DFIR MUSSING – EPISODE 1

Winternals for Memory Forensics

Kirtar Oza, GCFA, CISSP

02-Oct-20 [5-6 PM IST]

ABOUT ME

https://www.linkedin.com/in/kirtaroza/

Krishna @ kirtar_oza

Kirtar.oza@gmail.com

kirtar22

- 14+ years of experience in multiple cyber security domains
- Worked with Qualys, Infosys, SydneyAirport
- Global clients including UK,USA, Australia, Scandinavia
- Core Areas : DFIR, Advanced IR, Detection, Threat Hunting, ATT&CK, Memory Forensics, SIEM, Blue/Purple teaming/SOC, Threat Intel, Splunk, etc.
- GCFA,CISSP
- Many articles on eForenscis, LinkedIn, Securityaffairs.co etc.

SERIES ON WINTERNALS FOR MEMORY FORENSICS

eForensics Magazine

eForensics Magazine ONLINE COURSES MAGAZINES SHOP SUBSCRIPTION BLOG ABOUT MY PRODUCTS GOOGLE

HOME / BLOG / WINDOWS PROCESS INTERNALS: A FEW CONCEPTS TO KNOW BEFORE JUMPING ON MEMORY FORENSICS [PART 4] — JOURNEY IN TO THE UNDOCUMENTED VAD STRUCTURES (VIRTUAL ADDRESS DESCRIPTORS) | BY KIRTAR OZA

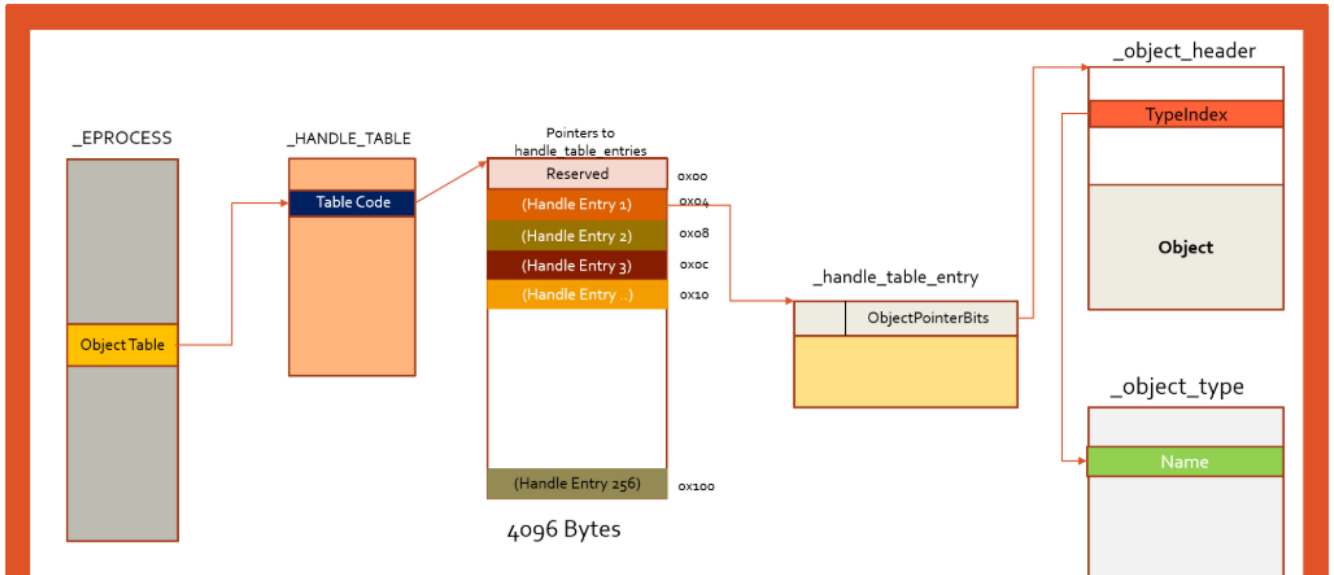
Windows Process Internals: A few Concepts to know before jumping on Memory Forensics [Part 4] — Journey in to the Undocumented VAD Structures (Virtual Address Descriptors) | By Kirtar Oza

cs [Part 2] –

HOME / BLOG / WINDOWS PROCESS INTERNALS: A FEW CONCEPTS TO KNOW BEFORE JUMPING ON MEMORY FORENSICS [PART 3] – JOURNEY IN TO THE PSLOADEDMODULELIST (LOADED KERNEL MODULES) | BY KIRTAR OZA

