Memory Forensics for First Responders in Security Incidents

Swiss Cyberstorm

2022-10-25

About BLS, about me

About BLS

- Second-largest railroad company in Switzerland.
- Serving some of the most beautiful destinations in the alps.
- Cool place to work

About Me

- Doing IT for 40 years, doing IT security for 20+ years
  in other words: Just an old dude
- Loves anything in hex code: Network traces, memory dumps, disk images …

About Memory Forensics

- A technique to analyze a computers RAM to identify artefacts of a suspected or actual hacking attack
- Goals of the analysis:
  - Identify Malware injected into legitimate processes
  - Extract important information like C&C servers from the dump

Getting Started

Obtaining A Dump

- My favorite techniques:
  - Virtual Machine Snapshot
  - Belkasoft RamCapturer.exe

Using Volatility

- Python-Based application
- Undergoing a transition from v2 to v3
- Some functions from V2 are not yet available in V3
- Legacy systems (Win XP, Server 2003) not supported by V3

V2 special: Identify the operating system

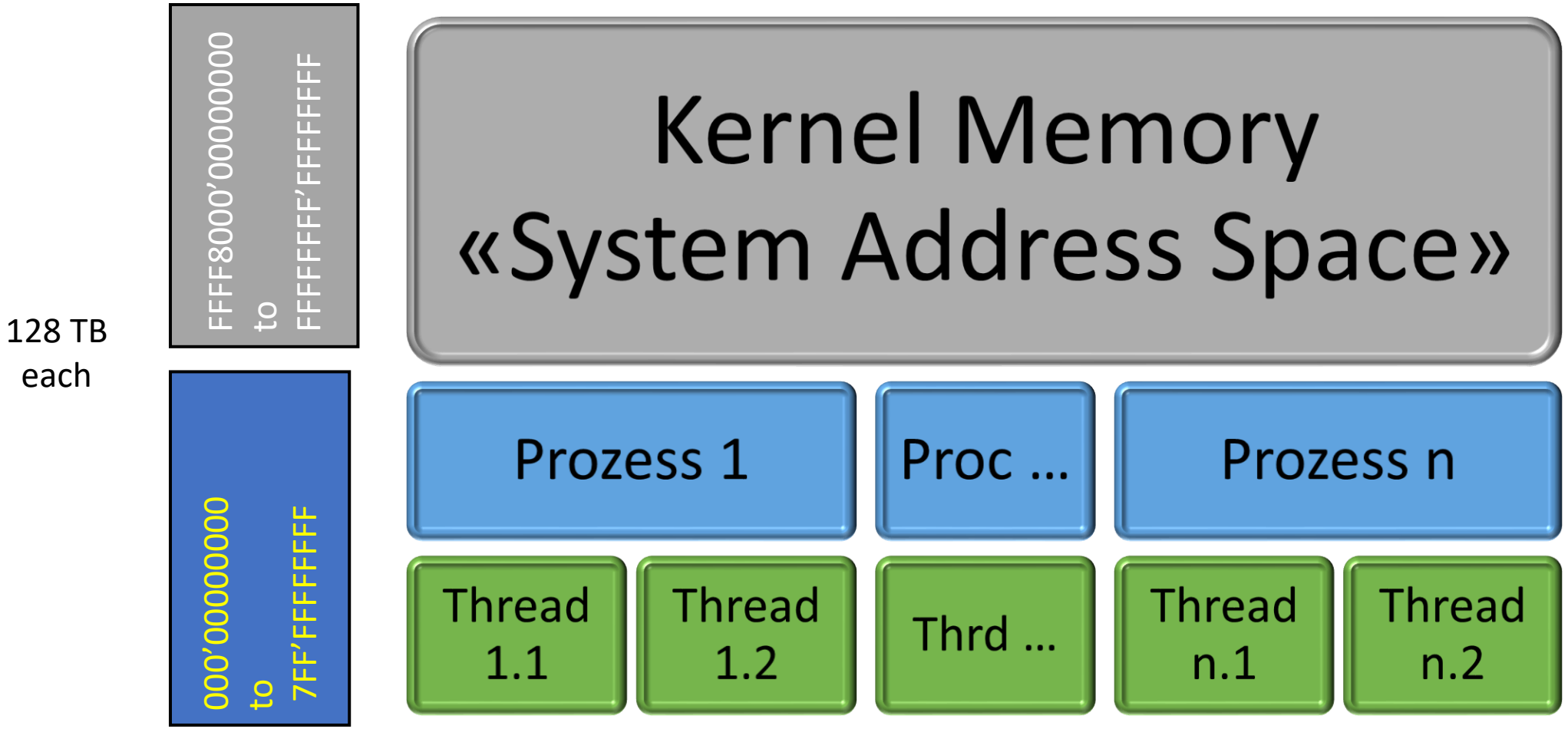
```
# vol2.py -f dumpfile.mem imageinfo
     Suggested Profile(s) : Win10x64_17134, Win10x64_14393, …
                AS Layer1 : SkipDuplicatesAMD64PagedMemory
                AS Layer2 : FileAddressSpace (dumpfile.mem)
                 PAE type : No PAE
                      DTB : 0x1ad002L
                     KDBG : 0xf802a3437520L
     Number of Processors : 4
Image Type (Service Pack) : 0
```

