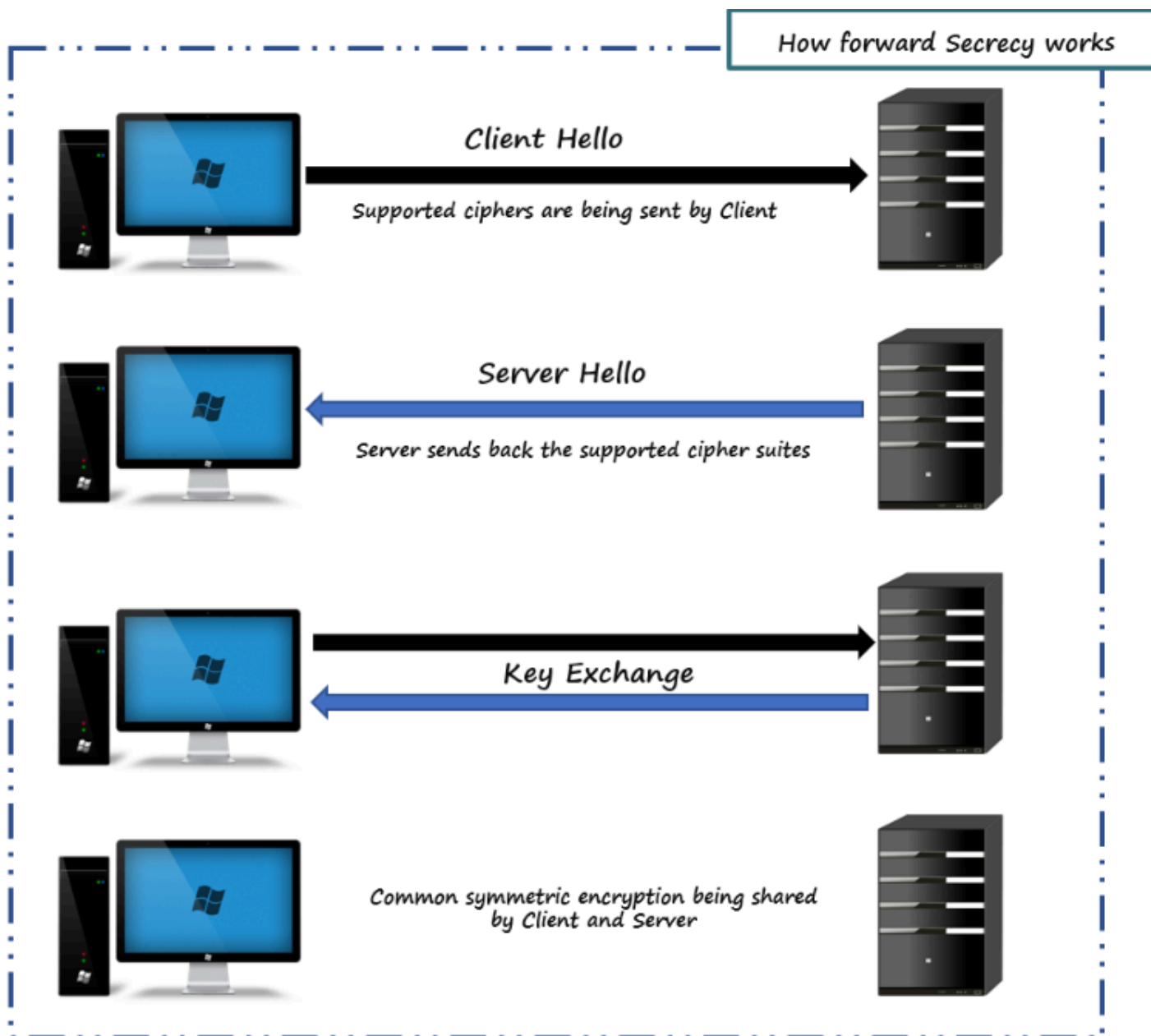
In cryptography, forward secrecy (FS), also known as perfect forward secrecy (PFS), is a feature of specific key agreement protocols that gives assurances that session keys will not be compromised even if long-term secrets are used in the session key exchange are compromised. For HTTPS, the long-term secret is typically the Private

signing key of the server. Forward secrecy protects past sessions against future compromises of keys or passwords. By generating a unique session key for every session a user initiates, the compromise of a single session key will not affect any data other than that exchanged in the specific session protected by that particular key.



The value of forwarding secrecy is limited not only by the assumption that an adversary will attack a server by only stealing keys and not modifying the random number generator used by the server but it is also limited by the assumption that the adversary will only passively collect traffic on the communications link and not be activated using a **Man-in-the-Middle** (MITM) attack. Forward secrecy typically uses an ephemeral Diffie-Hellman key exchange to prevent reading past traffic. The ephemeral Diffie-Hellman key exchange is often signed by the server using a static signing key.

More can be ridden at – [Wikipedia.org](https://en.wikipedia.org/wiki/Forward_secrecy)

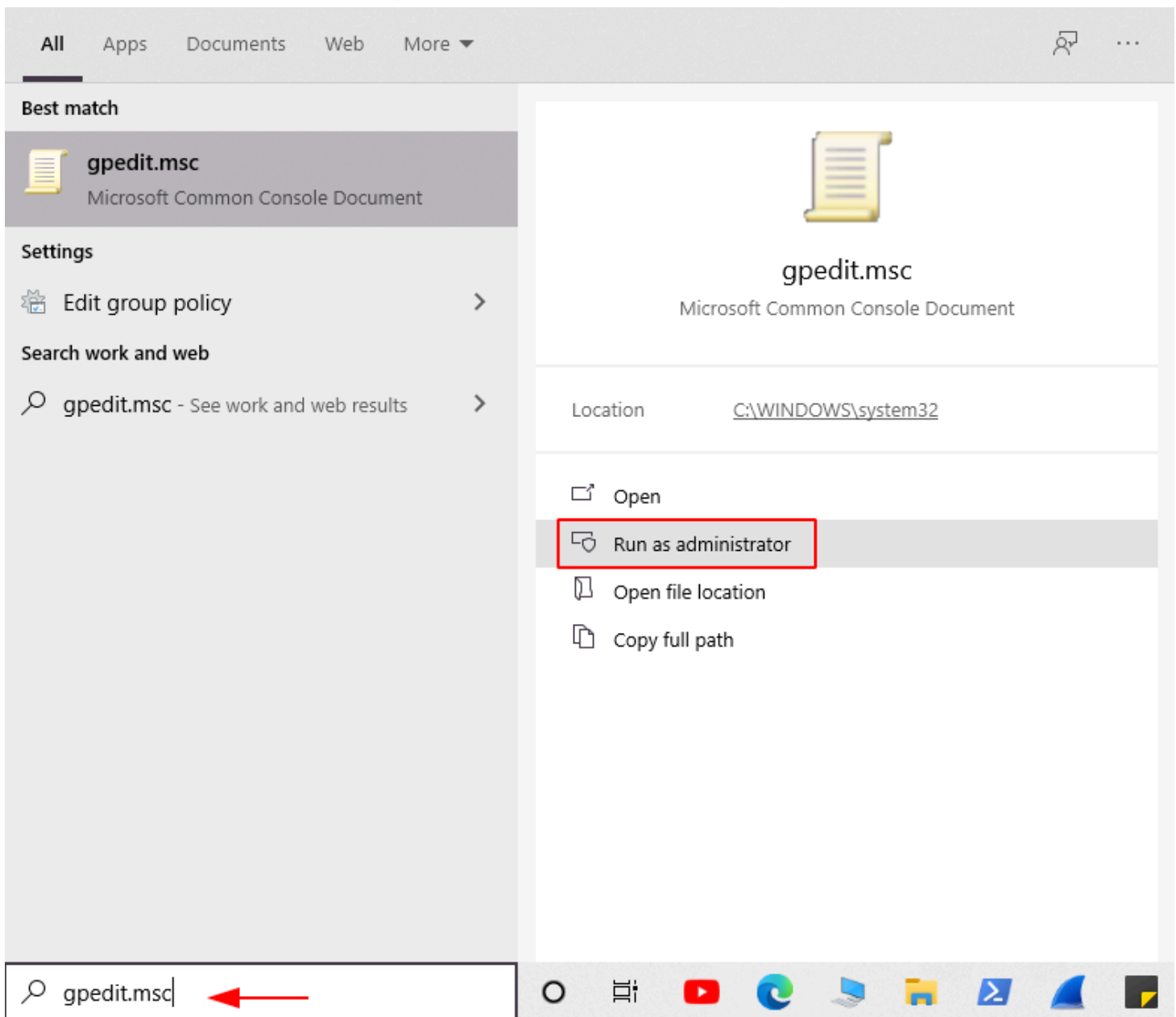


This whole scenario looks like a Perfect Forward Secrecy. Isn't it...?

These types of ciphers create multiple SSL/TLS connection session keys. We can't decrypt SSL/TLS traffic with the forward secrecy using RDP server private key. Therefore, we have to delete settings that support the Forward Secrecy on the RDP client.

**For this, we are using a Windows 10 host as an RDP Client.**

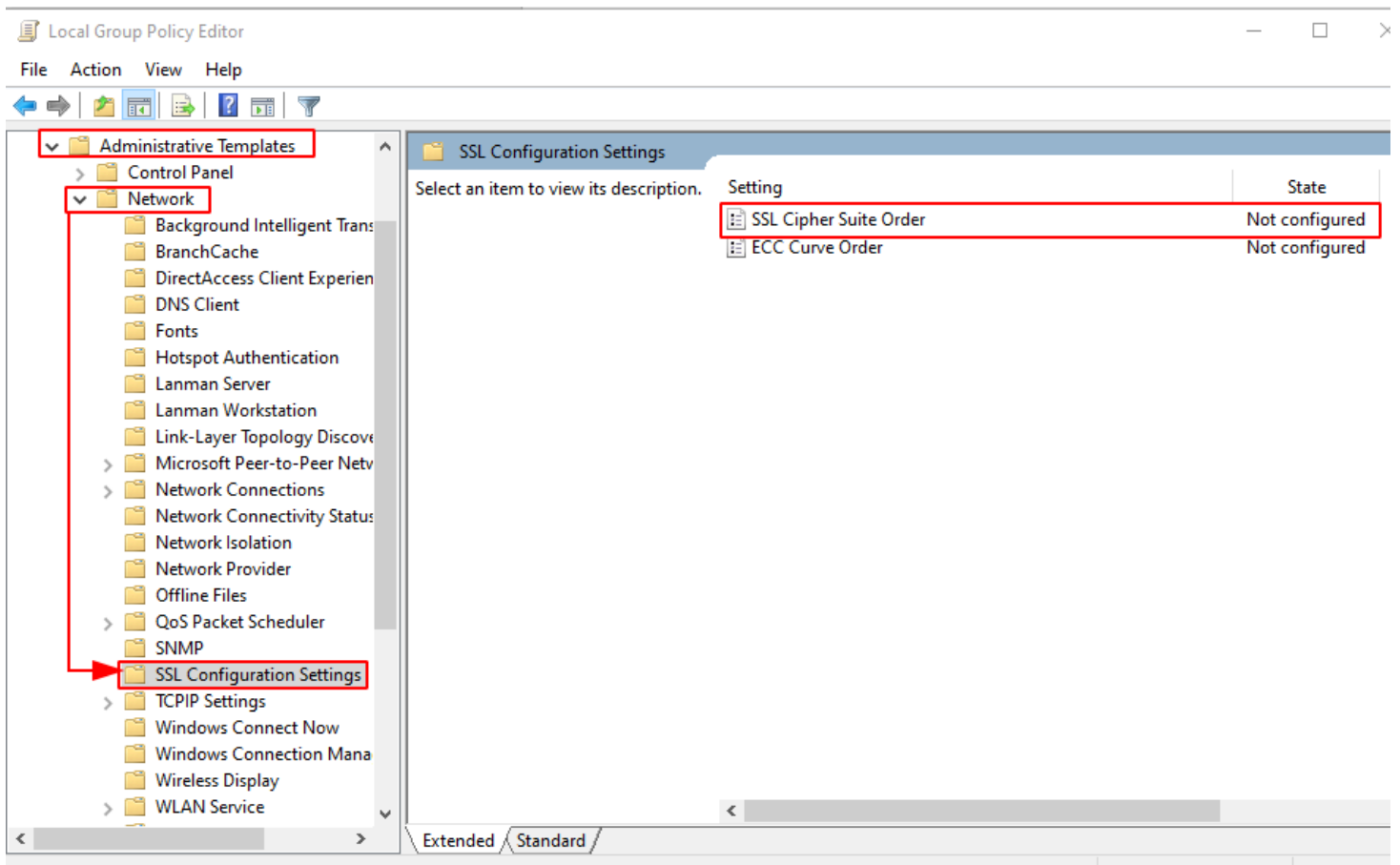
To do this, open Group Policy Management console gpedit.msc as an administrator



After opening the console navigate to the following menu path:

