MAY 24, 2021

Dumping RDP Credentials

Administrators typically use Remote Desktop Protocol (RDP) in order to manage Windows environments remotely. It is also typical RDP to be enabled in systems that act as a jumpstation to enable users to reach other networks. However even though this protocol is widely used most of the times it is not hardened or monitor properly.

Support pentestlab.blog

Pentestlab.blog has a long term history in the offensive security space by delivering content for over a decade. Articles discussed in pentestlab.blog have been used by cyber security professionals and red teamers for their day to day job and by students and lecturers in academia. If you have benefit by the content all these

From red teaming perspective dumping credentials from the lsass process can lead either to lateral movement across the network or directly to full domain compromise if credentials for the domain admin account have been stored. Processes which are associated with the RDP protocol can also be in the scope of red teams to harvest credentials. These processes are:

1. svchost.exe
2. mstsc.exe

The above processes can be targeted as an alternative method to retrieve credentials without touching lsass which is a heavily monitored process typically by endpoint detection and response (EDR) products.

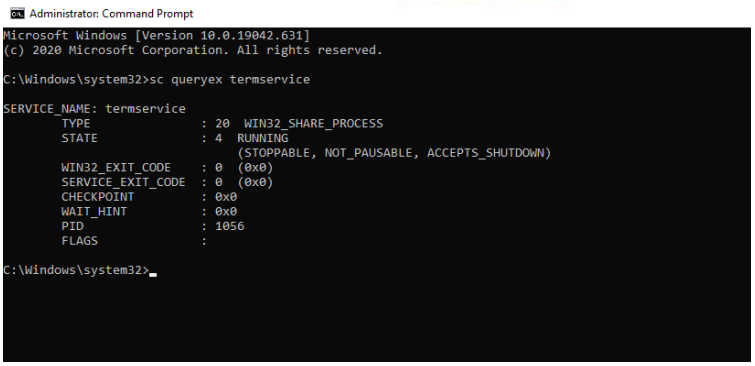
svchost

The service host (svchost.exe) is a system process which can host multiple services to prevent consumption of resources. When a user authenticates via an RDP connection the terminal service is hosted by the svchost process. Based on how the Windows authentication mechanism works the credentials are stored in memory of the svchost process in plain-text according to the

discovery of Jonas Lyk. However, looking at the process list, there are multiple svchost processes so identification of which process, hosts the terminal service connection can be achieved by executing one of the following commands.

Querying the terminal service:

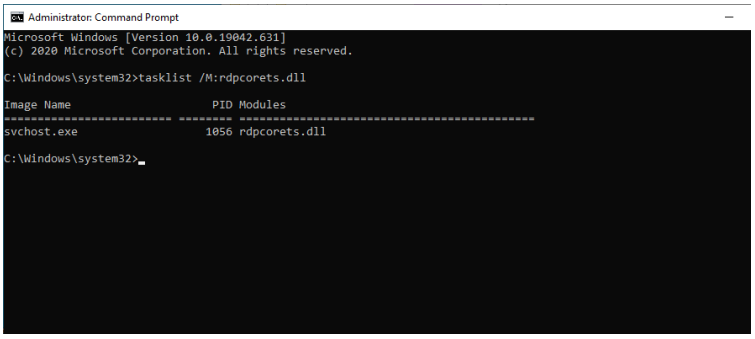
```
sc queryex termservice
```



svchost Identification – Service Query

Querying which task has loaded the rdpcorets.dll:

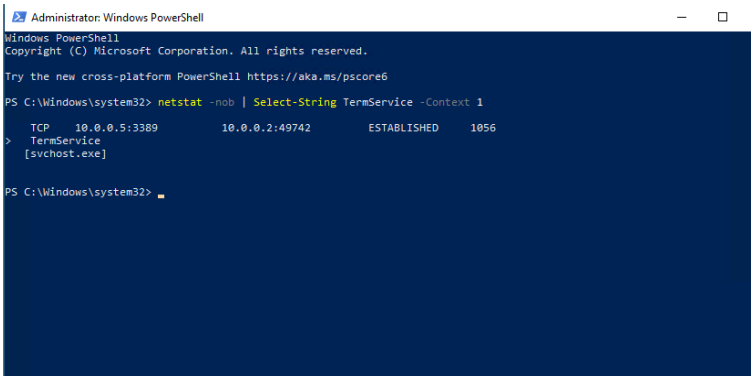
```
tasklist /M:rdpcorets.dll
```



svchost Identification – RDP Core DLL

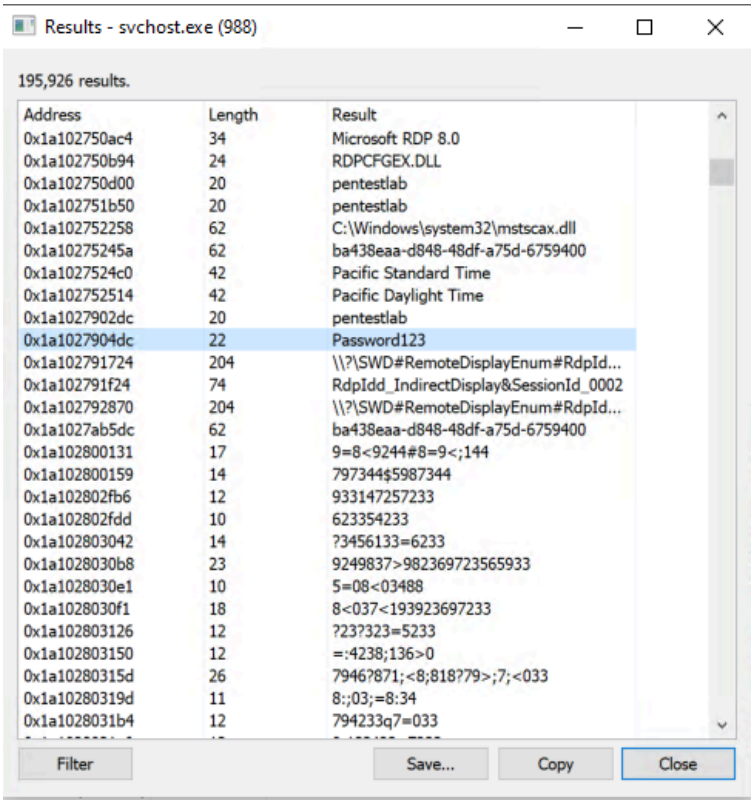
Running netstat:

```
netstat -nob | Select-String TermService -Context 1
```



svchost Identification – netstat

Looking at the memory strings of the process the password is displayed below the username.



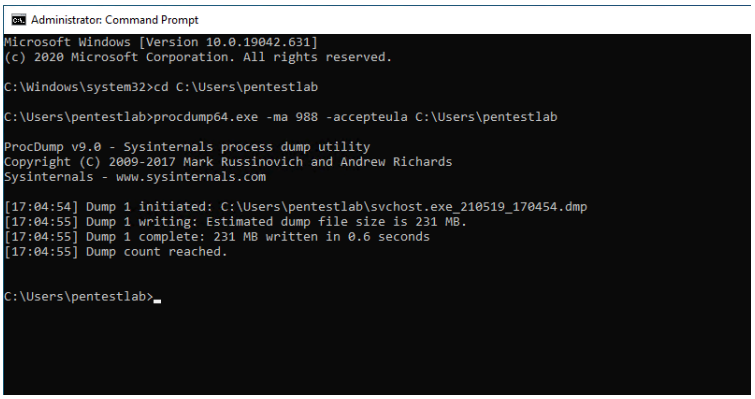
Memory Strings

Process dump from Sysinternals can be used also to dump the memory by specifying the PID

CATEGORIES

and the directory which the .dmp file will be written.

```
procdump64.exe -ma 988 -accepteula
C:\Users\pentestlab
```



Memory Dumping – Process Dump

The .dmp file can be transferred to another host for offline analysis. Performing a simple grep will identify the password stored in the memory file below the username.

```
strings -el svchost* | grep Password123 -C3
```