Windows Process Internals: A few Concepts to know before jumping on Memory Forensics [Part 5] – A Journey in to the Undocumented Process Handle Structures (_handle_table & _handle_table_entry) | By Kirtar Oza

nory Forensics [Part 3] – Oza



SEARCH

Newsletter

Signup for news and special offers!

Email*

Yes, please sign me up for newsletters. This includes offers, latest news, and exclusive promotions

Subscribe

SEARCH

Newsletter

Signup for news and special offers!

Email*

Yes, please sign me up for newsletters. This includes offers, latest news, and exclusive promotions

3 Subscribe

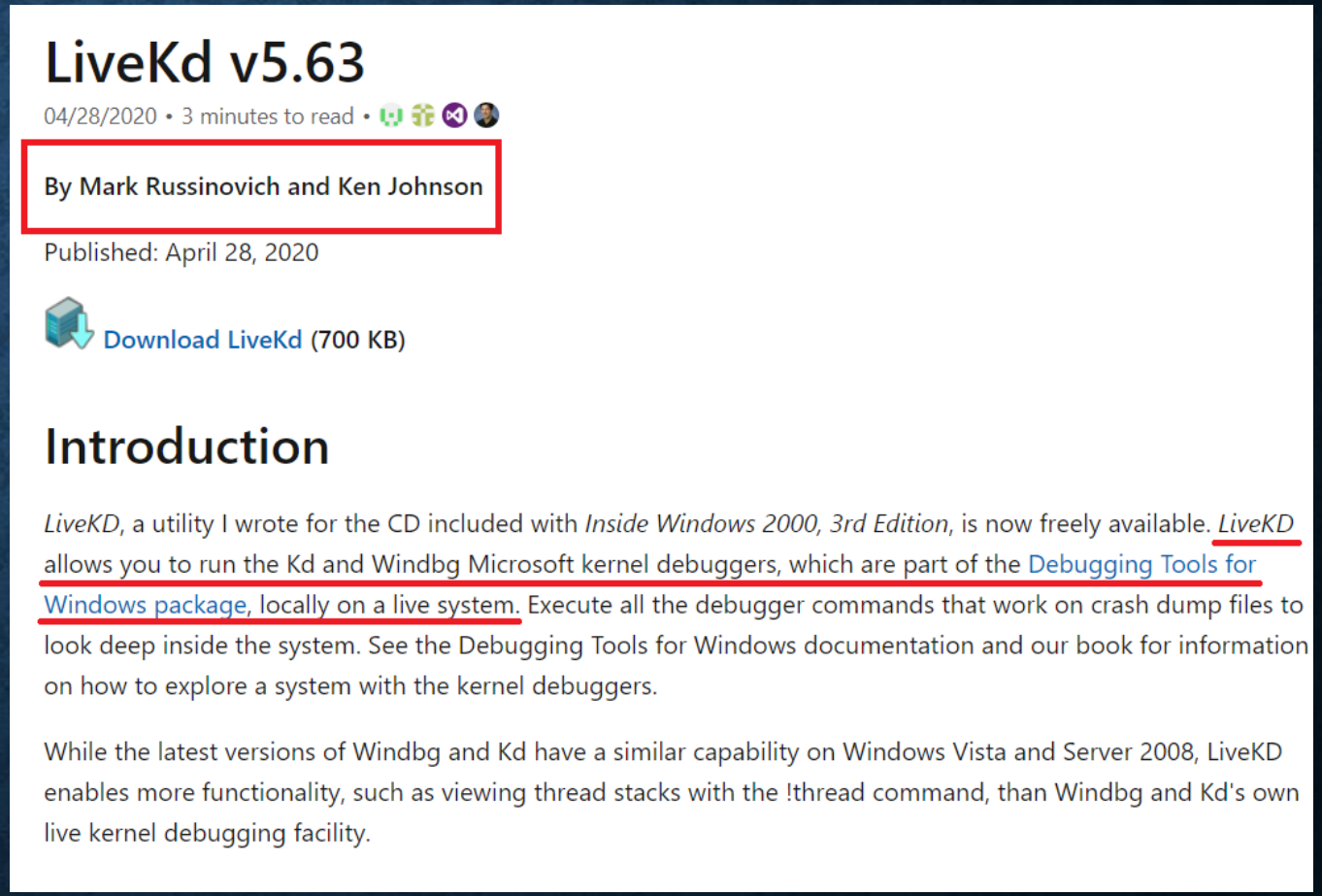
AGENDA

- Kernel Debugging - Intro to LIVEKD
- Process Internals - _EPROCESS
- Traversing "ActiveProcessLinks"

PART 1 – Intro to LiveKD

KERNEL DEBUGGING

- LiveKD (SysInternals)
- Why LiveKD
  - No need to reboot the machine in a debug mode
  - FreeTool
  - Quite easy to setup and running
- Two Modes
  - Cmdline
  - Windbg



LiveKd v5.63

04/28/2020 • 3 minutes to read •

By Mark Russinovich and Ken Johnson

Published: April 28, 2020

Download LiveKd (700 KB)

Introduction