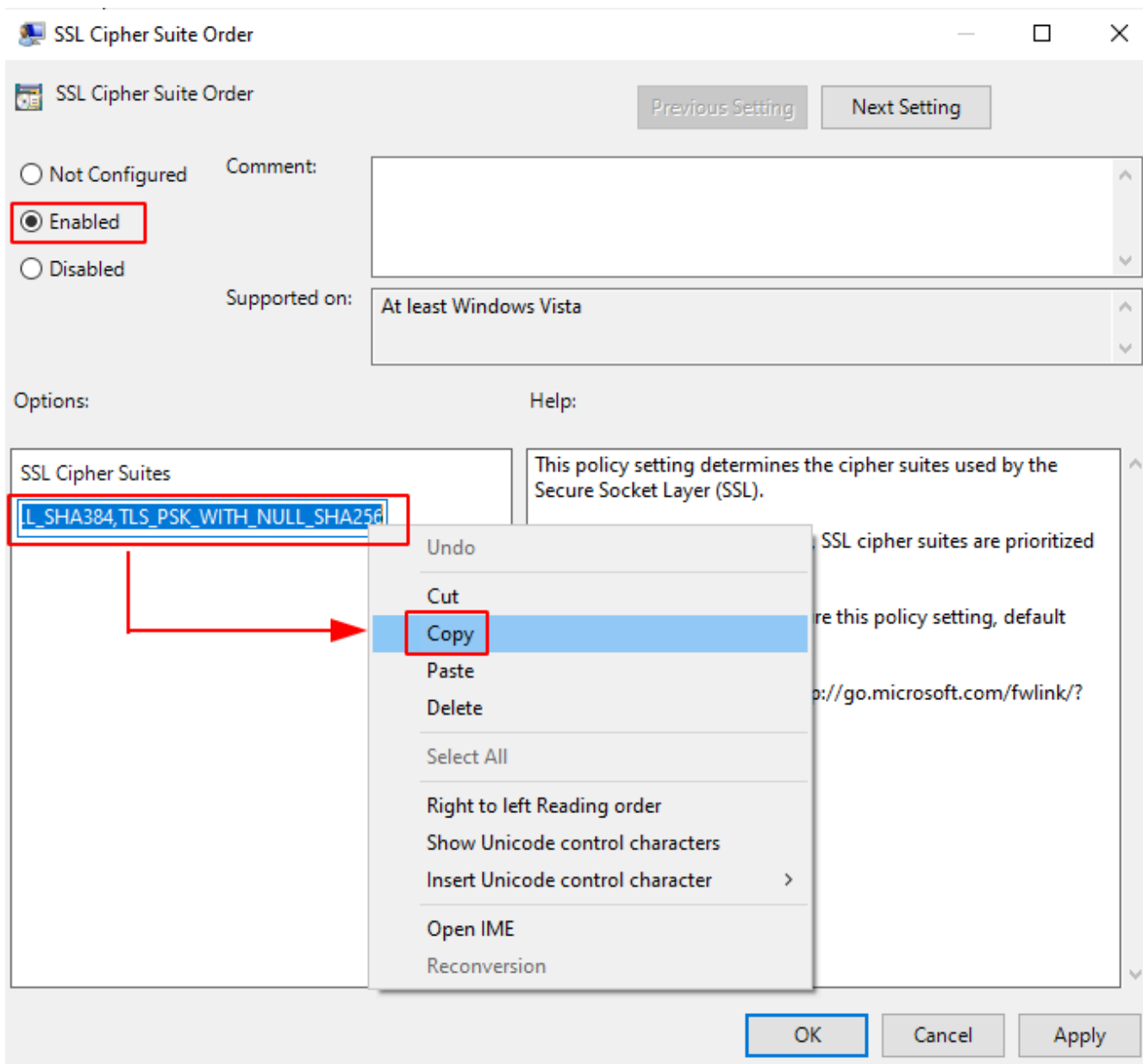
**Administrative Templates > Network > SSL Configuration Settings**

And then Double-click to the entry of the **SSL Cipher Suite Order**

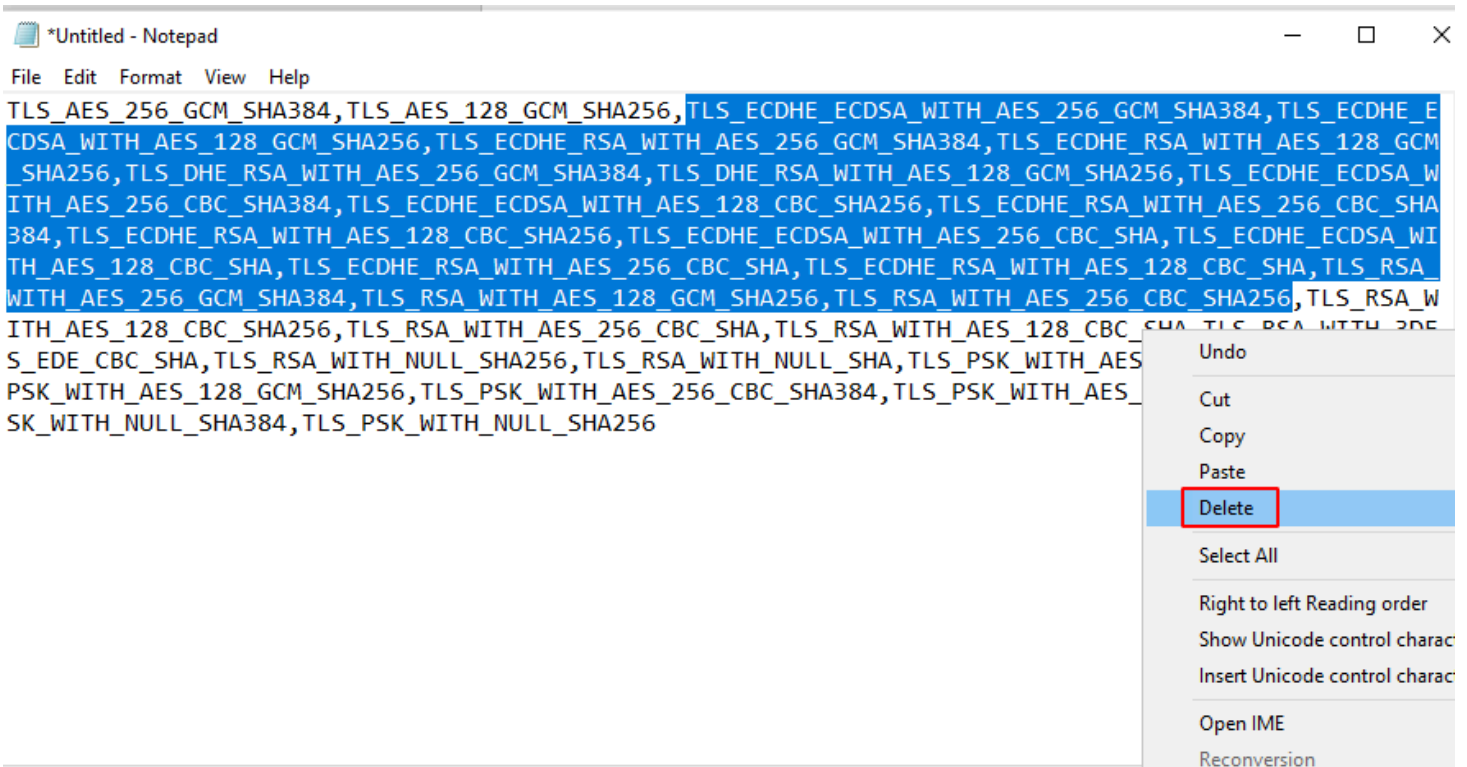


After that select the “**Enable**” option to Enable the SSL Cipher Suite Order.

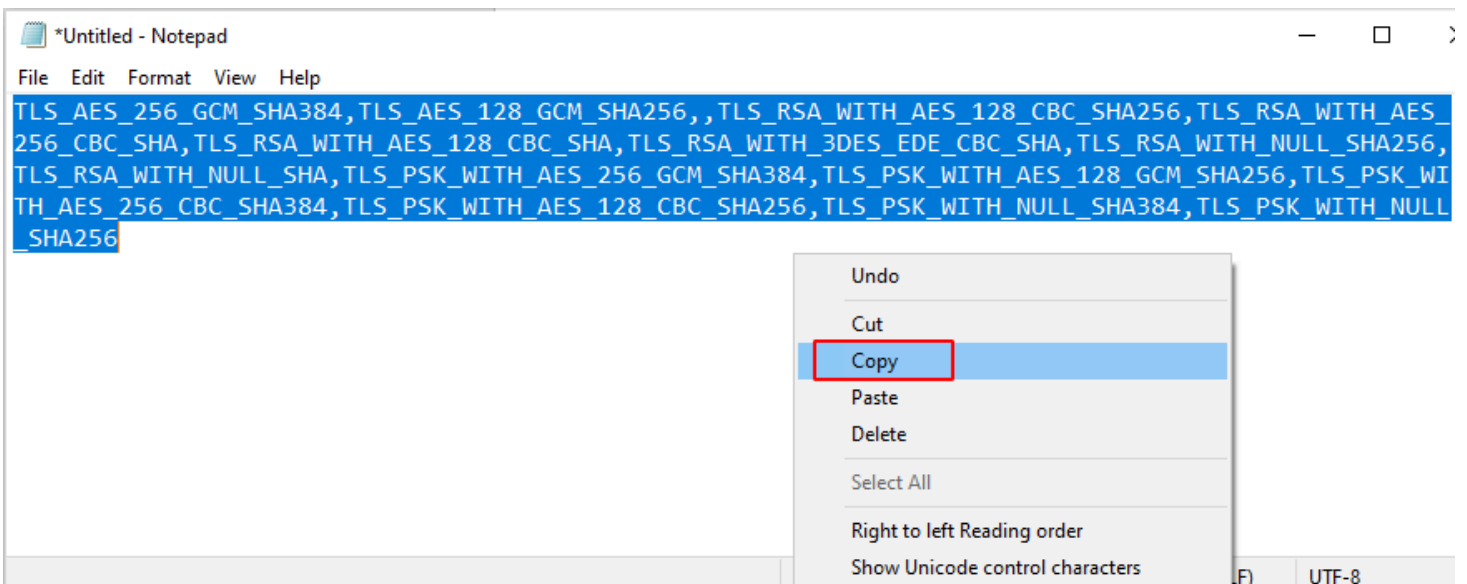
Double-click the list of Ciphers and then select and copy the entire list.



Paste the copied ciphers into a notepad and remove all the ciphers that support Elliptic Curve Cryptography using Diffie-Hellman Ephemeral (ECDHE) or Digital Signature Algorithm (ECDSA) encryption. Remove all the ciphers that contain the word ECDSA and ECHDE. All of these ciphers are managed sequentially, so they were easy to delete from the text.

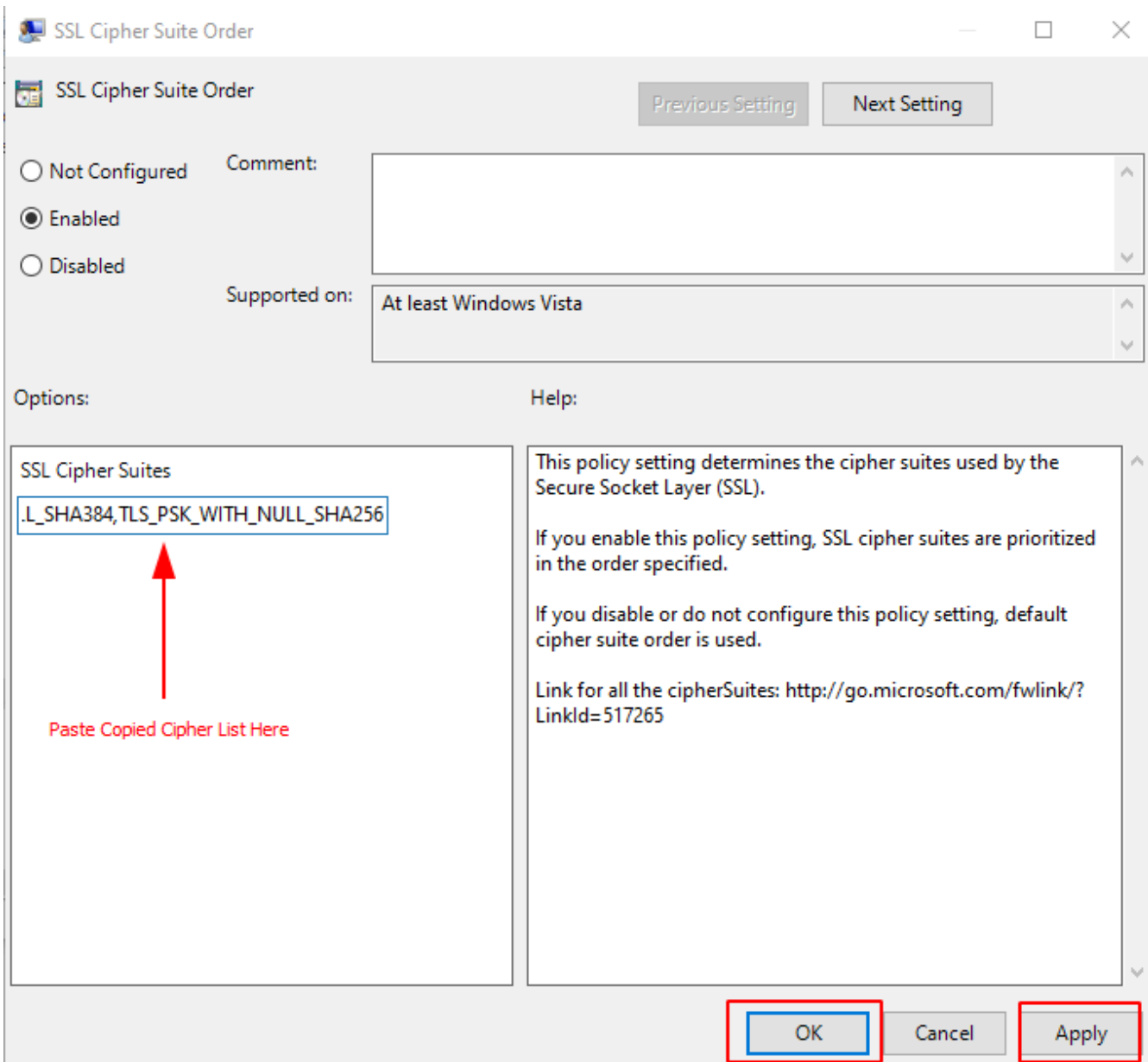


Now, an updated list of ciphers is shown below



Copy the updated ciphers list and paste it back into the SSL Cipher Suites field but make sure you have overwritten the original list and then click on the **Apply** button to save the configuration and hit **OK** to close the window.