V3 uses Debugger Symbols, identifies OS and version automatically

Typical Investigations

Running or hidden processes
Loaded or hidden drivers
Network connections
Command Lines
API Hooks
Injected Code
…

Windows Memory Model in one Slide



Analyzing the Kernel

Malware in the Kernel

- Windows Kernel Patchguard made the life difficult for rootkits
- Scene changed with a 2018 talk at Offensivecon,
  Code to disable Patchguard was published on Github in 2019
- Malicious drivers are back in play
  - Purpose build by actors
  - Abusing existing signed drivers with known vulnerabilites
- Dedicated ASR rule in Windows blocks drivers with known vulnerabilities

Why does it matter?

- Code running in the kernel can hide attackers activity from the administrator (or first responder)
  - Files, registry keys, network connections
- Code running in the kernel can access memory of all processes
  - One place to rule them all
- Code running in the kernel has access to all devices
  - Think of encryption

Kernel investigations

- Hidden drivers
- Hooked jump tables like DDT or SSDT
- Code not related to a driver, hanging “somewhere”
- Callbacks

System Service Descriptor Table (SSDT)

Entry point to all functions provided by the kernel

- Hundreds of functions serviced by ntoskrnl.exe or win32k.sys
- File and registry operations
- Process creation, termination and manipulations

```
# vol2.py -f mydump.dmp --profile=Win10x64_17134 ssdt
SSDT[1] at fffff802a3435ea0 with 462 entries
Entry 0x1000: 0xfffff802a30cb9ec (NtAccessCheck) owned by ntoskrnl.exe
Entry 0x1001: 0xfffff802a3136fc0 (NtWorkerFactoryWorkerReady) owned by
ntoskrnl.exe
Entry 0x1002: 0xfffff802a35d830c (NtAcceptConnectPort) owned by ntoskrnl.exe
…
Entry 0x101d: 0xfffff802a359e45c (NtCreateKey) owned by AntiVirus.sys
…
```

Checking the loaded drivers

- Modlist vs. modscan
  - Note: Output shortened for readability
  - Output lists two more columsn (offset and size)

```
Name                 Base                 File
---------------- ------------------   ----
ntoskrnl.exe     0xfffff802a30a0000   \SystemRoot\system32\ntoskrnl.exe
hal.dll          0xfffff802a3014000   \SystemRoot\system32\hal.dll
kdcom.dll        0xfffff802a3a00000   \SystemRoot\system32\kdcom.dll
msrpc.sys        0xfffff800c5450000   \SystemRoot\System32\drivers\msrpc.sys
…
```

Working with Volatility

- List all kernel modules:
  `vol2.py -f mydump.dmp --profile=Win10x64_17134 modules`

- Most Volatility operations take time. I usually pipe the output into a file:
  `vol2.py -f mydump.dmp … modules > modules.txt`

- Check if modscan reveals a hidden driver:
  `grep -vFf <(awk '{ print $5 }' modules.txt) modscan.txt`

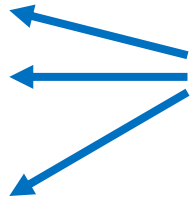
Userland Investigations

Process Investigation

- Running, hidden or terminated processes
- Injected code
- Hooks
- Unusual process rights

Case Study: The Rogue Service Host

```
Name                                          Pid   PPid
---------------------------------------- ----- -----
 0xffffc98de3f37080:wininit.exe            536    452
. 0xffffc98dea58b080:services.exe          748    536
.. 0xffffc98de4b38580:VGAuthService.      3088    748
.. 0xffffc98de4690580:svchost.exe         2052    748
...
.. 0xffffc98de5dff580:svchost.exe          976    748
... 0xffffc98de59bc580:RuntimeBroker.     6940    976
.... 0xffffc98de41cd080:WINWORD.EXE       7336   6940
..... 0xffffc98de66c9580:svchost.exe      4492   7336
...
```



Correct Process Hierarchy:
svchost spawned by services

Suspicious Process Hierarchy:
svchost spawned by winword

Analyzing the Rogue Service Host

- Dump memory content for this process
- Identify executable memory blocks, that are not mapped to a DLL
  → This is likely injected code
  → Reveals C&C servers and more, which are decrypted to use
- Search process memory for URLs, Registry Keys and other IOCs
  → Reveals information, that is decrypted at run time

- Or it is just a backdoor that uses the name of a well-known executable

What else to find

- Open Files
- Network Connections
- Registry Keys
- File System Metadata

Important Malware Artefacts

- Injected Code
- C&C Servers for Malware
- Cryptographic keys for configuration files, communication, access to the C2 server

Memory compression in Win 8 and later

- Kernel can compress process memory to conserve RAM
- User data from applications like notepad.exe is only visible after decompression

Questions?