LiveKD, a utility I wrote for the CD included with *Inside Windows 2000, 3rd Edition*, is now freely available. *LiveKD* allows you to run the Kd and Windbg Microsoft kernel debuggers, which are part of the Debugging Tools for Windows package, locally on a live system. Execute all the debugger commands that work on crash dump files to look deep inside the system. See the Debugging Tools for Windows documentation and our book for information on how to explore a system with the kernel debuggers.

While the latest versions of Windbg and Kd have a similar capability on Windows Vista and Server 2008, LiveKD enables more functionality, such as viewing thread stacks with the !thread command, than Windbg and Kd's own live kernel debugging facility.

SYMBOLS FOR WINDEBUGGER

- Symbol Files contain
  - Names of Functions & Variables
  - Layouts and Formats of Data Structures
- Used by debugger to reference and display these names during debugging
- Setting up Symbol Path for debugger

```
.sympath srv*c:\MyServerSymbols*https://msdl.microsoft.com/download/symbols
```

This command tells the debugger to use a symbol server to get symbols from the symbol store at https://msdl.microsoft.com/download/symbols and cache the symbols in c:\symbols

```
0: kd> .sympath
Symbol search path is: srv*c:\Symbols*http://msdl.microsoft.com/download/symbols
Expanded Symbol search path is: srv*c:\symbols*http://msdl.microsoft.com/download/symbols
```