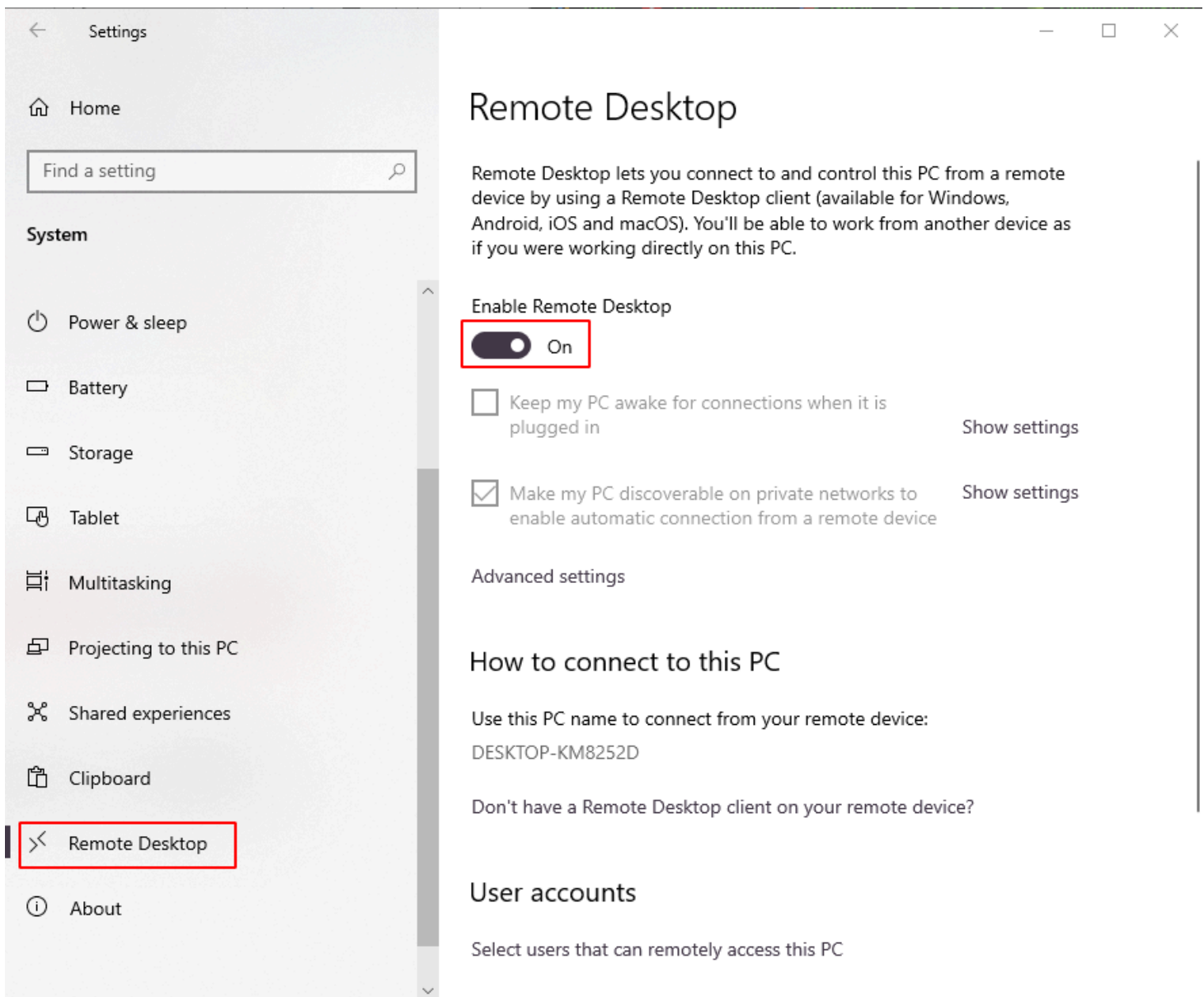


Generate and Download RDP Server's Private Key

## **Machine 2**

We have used another Running Windows 10 Host as an RDP server.

Now our main task is to extract the private key from the host's operating system. First of all, we have to be ensured that our host is acting like an RDP Server or not, if not go to the “**Settings > System > Remote Desktop**” and then “**Enable Remote Desktop**”.



After that, we have to Extract Private Key from its Operating system. Windows don't allow to export of any kind of certificate so we are going to use tools like Mimikatz or Jailbreak.

Mimikatz or Jailbreak is a tool that can dump or export any kind of Certificate easily.

***So, I'm going to show you how this is done from both methods. You can stick with the method which feels you like easy.***

### **Method 1: – Mimikatz**

Mimikatz is a shell for various modules. Run the following commands to export RDP keys or Certificates with private Keys. Run Mimikatz as an administrator.

# Enable “debug” privilege to be able to patch CNG service

```
privilege::debug
```

```
# Patch CNG service lasts until the next reboot
```

```
crypto::cng
```

```
# Patch CAPI library in memory of this process
```

```
crypto::capi
```

```
# Export Remote Desktop certificate(s) with private keys, password is “mimikatz”
```

```
crypto::certificates -systemstore:CERT_SYSTEM_STORE_LOCAL_MACHINE -store:"Remote D
```



```
mimikatz 2.2.0 x64 (oe.eo)

.#####. mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
.## ^ ##. "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

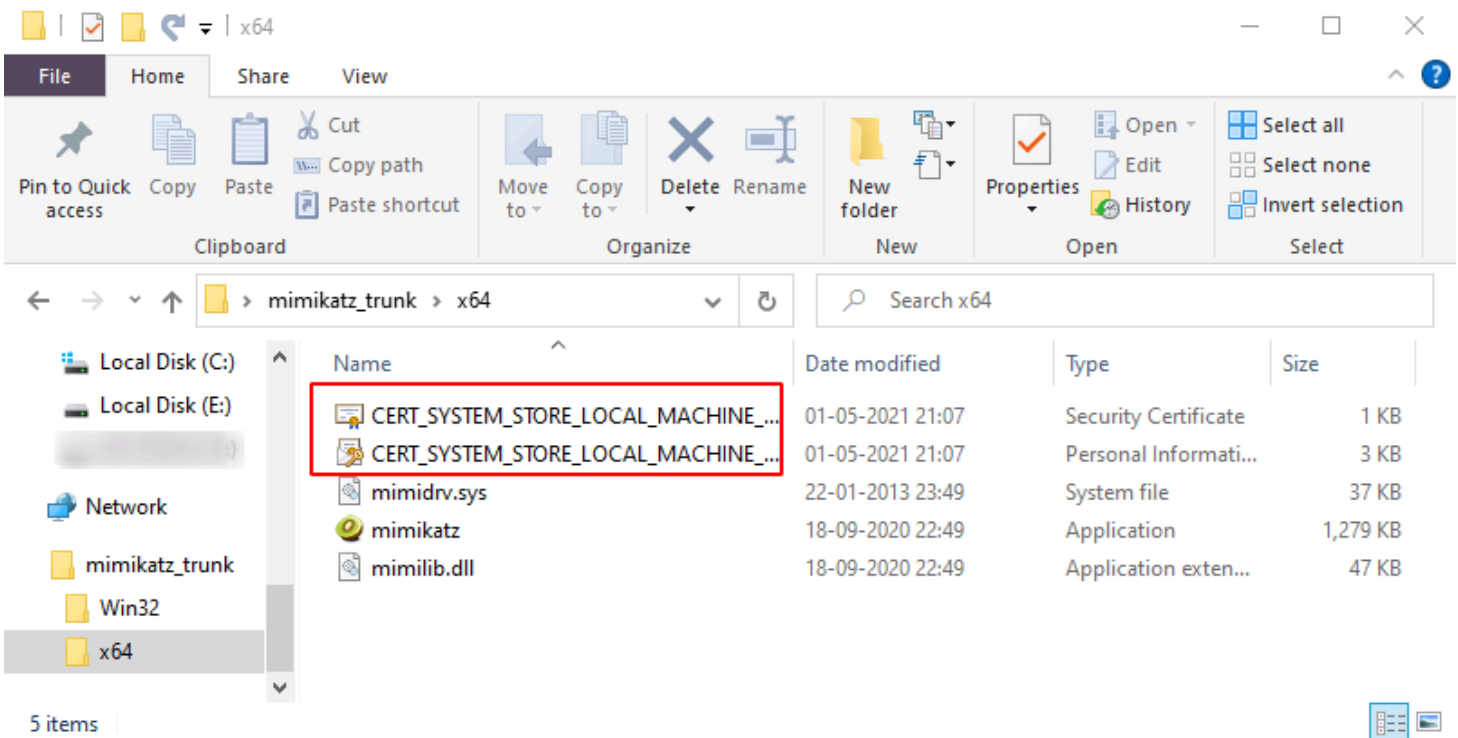
mimikatz # crypto::cng
ERROR kull_m_patch_genericProcessOrServiceFromBuild ; kull_m_patch (0x00000000)

mimikatz # crypto::capi
Local CryptoAPI RSA CSP patched
Local CryptoAPI DSS CSP patched

mimikatz # crypto::certificates -systemstore:CERT_SYSTEM_STORE_LOCAL_MACHINE -store:"Remote Desktop" /export
* System Store : 'CERT_SYSTEM_STORE_LOCAL_MACHINE' (0x00020000)
* Store : 'Remote Desktop'

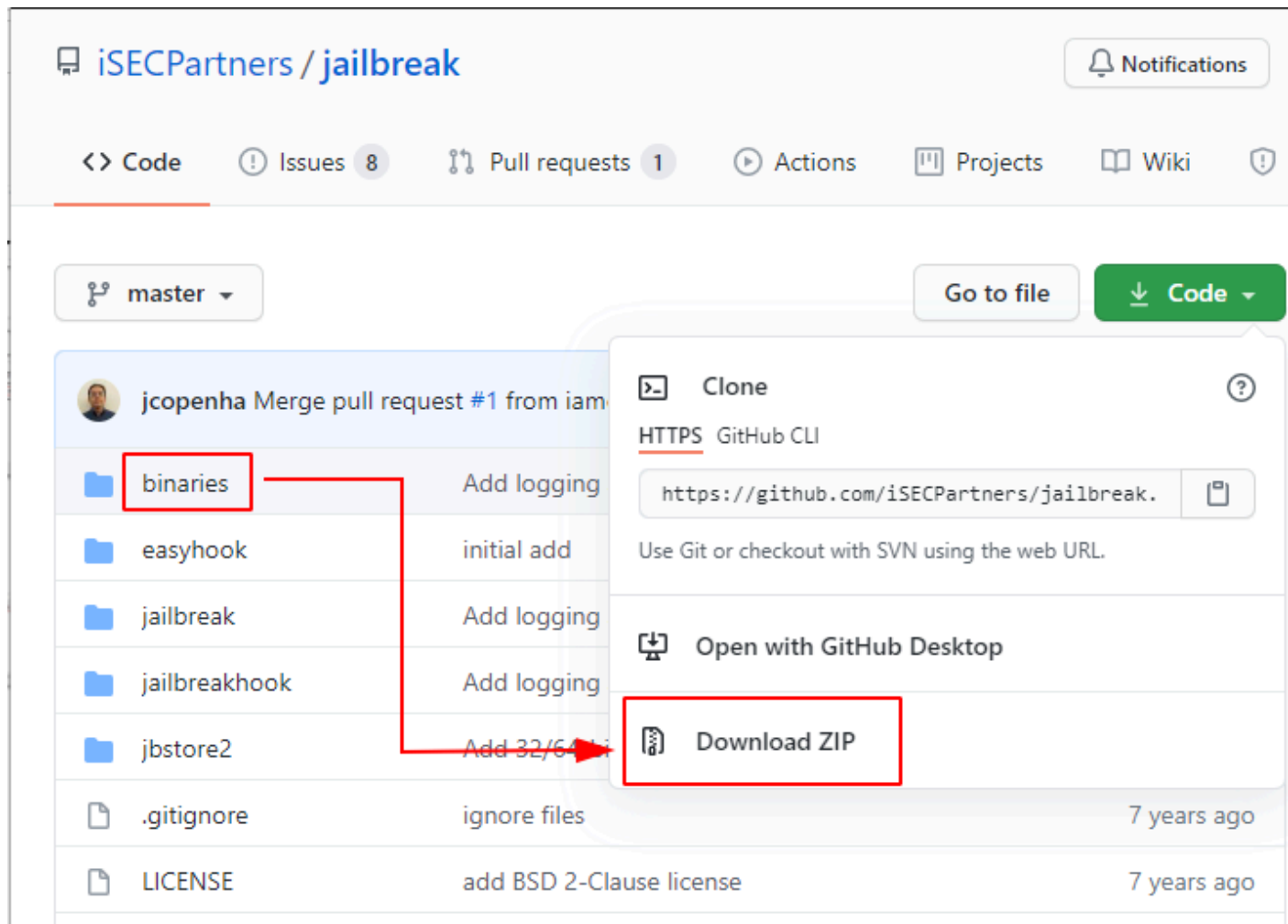
0. DESKTOP-KM8252D
Subject : CN=DESKTOP-KM8252D
Issuer : CN=DESKTOP-KM8252D
Serial : 301f81166e
Algorithm: 1.2.840.113549.1.1.1 (RSA)
Validity : 06-04-2021 15:25:55 -> 06-10-2021 15:25:55
Hash SHA1: 3c11624710663
Key Container : TSSecKeySet1
Provider : Microsoft Enhanced Cryptographic Provider v1.0
Provider type : RSA_FULL (1)
Type : AT_KEYEXCHANGE (0x00000001)
Provider name : Microsoft Enhanced Cryptographic Provider v1.0
Key Container : TSSecKeySet1
Unique name : f686aace6942fb7f7ceb231212eef4a4_943811f5-efda-407d-957d-bdf744d8eabf
Implementation: CRYPT_IMPL_SOFTWARE ;
Algorithm : CALG_RSA_KEYX
Key size : 2048 (0x00000800)
Key permissions: 0000003b ( CRYPT_ENCRYPT ; CRYPT_DECRYPT ; CRYPT_READ ; CRYPT_WRITE ; CRYPT_MAC ; )
Exportable key : NO
Public export : OK - 'CERT_SYSTEM_STORE_LOCAL_MACHINE_Remote Desktop_0_DESKTOP-KM8252D.der'
Private export : OK - 'CERT_SYSTEM_STORE_LOCAL_MACHINE_Remote Desktop_0_DESKTOP-KM8252D.pfx'
```

As you can see Mimikatz successfully dumped the certificate. You can find it in the Directory of Mimikatz.