Discovery of Password in Memory Dump

The above method doesn't consider fully reliable and it is still unknown in which

October 2024

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20

conditions the credentials are maintained in the svchost process. However, Mimikatz support the retrieval of credentials from existing RDP connections by executing the following:

```
privilege::debug
ts::logonpasswords
```

Mimikatz – RDP Credentials

mstsc

The mstsc.exe process is created when a user opens the remote desktop connection application in order to connect to other systems via the RDP protocol. API hooking could be used to intercept the credentials provided by the user and use them for lateral movement. Rio Sherri has developed a proof of concept tool called RdpThief which attempts to hook the functions used by mstsc process (CredIsMarshaledCredentialW & CryptProtectMemory) in order to retrieve the credentials and write them into a file on the

21	22	23	24	25	26	27
28	29	30	31			

« Aug

PEN TEST LAB STATS

7,614,832 hits

FACEBOOK PAGE

Facebook Page

disk. Details of the tool can be found in an article in the MDSec website.

From a system that has been compromised and the mstsc.exe is running the DLL needs to be injected into the process.

```
SimpleInjector.exe mstsc.exe RdpThief.dll
```

RdpThief.dll – DLL Injection

Once the user enter the credentials for authentication to the destination host these will be captured and written into a file on the C:\temp folder.

CredPrompt

The file creds.txt will include also the IP address. This information could be utilized to move laterally across the network or even to escalate privileges if an elevated account is used.

The tool has been rewritten in C# by Josh Magri. However comparing to RdpThief, SharpRDPThief uses an IPC server in order to receive the credentials from the mstsc.exe process. In the event that the mstsc.exe is terminated the server will continue to run and when the process is initiated again will attempt to perform the hooking. This removes the limitation that RdpThief had that the process should already exist.

SharpRDPThief

RDP Files

Users that tend to authenticate multiple times to a particular host via an RDP connection they might save the connections details for quick authentication. These credentials are stored in an encrypted form in the Credential Manager of Windows by using the Data Protection API.

Credential Manager

The location of the Windows Credentials on the disk is the following:

```
C:\Users\username\AppData\Local\Microsoft\Credentials
```

Windows Credentials Location

The file can be viewed through the Mimikatz binary:

```
dpapi::cred
/in:C:\Users\pentestlab\AppData\Local\Microsoft\C
redentials\ACC240EEE479C1B634EC496F9838074B
```

DPAPI Credentials – Mimikatz

The "*pbData*" field contains the information in an encrypted form. However the master key for decryption is stored in the lsass and can be retrieved by executing the following Mimikatz module. The "*guidMasterKey*" is also important as multiple entries might exist when the lsass is queried and it is needed to match the GUID with the Master Key.

```
sekurlsa::dpapi
```

Mimikatz – DPAPI Master Key

Executing again the dpapi::cred module with the master key switch will have as a result the decryption of the contents and the RDP credentials to be disclosed in plain-text.

```
dpapi::cred
/in:C:\Users\pentestlab\AppData\Local\Microsoft\C
redentials\ACC240EEE479C1B634EC496F9838074B
/masterkey:05d8e693421698148d8a4692f27263201f1c65
e0b3ac08e3be91ea75f43e71e9b398e2418ba0f0c62ea70a3
17bdba88f11da3adebd07d65d2b349f933eab85e1
```

DPAPI – Decrypting Credentials

Executing the following command will provide the details in which server these credentials belong.

```
vault::list
```

Mimikatz – Vault List

YouTube

ef-extracting-clear-text-credentials-from-remote-desktop-clients/

- https://www.n00py.io/2021/05/dumping-plaintext-rdp-credentials-from-svchost-exe/
- https://github.com/0x09AL/RdpThief
- https://github.com/mantvydasb/RdpThief

- https://github.com/passthehashbrowns/SharpRDPThief
- https://www.ired.team/offensive-security/code-injection-process-injection/api-monitoring-and-hooking-for-offensive-tooling
- https://labs.f-secure.com/blog/attack-detection-fundamentals-2021-windows-lab-3/