DEMO LIVEKD

- Demo LiveKD

PART 2 – Traversing ***ActiveProcessLinks of*** *_EPROCESS*

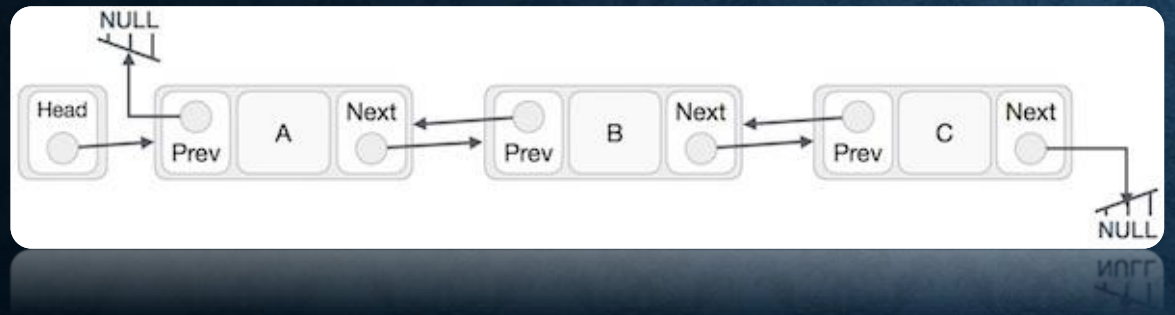
MEMORY FORENSICS & WINTRENALS

- Volatility
  - 2.6.1
  - 3.0 Beta
- pslist
- _EPROCESS

```
Memlabs volatility -f MemoryDump_Lab5.raw --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0xfffffa8000ca0040 System                    4      0     80      562 ------      0 2019-12-20 03:41:40 UTC+0000
0xfffffa80014b9040 smss.exe                248      4      3       37 ------      0 2019-12-20 03:41:40 UTC+0000
0xfffffa8000d96b30 csrss.exe               320    312     10      471      0      0 2019-12-20 03:41:45 UTC+0000
0xfffffa8001c5d060 csrss.exe               368    360      8      173      1      0 2019-12-20 03:41:47 UTC+0000
0xfffffa8001c6b060 psxss.exe               376    248     18      786      0      0 2019-12-20 03:41:47 UTC+0000
0xfffffa8001c775a0 winlogon.exe            416    360      3      108      1      0 2019-12-20 03:41:48 UTC+0000
0xfffffa8001c7c060 wininit.exe             428    312      3       75      0      0 2019-12-20 03:41:48 UTC+0000
0xfffffa8001c7bb30 services.exe            484    428      8      213      0      0 2019-12-20 03:41:50 UTC+0000
0xfffffa8001cb9880 lsass.exe               492    428      9      761      0      0 2019-12-20 03:41:50 UTC+0000
0xfffffa8001cbc4a0 lsm.exe                 500    428     10      171      0      0 2019-12-20 03:41:50 UTC+0000
0xfffffa8001ce6b30 svchost.exe             588    484     10      357      0      0 2019-12-20 03:41:54 UTC+0000
0xfffffa8001d4cb30 VBoxService.ex          656    484     13      135      0      0 2019-12-20 03:41:55 UTC+0000
0xfffffa8001d63b30 svchost.exe             724    484      8      282      0      0 2019-12-20 03:41:56 UTC+0000
0xfffffa8001da1240 svchost.exe             820    484     23      590      0      0 2019-12-20 03:41:57 UTC+0000
0xfffffa8001dba060 svchost.exe             856    484     28      535      0      0 2019-12-20 03:41:58 UTC+0000
0xfffffa8001dc2b30 svchost.exe             880    484     33      983      0      0 2019-12-20 03:41:58 UTC+0000
0xfffffa8000cf5b30 audiodg.exe             968    820      6      131      0      0 2019-12-20 03:42:00 UTC+0000
0xfffffa8001e1cb30 svchost.exe             340    484     22      504      0      0 2019-12-20 03:42:03 UTC+0000
0xfffffa8001e6c700 svchost.exe            1044    484     16      381      0      0 2019-12-20 03:42:04 UTC+0000
0xfffffa8001ee1060 spoolsv.exe            1232    484     13      283      0      0 2019-12-20 03:42:09 UTC+0000
0xfffffa8001efbb30 svchost.exe            1272    484     19      307      0      0 2019-12-20 03:42:10 UTC+0000
0xfffffa8001f775f0 svchost.exe            1372    484     22      303      0      0 2019-12-20 03:42:12 UTC+0000
0xfffffa8001f93b30 TCPSVCS.EXE            1416    484      4       97      0      0 2019-12-20 03:42:13 UTC+0000
0xfffffa8001dbf350 taskhost.exe           2012    484      8      189      1      0 2019-12-20 03:43:19 UTC+0000
```