## Method 2: – Jailbreak

A jailbreak is an iSECPartners tool that can export the RDP certificate of a server. We could extract the private key from the exported certificate. On our newly developed RDP server, we downloaded the following Jailbreak binaries from this GitHub repository to use Jailbreak.



After downloading the Jailbreak utility navigate to the directory of **Jailbreak-master > binaries** and copy the location of the binary folder

This PC > Downloads > jailbreak-master > binaries					Search binaries
Name	Date modified	Type	Size		
EasyHook32.dll	28-12-2015 22:41	Application exten...	225 KB		
EasyHook32.dll.sha256	28-12-2015 22:41	SHA256 File	1 KB		
EasyHook64.dll	28-12-2015 22:41	Application exten...	258 KB		
EasyHook64.dll.sha256	28-12-2015 22:41	SHA256 File	1 KB		
jailbreak32	28-12-2015 22:41	Application	79 KB		
jailbreak32.exe.sha256	28-12-2015 22:41	SHA256 File	1 KB		
jailbreak64	28-12-2015 22:41	Application	91 KB		
jailbreak64.exe.sha256	28-12-2015 22:41	SHA256 File	1 KB		
jailbreakhook32.dll	28-12-2015 22:41	Application exten...	81 KB		
jailbreakhook32.dll.sha256	28-12-2015 22:41	SHA256 File	1 KB		
jailbreakhook64.dll	28-12-2015 22:41	Application exten...	96 KB		
jailbreakhook64.dll.sha256	28-12-2015 22:41	SHA256 File	1 KB		
jbstore2_32	28-12-2015 22:41	Application	77 KB		
jbstore2_32.exe.sha256	28-12-2015 22:41	SHA256 File	1 KB		
jbstore2_64	28-12-2015 22:41	Application	90 KB		
jbstore2_64.exe.sha256	28-12-2015 22:41	SHA256 File	1 KB		

open a Command prompt with administrator privileges and went to the directory of downloaded jailbreak binaries and execute the following command.

```
jailbreak64.exe %WINDIR%\system32\mmc.exe %WINDIR%\system32\certlm.msc -64
```

if you are running the 32-bit version of windows then execute the following command

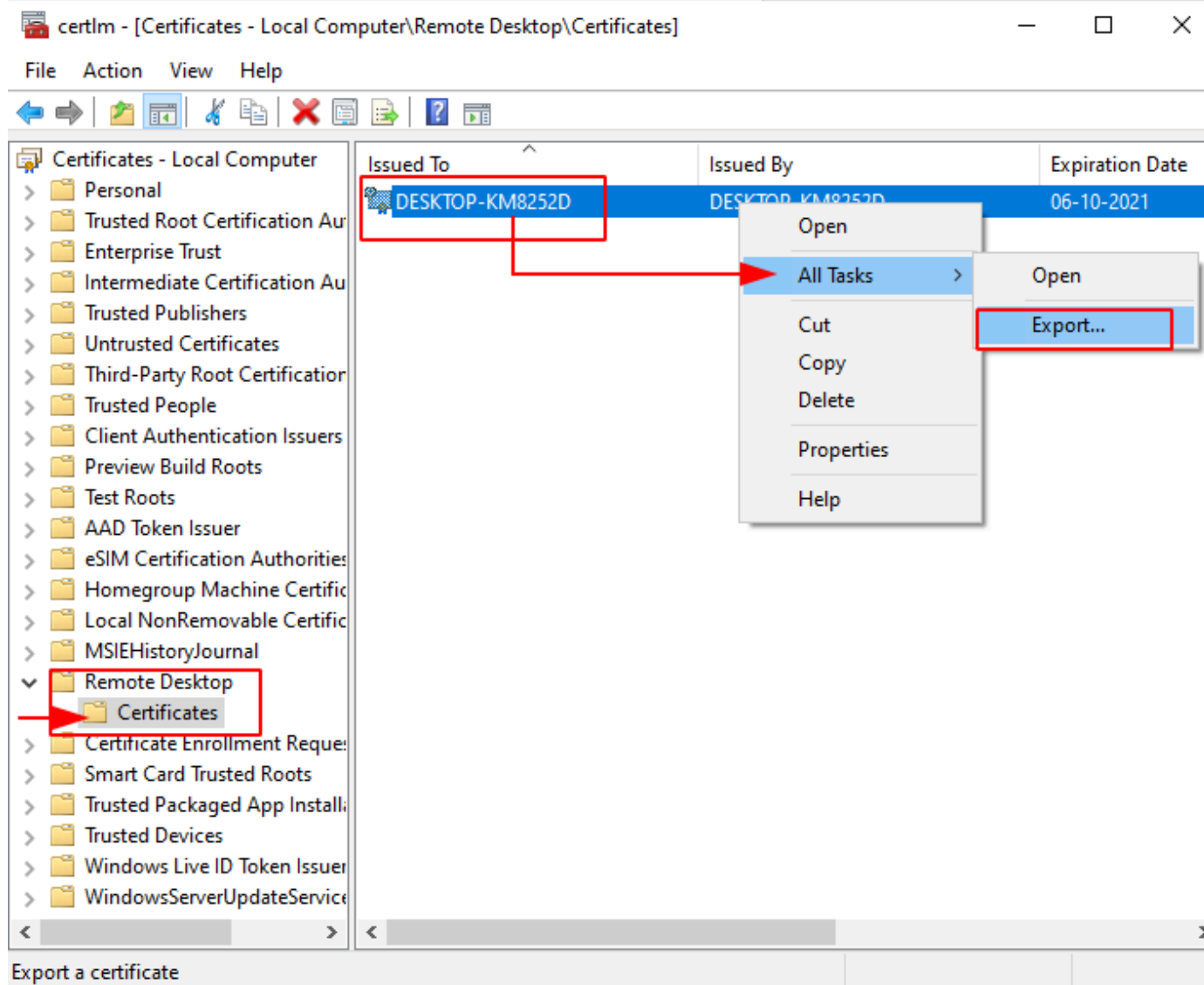
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\vijvi\Downloads\jailbreak-master\binaries
C:\Users\vijvi\Downloads\jailbreak-master\binaries>jailbreak64.exe %WINDIR%\system32\mmc.exe
%WINDIR%\system32\certlm.msc -64
C:\Users\vijvi\Downloads\jailbreak-master\binaries>_
```

jailbreak32.exe %WINDIR%\system32\mmc.exe %WINDIR%\system32\certlm.msc -32

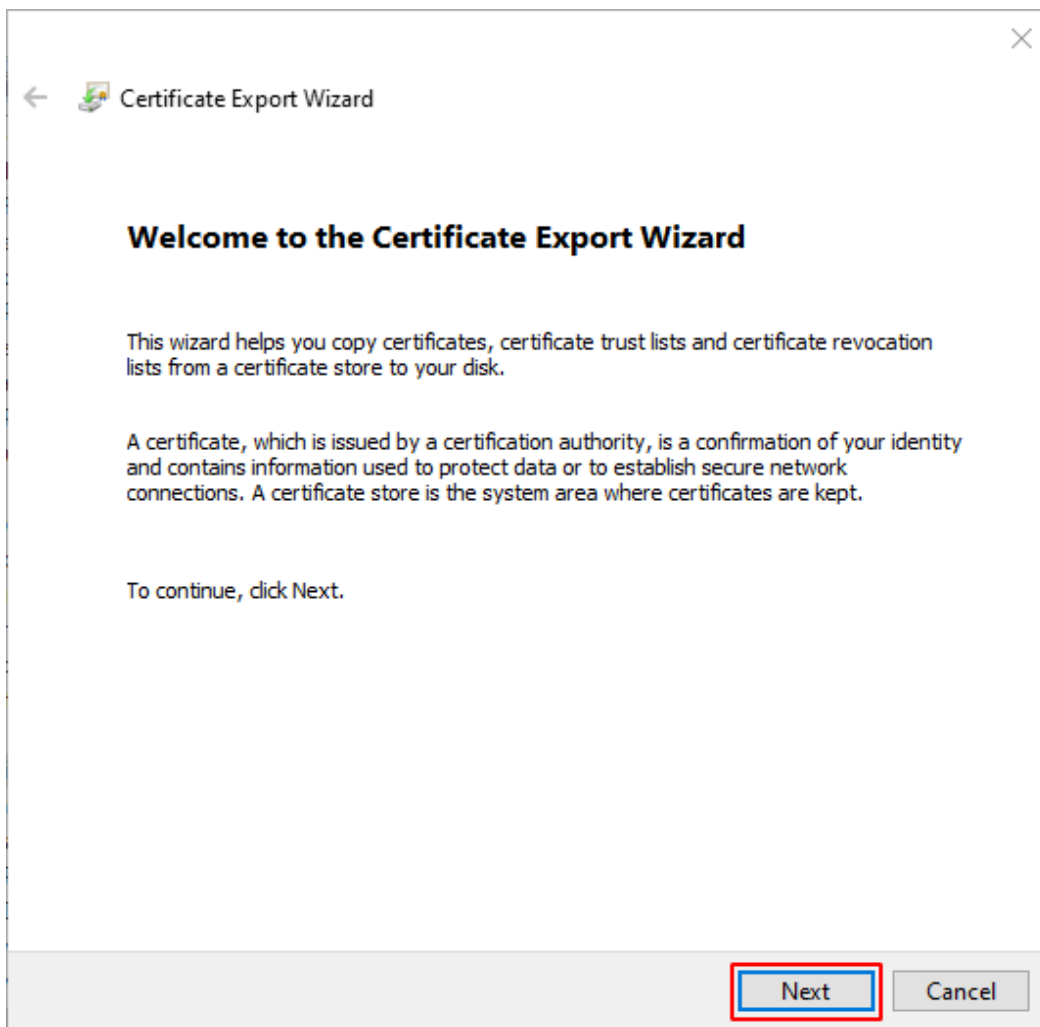
These commands will open up a certificate manager for the local machine. From the left column expand the section of Remote Desktop and navigate to the certificate folder this will show you a certificate with the most expiration

date. Do the right click on the certificate, **select All Tasks**, and then **Export**.



This will open up a Certificate Export Wizard... to export the Certificate just hit the “**Next**” button.





Make sure to select the option to export the private key when exporting the certificate.



**Export Private Key**

You can choose to export the private key with the certificate.

---

Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.

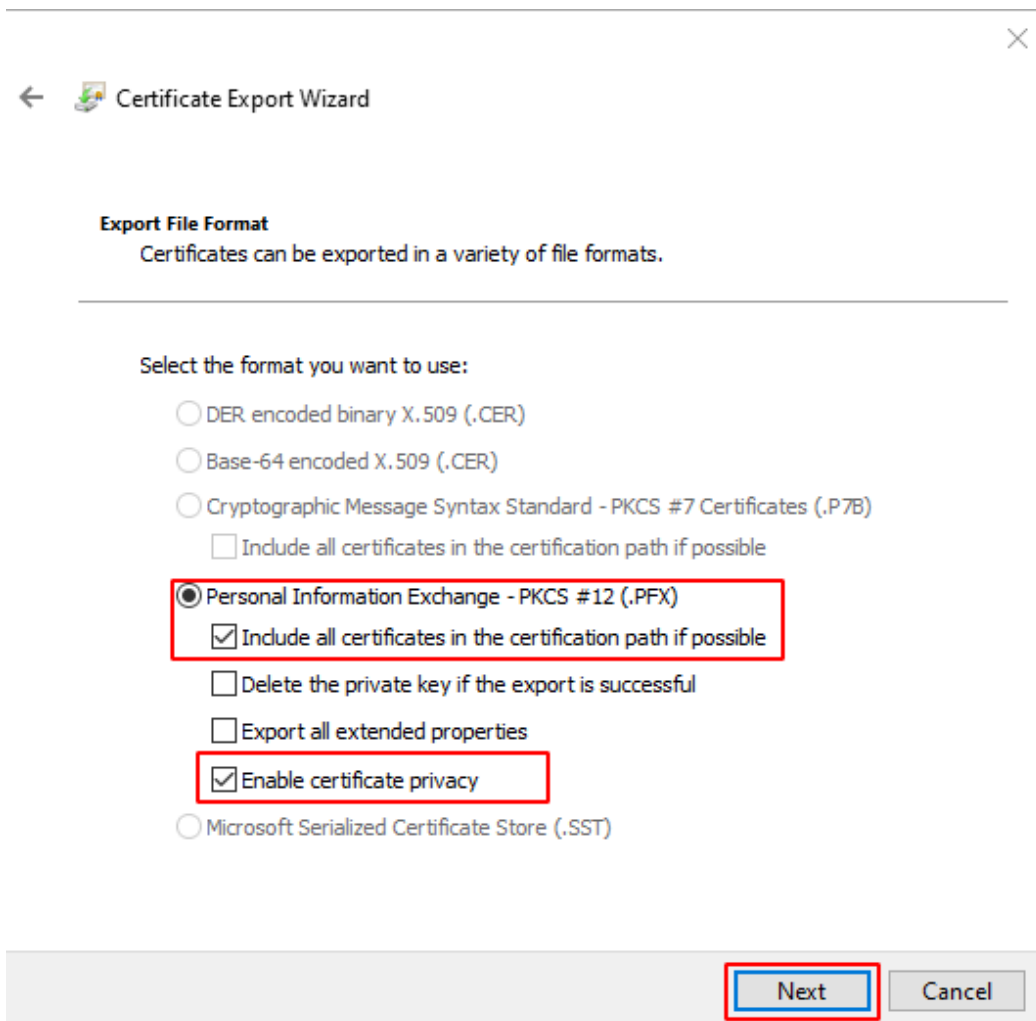
Do you want to export the private key with the certificate?

☒ Yes, export the private key

☐ No, do not export the private key

Next

Cancel



The image shows a Windows dialog box titled "Certificate Export Wizard". It has a back arrow icon on the left and a close 'X' icon on the top right. The main heading is "Export File Format" with a subtext "Certificates can be exported in a variety of file formats." Below this, it says "Select the format you want to use:". There are five radio button options: "DER encoded binary X.509 (.CER)", "Base-64 encoded X.509 (.CER)", "Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)", "Personal Information Exchange - PKCS #12 (.PFX)", and "Microsoft Serialized Certificate Store (.SST)". The "Personal Information Exchange - PKCS #12 (.PFX)" option is selected and highlighted with a red rectangle. Below this option, there are three checkboxes: "Include all certificates in the certification path if possible" (checked and highlighted with a red rectangle), "Delete the private key if the export is successful" (unchecked), and "Export all extended properties" (unchecked). Below these, there is another checkbox "Enable certificate privacy" (checked and highlighted with a red rectangle). At the bottom right, there are "Next" and "Cancel" buttons, with the "Next" button highlighted by a red rectangle.

← Certificate Export Wizard

**Export File Format**  
Certificates can be exported in a variety of file formats.



Select the format you want to use:

- ☐ DER encoded binary X.509 (.CER)
- ☐ Base-64 encoded X.509 (.CER)
- ☐ Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
  - ☐ Include all certificates in the certification path if possible
- ☒ Personal Information Exchange - PKCS #12 (.PFX)
  - ☒ Include all certificates in the certification path if possible
  - ☐ Delete the private key if the export is successful
  - ☐ Export all extended properties
  - ☒ Enable certificate privacy
- ☐ Microsoft Serialized Certificate Store (.SST)

Next Cancel

We could only export the certificate as a PKCS #12 (.PFX) file for our host.

The certificate needed to have a password in the next step. We didn't have any difficulty criteria, so we went with a simple password.

  Certificate Export Wizard

**Security**  
To maintain security, you must protect the private key to a security principal or by using a password.

☐ Group or user names (recommended)

Add

Remove

☒ Password:

...

Confirm password:

...


Encryption: TripleDES-SHA1

Next

Cancel

After that provide a directory where you want to save the certificate.



←  Certificate Export Wizard

**File to Export**

Specify the name of the file you want to export

File name:

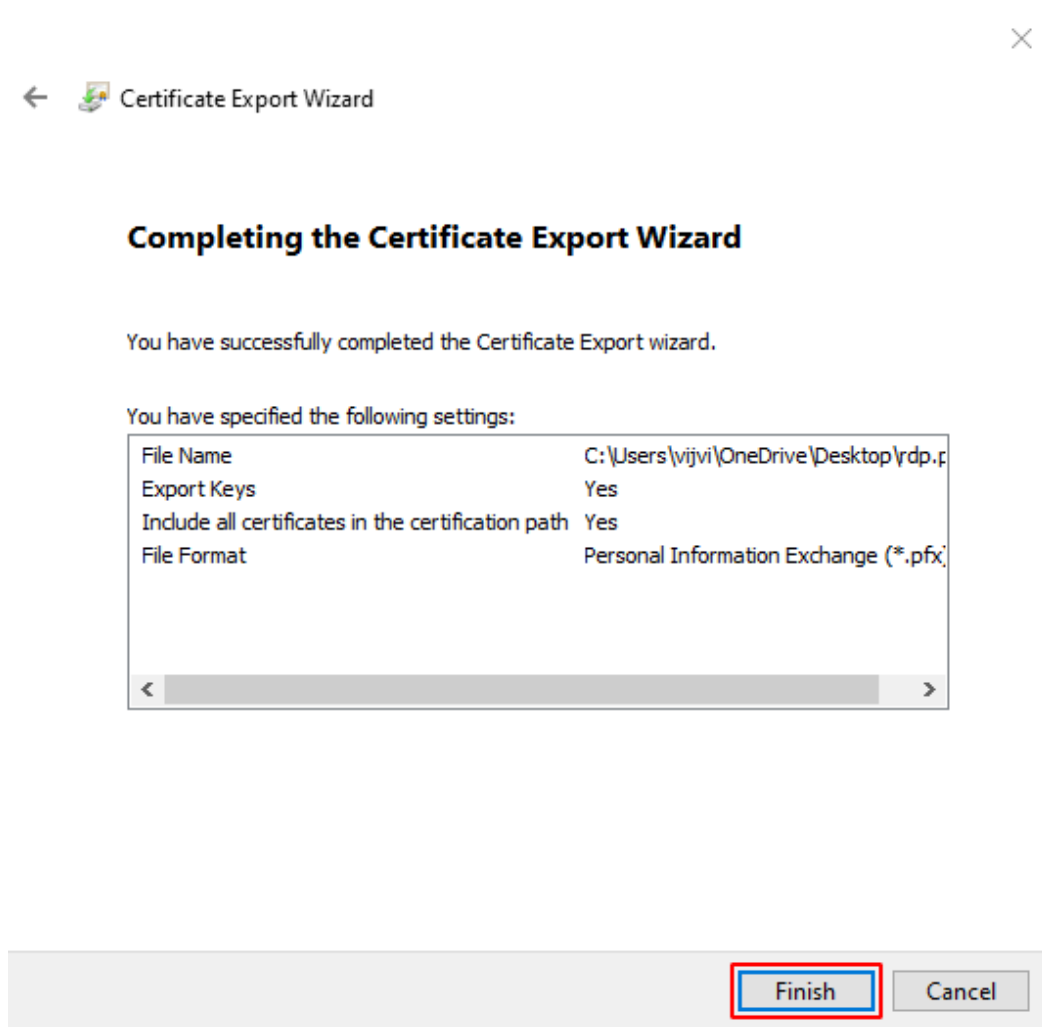
C:\Users\vijvi\OneDrive\Desktop\rdp.pfx



Browse...

Next

Cancel



Finally, we have successfully Exported the Certificate with the Private key.

Since our certificate was obtained through Mimikatz or Jailbreak, we transferred it to a **Linux host** and extracted the key using OpenSSL. To extract the key in PEM format, we first used the OpenSSL command

**You can download the certificate from [here](#)**

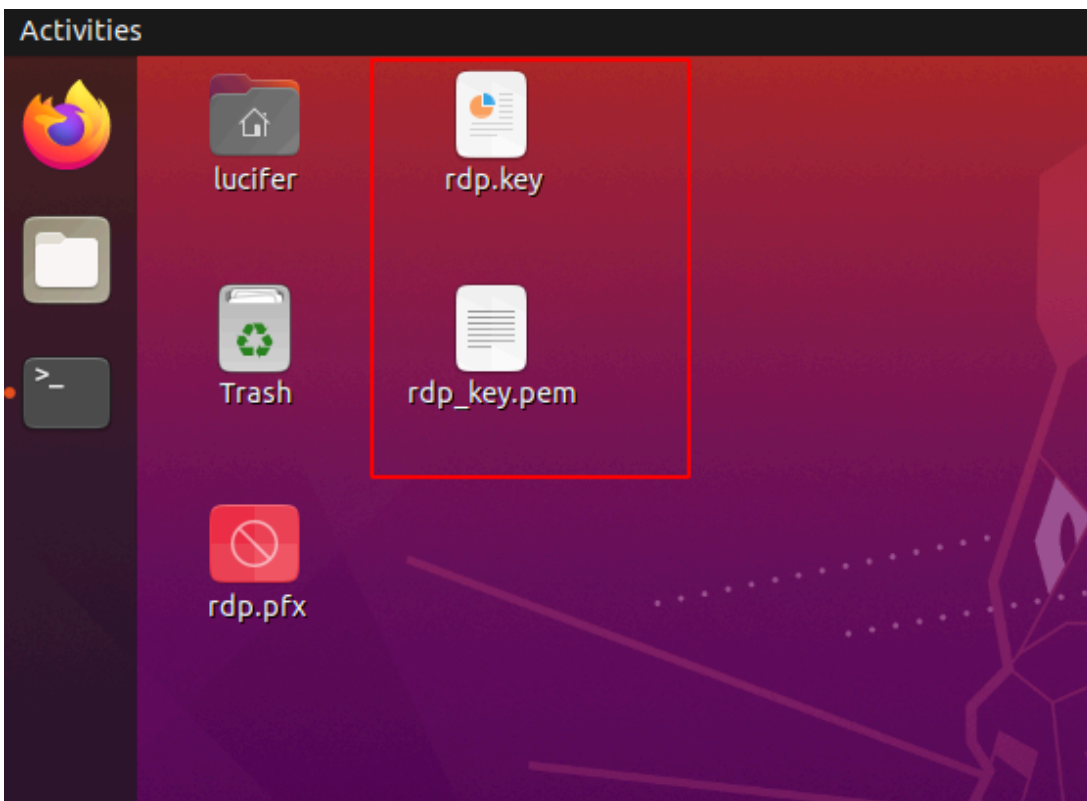
```
openssl pkcs12 -in rdp.pfx -nocerts -out rdp_key.pem -nodes
```

Password: 123

Then after we have to remove the passphrase from the key, to do this run the following command

```
root@ignite: /home/lucifer/Desktop
root@ignite:/home/lucifer/Desktop# openssl pkcs12 -in rdp.pfx -nocerts -out rdp_key.pem -nodes
Enter Import Password:
root@ignite:/home/lucifer/Desktop# openssl rsa -in rdp_key.pem -out rdp.key
writing RSA key
root@ignite:/home/lucifer/Desktop#
```

openssl rsa -in rdp\_key.pem -out rdp.key



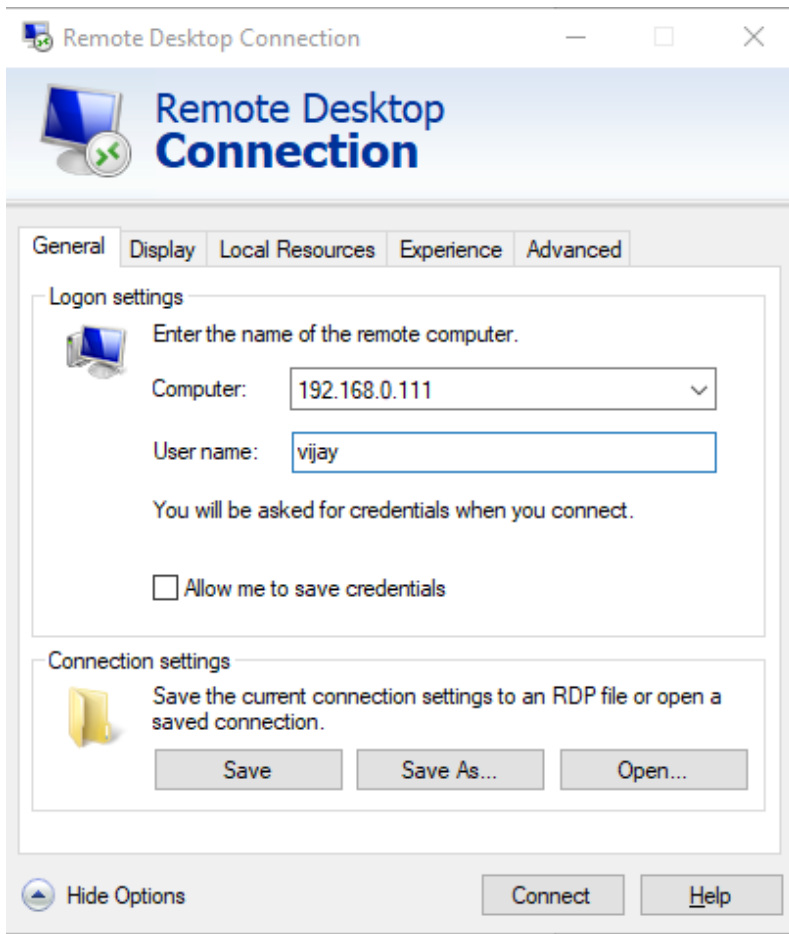
This will provide us with the RDP Server's key

As you can see, we have successfully extracted the RDP Server's private key. Don't forget to export this public key to the windows system for the use of later purposes.

Reference: [paloaltonetworks.com](http://paloaltonetworks.com)

## Capture RDP Traffic

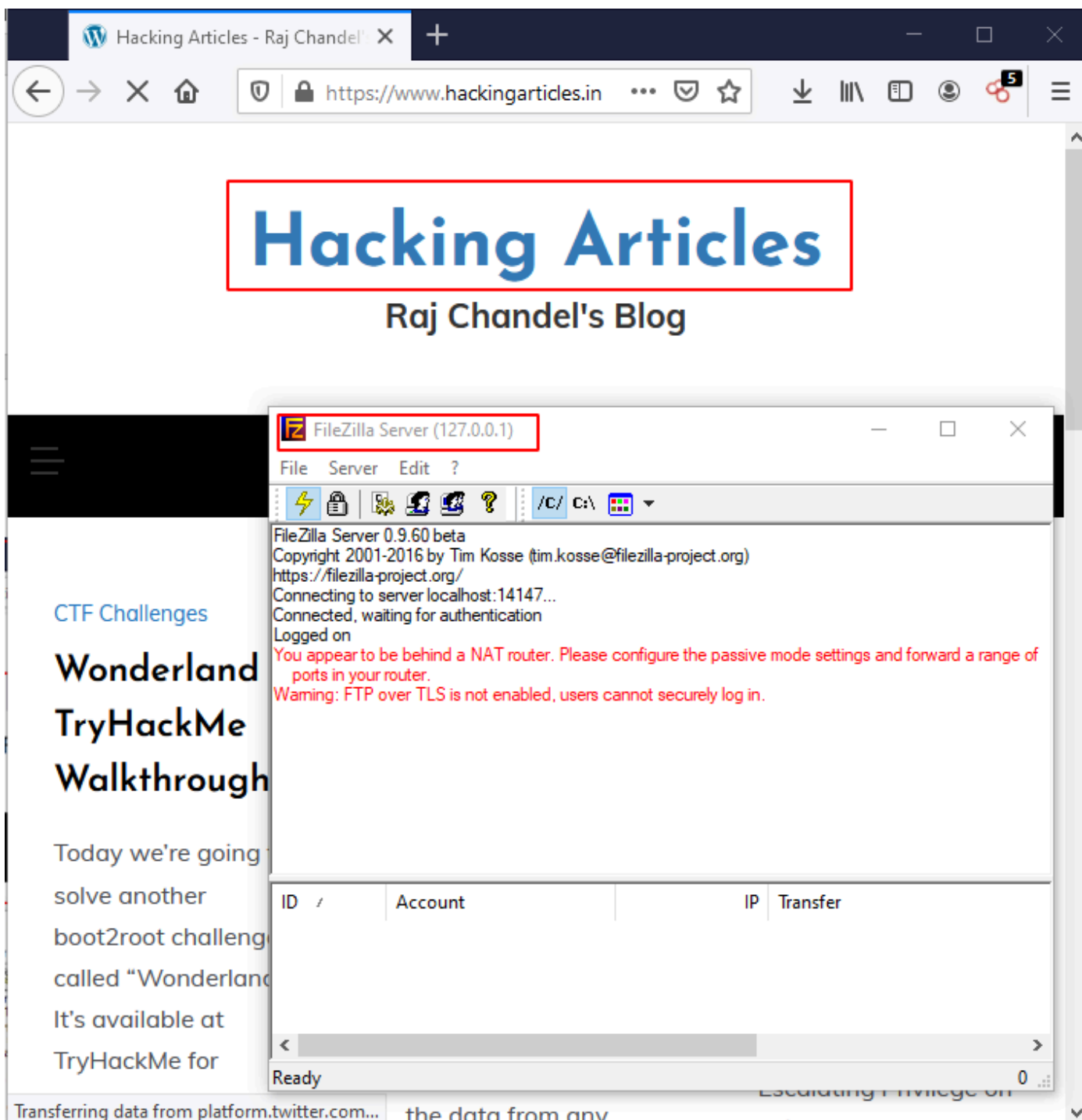
We may use a tool like Wireshark to record network traffic in the VLAN using promiscuous mode with our two Windows hosts in the same virtual network. once the recording starts Our Windows



client uses RDP to log in to the other Windows host that was operating as an RDP server. The server's host IP was 192.168.0.111.

We logged into 192.168.0.111 and performed some simple tasks including web surfing and trying to connect to an FTP server while the pcap is recording the traffic.





After of few minutes we logged out from the RDP Server and stopped recording Network traffic from Wireshark

rdpssl.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
46	4.472163	fe80::7055:f6...	fe80::7999:a3e...	UDP	86	63556 → 59466 Len=24
47	4.493246	fe80::7999:a3...	fe80::7055:f6e...	UDP	86	59466 → 63556 Len=24
48	4.493269	fe80::7999:a3...	fe80::7055:f6e...	UDP	86	59466 → 63556 Len=24
49	4.532459	fe80::7055:f6...	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
50	4.572397	fe80::7055:f6...	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
51	4.574264	fe80::7055:f6...	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
52	4.574264	fe80::7055:f6...	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
53	4.574821	192.168.0.111	224.0.0.251	MDNS	81	Standard query 0x0000 ANY DESKTOP-3KB362S.local, "QM" que
54	4.576040	192.168.0.111	224.0.0.251	MDNS	119	Standard query response 0x0000 AAAA fe80::7055:f6ed:a8e3:
55	4.577849	192.168.0.111	224.0.0.252	LLMNR	75	Standard query 0x820f ANY DESKTOP-3KB362S
56	4.805104	fe80::7055:f6...	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
57	5.005007	fe80::7055:f6...	fe80::7999:a3e...	UDP	78	63556 → 59466 Len=16
58	5.274596	fe80::7999:a3...	fe80::7055:f6e...	UDP	78	59466 → 63556 Len=16
59	5.274639	fe80::7999:a3...	fe80::7055:f6e...	UDP	78	59466 → 63556 Len=16
60	5.304405	fe80::7055:f6...	fe80::7999:a3e...	ICMPv6	86	Neighbor Solicitation for fe80::7999:a3e:7ddd:8f05 from 3
61	5.304460	fe80::7999:a3...	fe80::7055:f6e...	ICMPv6	86	Neighbor Advertisement fe80::7999:a3e:7ddd:8f05 (sol, ovr
62	5.304476	fe80::7999:a3...	fe80::7055:f6e...	ICMPv6	86	Neighbor Advertisement fe80::7999:a3e:7ddd:8f05 (sol, ovr

< Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface \Device\NPF\_{C9254A2F-E329-4054-A2F3-1522492B}

> Ethernet II, Src: HonHaiPr\_22:64:23 (34:68:95:22:64:23), Dst: IntelCor\_d6:94:0d (34:02:86:d6:94:0d)

> Internet Protocol Version 6, Src: fe80::7055:f6ed:a8e3:5014, Dst: fe80::7999:a3e:7ddd:8f05

> User Datagram Protocol, Src Port: 63556, Dst Port: 59466

> Data (16 bytes)

<

0000 34 02 86 d6 94 0d 34 68 95 22 64 23 86 dd 60 0b 4 .....4h . "d#...`

0010 3e 63 00 18 11 80 fe 80 00 00 00 00 00 00 70 55 >c.....pU

0020 f6 ed a8 e3 50 14 fe 80 00 00 00 00 00 00 79 99 ...P...y.

0030 0a 3e 7d dd 8f 05 f8 44 e8 4a 00 18 03 b4 00 00 ->}...D .J.....

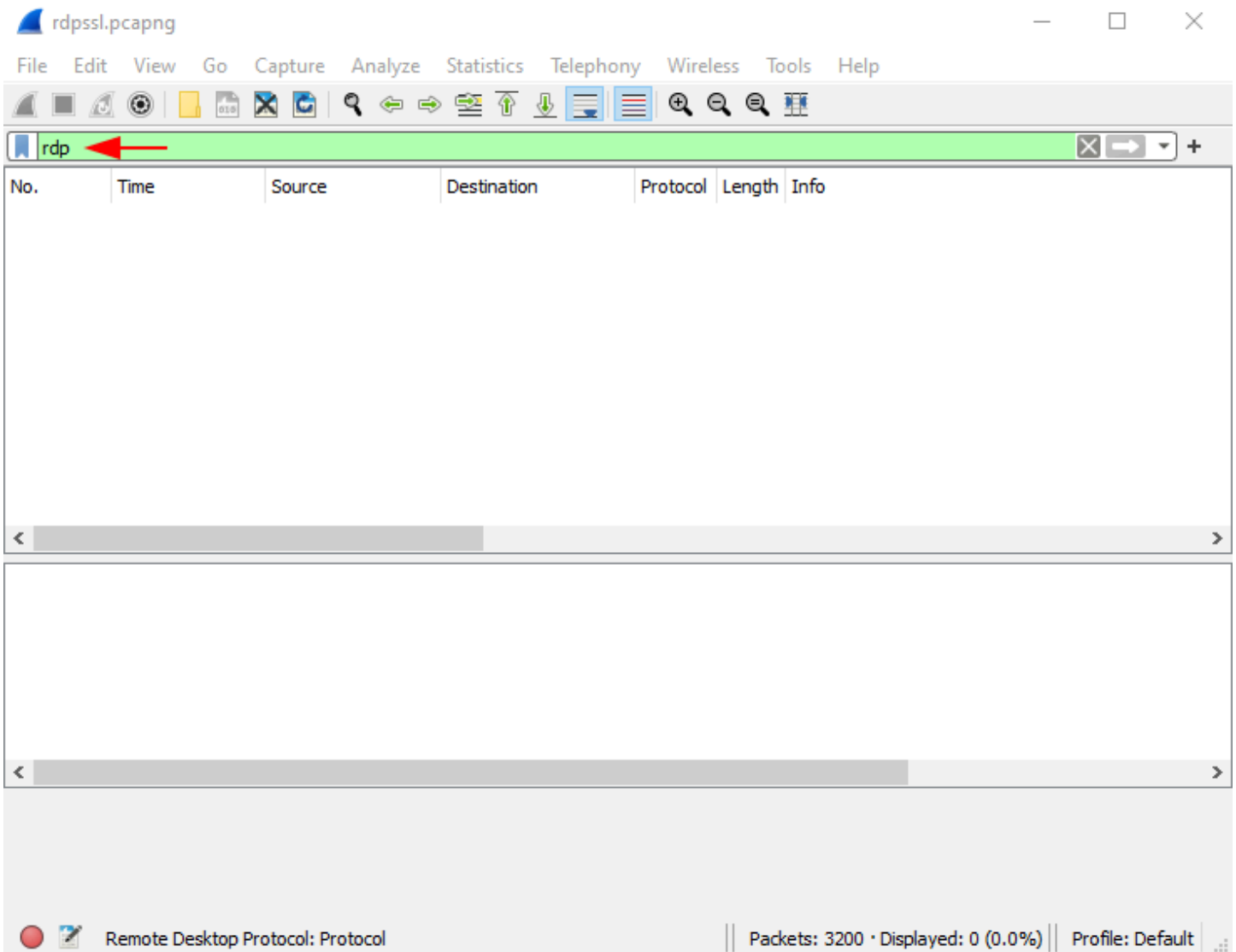
0040 00 00 ba 46 00 00 4b b9 03 17 24 6c 00 00 ...F..K. ..\$1..

rdpssl.pcapng | Packets: 3200 · Displayed: 3200 (100.0%) | Profile: Default

## Analyzing and Decrypting Wireshark Traffic

You can download the trace file from [here](#)

Open the pcap of RDP Session in Wireshark. We did not see any results when filtering RDP on our Wireshark display filter, as RDP traffic has been encrypted. We saw a blank column display during



filtering the RDP in our pcap as shown below.

## Let's Decrypt the Traffic

However, the results were quite different when we used our private server key for decrypting RDP Traffic in Wireshark.

The RDP server **DESKTOP-CDE7HJC** was at IP address **192.168.0.111** in the pcaps we captured, and RDP traffic was carried out over TCP port **3389**. This information was required in order for Wireshark to properly decrypt RDP traffic.

To do this navigate to **Edit > Preferences**, Expand the Protocols section and select TLS

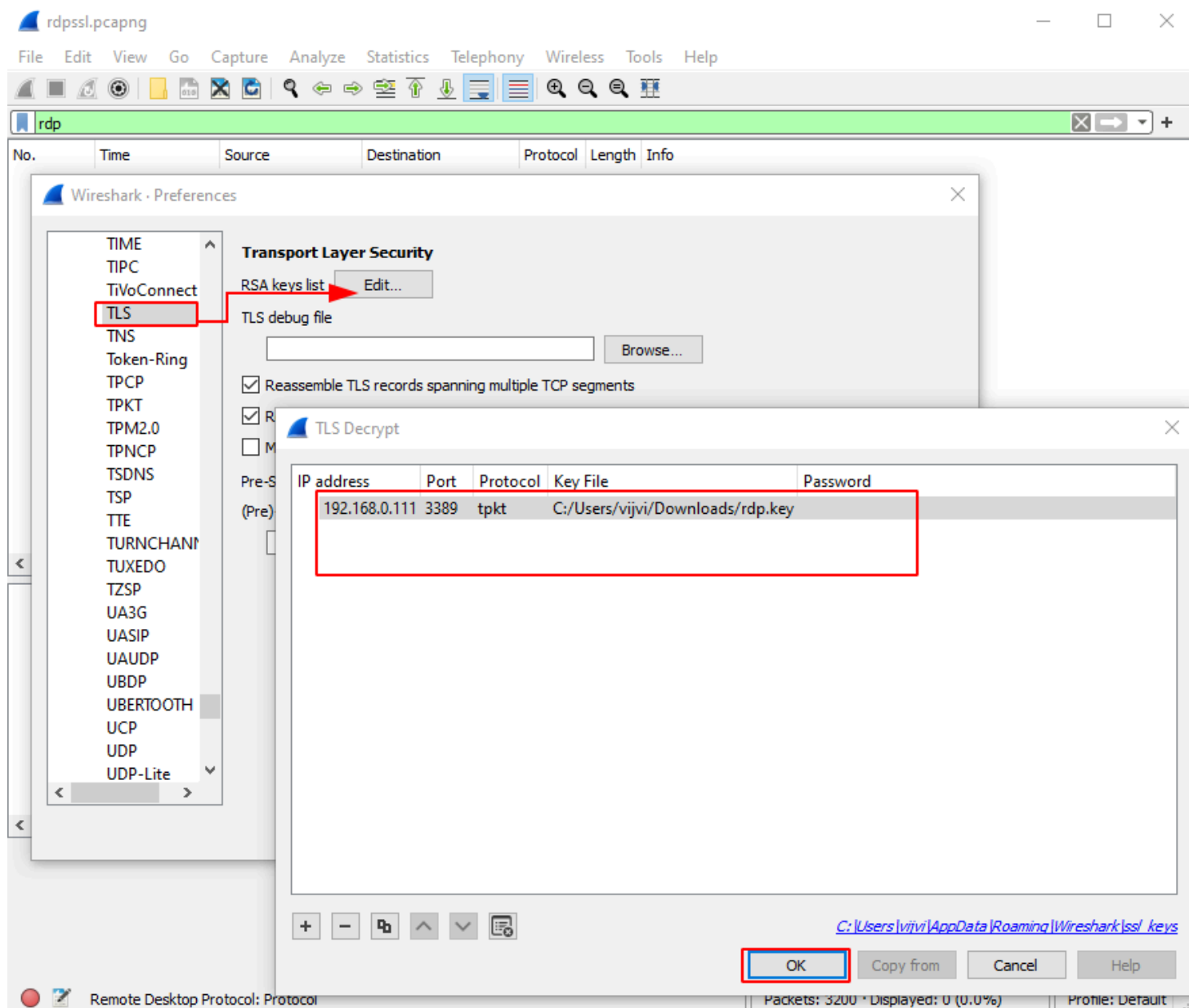
After selecting TLS navigate to the Edit section and provide the following details such as IP address of the RDP server, port no., and the path of the private key

Ip address: – 192.168.0.111

Port no.: – 3389

Protocol: – tpkt

Rsa key:- the path of the saved public key.



We had much better results when reviewing the pcap after Wireshark was configured to decrypt RDP traffic.

## Forensics with RDP Data

When we loaded our key, our column display was no longer blank when we filtered for RDP. As shown in the image below, we had a variety of outcomes.

The image shows a Wireshark capture of RDP traffic. The packet list pane highlights a series of RDP PDUs from 680 to 1940. The packet details pane shows the structure of the selected packet (680), including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, Transport Layer Security, TPKT, ISO 8073/X.224 COTP Connection-Oriented Transport Protocol, MULTIPOINT-COMMUNICATION-SERVICE T.125, GENERIC-CONFERENCE-CONTROL T.124, and Remote Desktop Protocol. The packet bytes pane shows the raw data of the frame and the decrypted TLS data.

No.	Time	Source	Destination	Protocol	Length	Info
680	43.399467	192.168.0.108	192.168.0.111	RDP	545	ClientData
687	43.674095	192.168.0.111	192.168.0.108	RDP	209	ServerData Encryption: None (None)
724	45.067005	192.168.0.108	192.168.0.111	RDP	704	ClientInfo
727	45.203956	192.168.0.111	192.168.0.108	RDP	117	Error Alert
1912	83.240227	192.168.0.111	192.168.0.108	RDP	555	Demand Active PDU
1917	83.454360	192.168.0.108	192.168.0.111	RDP	762	Confirm Active PDU
1919	83.454664	192.168.0.108	192.168.0.111	RDP	119	RDP PDU Type: Synchronize
1921	83.454963	192.168.0.108	192.168.0.111	RDP	123	RDP PDU Type: Control, Action: Cooperate
1923	83.455258	192.168.0.108	192.168.0.111	RDP	123	RDP PDU Type: Control, Action: Request control
1925	83.457619	192.168.0.108	192.168.0.111	RDP	139	RDP PDU Type: BitmapCache Persistent List
1927	83.457995	192.168.0.108	192.168.0.111	RDP	123	RDP PDU Type: FontList