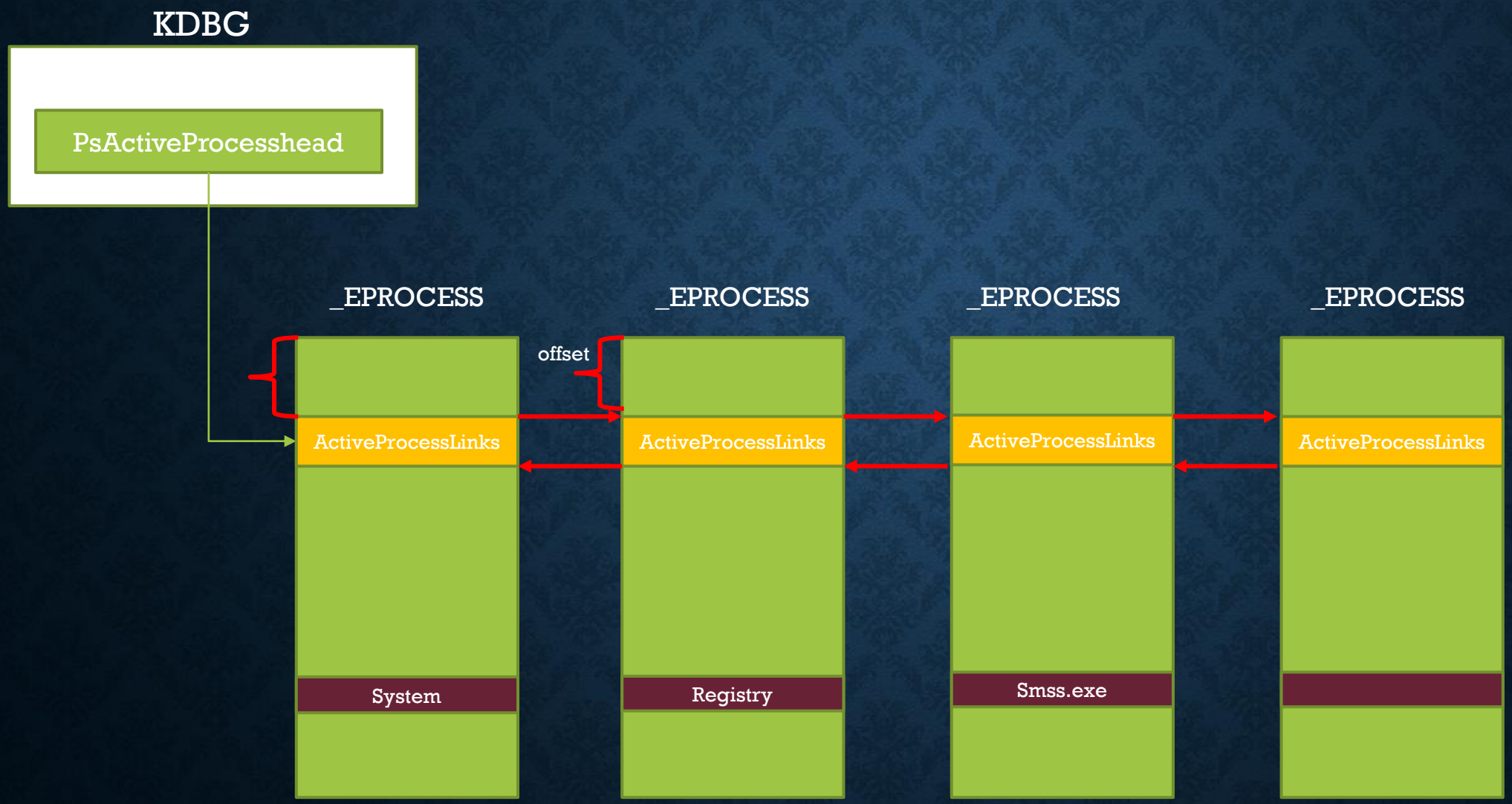
_EPROCESS

- _EPROCESS
  - PsActiveProcessLinks (doubly linked list)
    - FLINK
    - BLINK