1930	83.458334	192.168.0.111	192.168.0.108	RDP	119	RDP PDU Type: Synchronize
1931	83.458334	192.168.0.111	192.168.0.108	RDP	123	RDP PDU Type: Control, Action: Cooperate
1934	83.459123	192.168.0.111	192.168.0.108	RDP	123	RDP PDU Type: Control, Action: Granted control
1936	83.459433	192.168.0.111	192.168.0.108	RDP	123	RDP PDU Type: FontMap
1938	83.459716	192.168.0.111	192.168.0.108	RDP	117	Virtual Channel PDU
1940	83.460347	192.168.0.111	192.168.0.108	RDP	142	Virtual Channel PDU

Frame 680: 545 bytes on wire (4360 bits), 545 bytes captured (4360 bits) on interface \Device\NPF\_{C9254A2F-E329-4054-A...}

Ethernet II, Src: IntelCor\_d6:94:0d (34:02:86:d6:94:0d), Dst: HonHaiPr\_22:64:23 (34:68:95:22:64:23)

Internet Protocol Version 4, Src: 192.168.0.108, Dst: 192.168.0.111

Transmission Control Protocol, Src Port: 49796, Dst Port: 3389, Seq: 1361, Ack: 1329, Len: 491

Transport Layer Security

TPKT, Version: 3, Length: 462

ISO 8073/X.224 COTP Connection-Oriented Transport Protocol

MULTIPOINT-COMMUNICATION-SERVICE T.125

GENERIC-CONFERENCE-CONTROL T.124

Remote Desktop Protocol

0000 34 68 95 22 64 23 34 02 86 d6 94 0d 08 00 45 00 4h "d#4. ....E.

0010 02 13 66 46 40 00 80 06 10 73 c0 a8 00 6c c0 a8 ..fF@... .s...l..

0020 00 6f c2 84 0d 3d 2d cf c3 61 a0 16 23 c8 50 18 .o...=-. .a...#P.

0030 03 fd 6c 02 00 00 17 03 03 01 e6 00 00 00 00 00 ..l.....

0040 00 00 04 c9 af 5d c4 bc af 21 8e ae fc 8d 3a 69 .....].. .!....:i

Frame (545 bytes)    Decrypted TLS (462 bytes)

rdpssl.pcapng    Packets: 3200 · Displayed: 23 (0.7%)    Profile: Default

Let's do some Forensics with this data like what we can find with this data.

Hmm!!! Exited....

As we can see that we have now successfully decrypted the RDP data... but now have more question about what to do with this kind of data ....

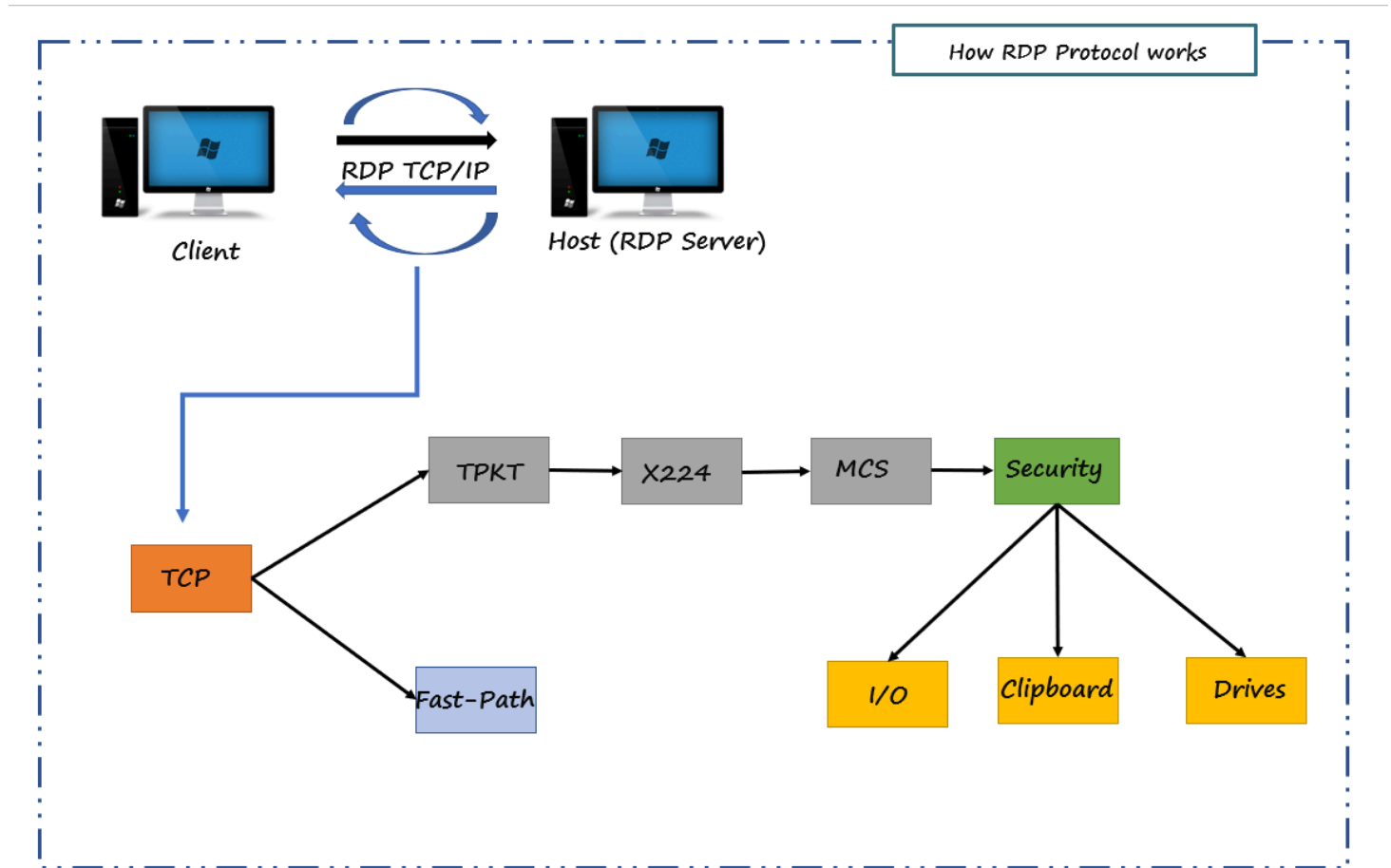
The answer is quite simple... we can gather some pieces of information like User ID, Password, Time of login, country of origin, etc... by arranging the binaries.

## The RDP Protocol

Understanding all of the underlying protocols used in RDP was one of the most difficult aspects of creating this library. It was difficult for some of them just to figure out what they were here for. The connection sequence alone employs four protocols: TPKT, X224, MCS (T.125), and GCC (T.124), as well as TLS at times and RC4 at others.

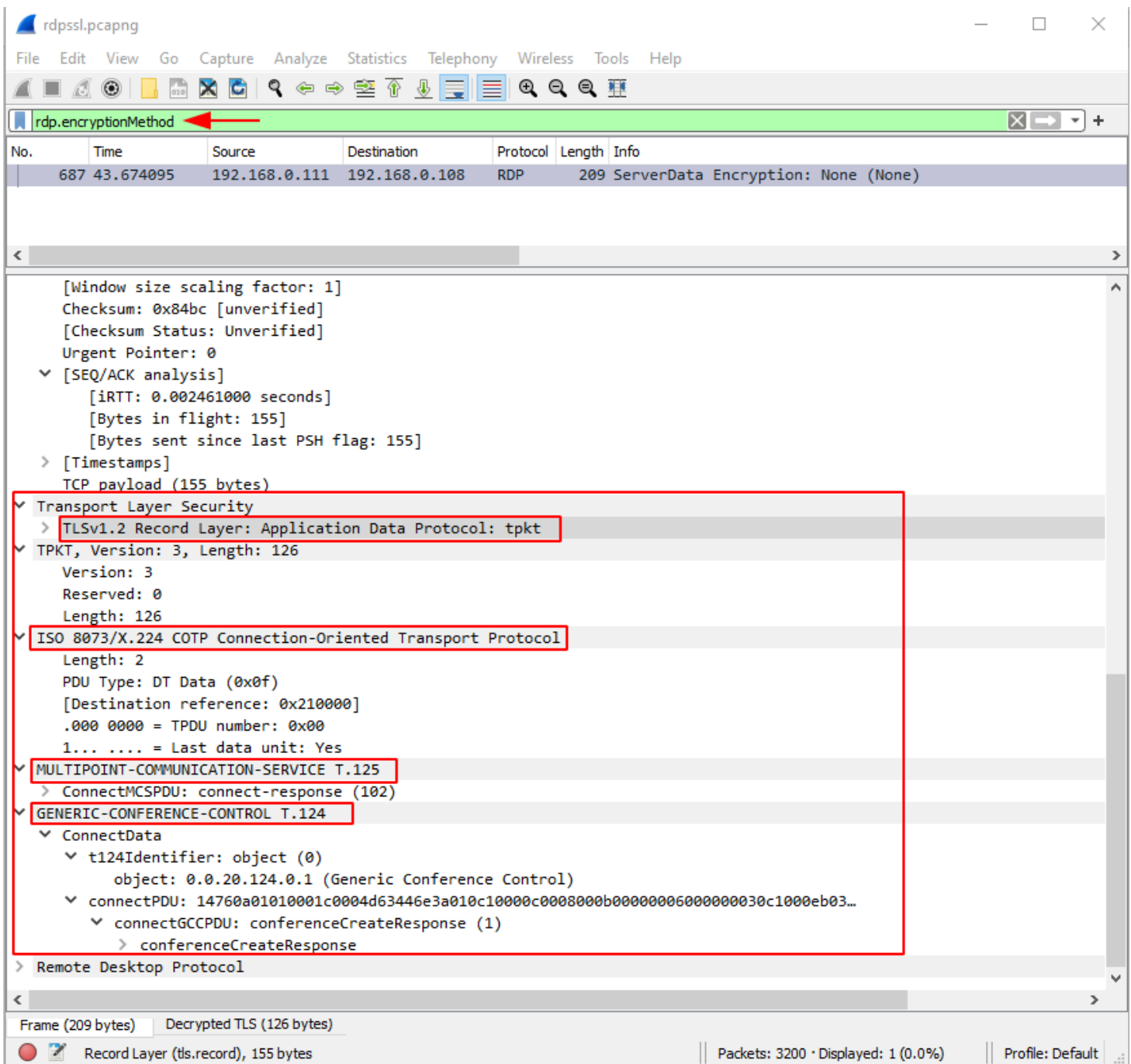
This implies that we must be able to initiate TLS after the first few packets of the connection if necessary so we must keep an eye out for that.

Are you confused? We were, too. As a result, we created this simple diagram to explain the connection sequence.



## Security Protocols used by RDP Server

Let's first understand which type of Security used by the RDP server



As shown in the figure of RDP protocol now we're much clear that the RDP server uses this type of protocol to secure the connection.

## Analyze connectivity Duration between Client & Host

Using the packet capture file, we were able to determine the time of the incident. We used a very cool Wireshark feature that's hidden in **Statistics -> Conversations -> TCP**: Conversations between two endpoints.

Wireshark · Conversations · rdpsl.pcapng

Ethernet · 15IPv4 · 30IPv6 · 5TCP · 31UDP · 22

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A	
192.168.0.103	58377	23.35.33.212	443	30	9229	18	2224	12	7005	69.397183	0.0863	206k	649k	
192.168.0.103	58378	20.190.175.23	443	39	30k	22	12k	17	17k	117.486753	0.4860	206k	290k	
192.168.0.103	58379	40.74.108.123	443	38	11k	26	5160	12	6434	118.158809	1.5818	26k	32k	
192.168.0.103	58380	34.213.55.174	443	5	306	4	240	1	66	120.258585	0.2804	6847	1883	
192.168.0.108	49786	40.119.211.203	443	13	1002	10	828	3	174	8.549800	0.2238	29k	6218	
192.168.0.108	49787	192.168.0.111	3389	33	5519	22	3662	11	1857	9.383636	22.9882	1274	646	
192.168.0.108	49782	117.18.237.29	80	18	972	12	648	6	324	18.728141	90.6012	57	28	
192.168.0.108	49781	117.18.232.200	443	6	324	4	216	2	108	20.512174	61.4542	28	14	
192.168.0.108	49788	40.74.108.123	443	8	804	6	684	2	120	20.926347	0.2799	19k	3430	
192.168.0.108	49789	40.74.108.123	443	8	704	6	584	2	120	21.207744	0.2817	16k	3407	
192.168.0.108	49790	40.74.108.123	443	8	572	6	452	2	120	21.490530	0.2930	12k	3276	
192.168.0.108	49783	204.79.197.254	443	1	54	0	0	1	54	22.769468	0.0000	—	—	
192.168.0.108	49785	204.79.197.222	443	1	54	0	0	1	54	27.156114	0.0000	—	—	
192.168.0.108	49791	40.74.108.123	443	9	858	6	684	3	174	35.239992	0.2744	19k	5072	
192.168.0.108	49792	40.74.108.123	443	8	704	6	584	2	120	35.515121	0.2752	16k	3488	
192.168.0.108	49793	40.74.108.123	443	8	572	6	452	2	120	35.791824	0.2746	13k	3495	
192.168.0.108	49794	40.119.211.203	443	13	1002	10	828	3	174	38.596226	0.2208	29k	6303	
192.168.0.108	49795	192.168.0.111	3389	23	5071	16	3386	7	1685	41.008906	0.1289	210k	104k	
192.168.0.108	49796	192.168.0.111	3389	500	51k	372	41k	128	10k	43.267280	64.8929	5142	1263	
192.168.0.108	49797	52.139.250.253	443	13	1002	10	828	3	174	68.651401	0.2262	29k	6153	
192.168.0.108	49798	52.242.101.226	443	8	794	6	674	2	120	124.110820	0.4432	12k	2165	
192.168.0.108	49799	52.242.101.226	443	8	794	6	674	2	120	124.560574	0.4475	12k	2145	
192.168.0.108	49800	52.242.101.226	443	8	794	6	674	2	120	125.022160	0.4430	12k	2167	
192.168.0.108	49801	52.242.101.226	443	8	794	6	674	2	120	125.469183	0.4459	12k	2152	

☐ Name resolution☐ Limit to display filter☐ Absolute start time

Conversation Types ▾

Copy ▾Follow Stream...Graph...CloseHelp

As we can see there is something that happened In the highlighted fields. I managed to understand they are using RDP protocol (port no.3389) but won't be able to find out the actual duration of the connectivity between the client and the host so, I applied "Limit to the Display filter"

Wireshark · Conversations · rdpsl.pcapng

Ethernet · 1IPv4 · 1IPv6TCP · 1UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.0.108	49796	192.168.0.111	3389	23	5002	12	3084	11	1918	43.399467	40.4877	609	378

☐ Name resolution☒ Limit to display filter☐ Absolute start time

Conversation Types ▾

Copy ▾Follow Stream...Graph...CloseHelp

Now, the scenario is pretty much clear and we managed to see the actual duration of connectivity and we got two IP addresses of the client and host that is **192.168.0.108** and **192.168.0.111**. but still, we were not able to identify



which one is the host and which one is the client.

## Analyzing Credentials

So, have got the two specific IP address and port number. Without wasting I applied a quick filter on behalf of this information

```
Ip.addr == 192.168.0.111 && tcp.port==3389
```

rdpsll.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 192.168.0.111 && tcp.port==3389

No.	Time	Source	Destination	Protocol	Length	Info
151	9.556733	192.168.0.108	192.168.0.111	TLSv1...	73	Ignored Unknown Record
152	9.556750	192.168.0.108	192.168.0.111	TCP	73	[TCP Retransmission] 49787 → 3389 [PSH, ACK] Seq=1...
153	9.600268	192.168.0.111	192.168.0.108	TCP	54	3389 → 49787 [ACK] Seq=1 Ack=20 Win=63981 Len=0
171	12.240314	192.168.0.111	192.168.0.108	TLSv1...	73	Ignored Unknown Record
172	12.293436	192.168.0.108	192.168.0.111	TCP	54	49787 → 3389 [ACK] Seq=20 Ack=20 Win=262656 Len=0
173	12.293472	192.168.0.108	192.168.0.111	TCP	54	[TCP Dup ACK 172#1] 49787 → 3389 [ACK] Seq=20 Ack=20
448	31.787797	192.168.0.108	192.168.0.111	TLSv1...	186	Client Hello
449	31.787820	192.168.0.108	192.168.0.111	TCP	186	[TCP Retransmission] 49787 → 3389 [PSH, ACK] Seq=20
450	31.831274	192.168.0.111	192.168.0.108	TCP	54	3389 → 49787 [ACK] Seq=20 Ack=152 Win=63849 Len=0
451	31.867572	192.168.0.111	192.168.0.108	TLSv1...	900	Server Hello, Certificate, Server Hello Done
452	31.869446	192.168.0.108	192.168.0.111	TLSv1...	372	Client Key Exchange, Change Cipher Spec, Finished
453	31.869461	192.168.0.108	192.168.0.111	TCP	372	[TCP Retransmission] 49787 → 3389 [PSH, ACK] Seq=15
454	31.913623	192.168.0.111	192.168.0.108	TCP	54	3389 → 49787 [ACK] Seq=866 Ack=470 Win=63531 Len=0
455	31.941830	192.168.0.111	192.168.0.108	TLSv1...	105	Change Cipher Spec, Finished
456	31.994322	192.168.0.108	192.168.0.111	TCP	54	49787 → 3389 [ACK] Seq=470 Ack=917 Win=261632 Len=0
457	31.994359	192.168.0.108	192.168.0.111	TCP	54	[TCP Dup ACK 456#1] 49787 → 3389 [ACK] Seq=470 Ack=9
458	32.011716	192.168.0.108	192.168.0.111	TPKT	140	Continuation

> Frame 639: 724 bytes on wire (5792 bits), 724 bytes captured (5792 bits) on interface \Device\NPF\_{C9254A2F-E329-4054-A...}

> Ethernet II, Src: IntelCor\_d6:94:0d (34:02:86:d6:94:0d), Dst: HonHaiPr\_22:64:23 (34:68:95:22:64:23)

Internet Protocol Version 4, Src: 192.168.0.108, Dst: 192.168.0.111

- 0100 .... = Version: 4
- .... 0101 = Header Length: 20 bytes (5)
- > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 710
- Identification: 0x663c (26172)
- > Flags: 0x40, Don't fragment
- Fragment Offset: 0

0000 34 68 95 22 64 23 34 02 86 d6 94 0d 08 00 45 00 4h"d#4. ....E.

0010 02 c6 66 3c 40 00 80 06 0f ca c0 a8 00 6c c0 a8 ..f<@.. ....1..

0020 00 6f c2 83 0d 3d d2 be 1b c1 10 24 df 57 50 18 .o...=... ..\$.WP.

0030 03 fd 5b 19 00 00 17 03 03 02 99 00 00 00 00 00 ..[.....

0040 00 00 02 b7 c3 23 45 dc 74 d5 8f fd 28 1b 5d 63 .....#E. t...(.)c

0050 b6 e1 f0 f4 20 67 be 05 10 45 38 ed 89 64 92 86 ....g...E8..d..

rdpsll.pcapng | Packets: 3200 · Displayed: 556 (17.4%) | Profile: Default

And look what we have, a whole communication between the client and host. But I can't able to understand which type of information has been shared. So, I managed to apply some more filters to make it more understandable. For example, now we have the client and host IP address so applied a display filter to find data shared by the client....

Let's see what happen