Volatility Module	OS Data Structure
psslist	_EPROCESS ActiveProcessLinks
pstree	_EPROCESS ParentCid
Ldrmodules	_EPROCESS _PEB Ldr.InInitializationOrderModuleList Ldr.InLoadOrderModuleList Ldr.InMemoryOrderModuleList Vadroot (!vad) Vads (!vad)
dlllist	_EPROCESS _PEB
getsids	_EPROCESS _PEB Token
Hollowfind	_EPROCESS _PEB VAD (!vad)
Psxview	_EPROCESS ActiveProcessLinks

_EPROCESS


KDBG
PsActiveProcesshead
_EPROCESS
_EPROCESS
_EPROCESS
_EPROCESS
offset
ActiveProcessLinks
ActiveProcessLinks
ActiveProcessLinks
ActiveProcessLinks
System
Registry
Smss.exe

_EPROCESS

```
kd> dt nt!_EPROCESS
   +0x000 Pcb                : _KPROCESS
   +0x2e0 ProcessLock        : _EX_PUSH_LOCK
   +0x2e8 UniqueProcessId    : Ptr64 Void
   +0x2f0 ActiveProcessLinks : _LIST_ENTRY
   +0x300 RundownProtect     : _EX_RUNDOWN_REF
   +0x308 Flags2             : Uint4B
   +0x308 JobNotReallyActive : Pos 0, 1 Bit
   +0x308 AccountingFolded   : Pos 1, 1 Bit
   +0x308 NewProcessReported : Pos 2, 1 Bit
   +0x308 ExitProcessReported : Pos 3, 1 Bit
   +0x308 ReportCommitChanges : Pos 4, 1 Bit
   +0x308 LastReportMemory   : Pos 5, 1 Bit
   +0x308 ForceWakeCharge    : Pos 6, 1 Bit
   +0x308 CrossSessionCreate : Pos 7, 1 Bit
```

ActiveProcessLinks

```
   +0x3f8 Peb                : Ptr64 _PEB
   +0x400 Session            : Ptr64 _MM_SESSION_SPACE
   +0x408 Spare1             : Ptr64 Void
   +0x410 QuotaBlock         : Ptr64 _EPROCESS_QUOTA_BLOCK
   +0x418 ObjectTable        : Ptr64 _HANDLE_TABLE
   +0x420 DebugPort          : Ptr64 Void
   +0x428 WoW64Process       : Ptr64 _EWOW64PROCESS
```

Process Environment Block (PEB)

```
   +0x658 VadRoot            : _RTL_AVL_TREE
   +0x660 VadHint            : Ptr64 Void
   +0x668 VadCount           : Uint8B
   +0x670 VadPhysicalPages   : Uint8B
   +0x678 VadPhysicalPagesLimit : Uint8B
```

Virtual Address Descriptor (VAD)

TRAVERSING ACTIVEPROCESSLINKS

- Step 1 – Find out the memory address for *PsActiveProcessHead*

```
x nt!psactiveprocesshead
```

- *Step 2 – Enumerate the Pointer (_list_entry) by the PaActiveProcessHead*

```
dt nt!_list_entry <$PsActiveProcessHead>
```

- *Step 3 – Enumerate _eprocess referenced by the FLINK of _list_entry in Step 2*
  - *That will essentially pointing to the First node/Process (System Process) in the doubly linked list that is tied together by the "ActiveProcessLinks"*

```
dt nt!_eprocess <$PsActiveProcessHead.FLINK>-0x2f0 –y ImageFileName
```

Note: 0x2f0 is the offset that needs to be subtracted from the pointer to reach on the top of the _eprocess structure

- *Step 4 – Traversing through the list from Head to tail*

```
dt nt!_eprocess -l ActiveProcessLinks.Flink <$PsActiveProcessHead.FLINK>-0x2f0 -y ImageFileName
```

- Traversing through the ActievProcessLinks

DEMO

THANKS

Please route your questions/feedback to kirtar.oza@gmail.com

Connect with me On LinkedIn - https://www.linkedin.com/in/kirtaroza/