```
rdp.client.address=192.168.0.108
```

rdpssl.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

rdp.client.address==192.168.0.108

No.	Time	Source	Destination	Protocol	Length	Info
724	45.067005	192.168.0.108	192.168.0.111	RDP	704	ClientInfo

Remote Desktop Protocol

- SendData
  - clientInfoPDU
    - flags: 0x0040
    - flagsHi: 0x7ffd
    - codePage: 67699721
    - optionFlags: 0x000347bb
    - cbDomain: 0
    - cbUserName: 10
    - cbPassword: 20
    - cbAlternateShell: 0
    - cbWorkingDir: 0
    - domain:
    - userName: vijvi
    - password: Vijay@9911
    - alternateShell:
    - workingDir:
    - clientAddressFamily: 0x0002
    - cbClientAddress: 28
    - clientAddress: 192.168.0.108
    - cbClientDir: 64
    - clientDir: C:\Windows\system32\mstscax.dll
    - clientTimeZone
      - Bias: 4294966966
      - StandardName: India Standard Time
      - StandardDate
        - StandardBias: 0
        - DaylightName: India Daylight Time
      - DaylightDate
        - wYear: 0
        - wMonth: Unknown (0)
        - wDayOfWeek: Sunday (0)

Frame (704 bytes) | Decrypted TLS (621 bytes) | Bitstring tvb (1 byte)

rdpssl.pcapng | Packets: 3200 · Displayed: 1 (0.0%) | Profile: Default

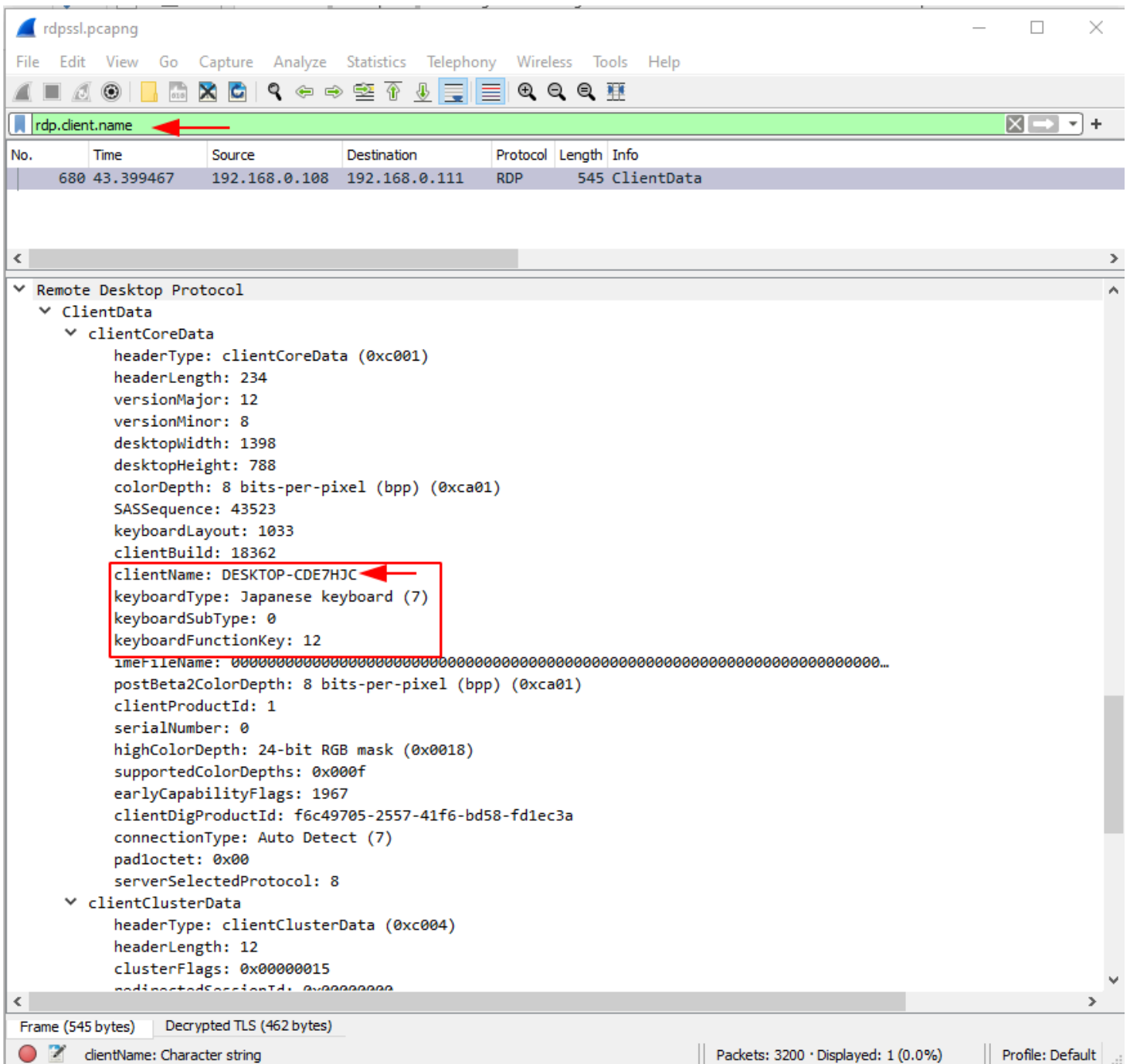
Wooh!!! As you can see we are managed to find user credentials used by clients on the RDP server just like user id, password, working directory, Time zone, weekday... etc...

As an attacker, we can use these credentials to completely take over the system.

But one more thing, still we don't know the actual user name that the client used to log in to the RDP Server. Let's find that

To find the client user name I quickly apply a display filter that shows us the client user name is entered by the client....

rdp.client.name



Whoa!!!... As you can see we have successfully got the client user name and the keyboard used by the RDP server.

By applying these kinds of filters you can easily drill down the traffic to gather the information.

## Conclusion

This article discussed how to set up an environment for decrypting RDP traffic. This is best accomplished in a virtual environment with two hosts running Windows 10 Professional. We extracted the private key from our Windows host acting as the RDP server after ensuring the client did not use any forward secrecy ciphers. Then we quickly captured a pcap of network traffic. We were able to decrypt RDP traffic after the session ended by using the server's private key.

When creating signatures to detect RDP vulnerabilities and attacks, this type of environment can be useful to security professionals.