Physical Memory Analysis
Fundamentals
Prerequisites

Working knowledge of:

- WinDbg (installation, symbols)
- Basic user process dump analysis
- Basic kernel memory dump analysis
Agenda (Summary)

- Basics
- Patterns
- Exercise
- Guide
Agenda (Basics)

- Dump generation
- Memory spaces
- Major challenges
- Common commands
Platform: Windows

**Note:** The pattern-oriented approach is applicable to other OS through different memory analysis pattern implementations
Memory Analysis

Postmortem patterns

Live patterns
Dump Configuration

- Control Panel \ System \ Advanced system settings \ Startup and Recovery
- Page file size should be greater than the amount of physical memory by a few MB
- For small system partitions or virtual disk systems: DedicatedDumpFile (KB969028)
- No complete memory dumps saved (+W7) WER services blog
- Page file preservation (+W8.1) Dev Center

Troubleshooting note:

HKLM \ SYSTEM \ CurrentControlSet \ Control \ CrashControl
CrashDumpEnabled = 1 (DWORD)
Dump and Memory Acquisition

- Keyboard (KB972110, Step 6), NMI button
- Tools: NotMyFault (Step 5), SystemDump
- Killing a system process like csrss.exe (-W8.1)
- LiveKd (options for more consistency)
- Live debugging (.dump)
- Memory forensic tools
Physical Memory

Address 0

Address M
Memory Spaces

- Complete memory == Physical memory
- We always see the current virtual process space
- Kernel space is the same

To Be Discussed Later
WinDbg command to switch to a different process context:

```
.process
```
Major Challenges

- Vast memory space to search
- Multiple processes (user spaces) to examine
- User space view needs to be correct when we examine another thread
- Huge file size (x64)

To Be Discussed Later

WinDbg extension command to dump all stack traces:

!process 0 3f
Fibre Bundles

The name borrowed from mathematics (topology)

**Problem:** mild freeze of a 128GB memory system

**Solution:** dump domain specific processes and generate a kernel memory dump
Common Commands

- `.logopen <file>`
  Opens a log file to save all subsequent output

- **View commands**
  Dump everything or selected processes and threads (context changes automatically)

- **Switch commands**
  Switch to a specific process or thread for a fine-grain analysis
View Commands

- **!process 0 3f**
  Lists all processes (including times, environment, modules) and their thread stack traces

- **!process 0 1f**
  The same as the previous command but without PEB information (more secure)

- **!process <address> 3f** or **!process <address> 1f**
  The same as the previous commands but only for an individual process

- **!thread <address> 1f**
  Shows thread information and stack trace

- **!thread <address> 16**
  The same as the previous command but shows the first 3 parameters for every function
Switch Commands

- `.process /r /p <address>`
  Switches to a specified process. Its context becomes current. Reloads symbol files for user space.
  Now we can use commands like !cs

  ```
  0: kd> .process /r /p fffffa80044d8b30
  Implicit process is now fffffa80`044d8b30
  Loading User Symbols
  ...........................................
  ```

- `.thread <address>`
  Switches to a specified thread. Assumes the current process context
  Now we can use commands like k*

- `.thread /r /p <address>`
  The same as the previous command but makes the thread process context current and reloads symbol files for user space:

  ```
  0: kd> .thread /r /p fffffa80051b7060
  Implicit thread is now fffffa80`051b7060
  Implicit process is now fffffa80`044d8b30
  Loading User Symbols
  ...........................................
  ```

To Be Discussed Later

x86 stack trace from WOW64 process:

```
.thread /w
```
Agenda (Patterns)

- Pattern-driven analysis
- Pattern classification
- Pattern examples
- Common mistakes
Pattern-driven Analysis

**Pattern:** a common recurrent identifiable problem together with a set of recommendations and possible solutions to apply in a specific context.

**Note:** we do not discuss BSOD crashes here as most of the time kernel memory dumps are sufficient for analysis.
Pattern Classes

- Blocked threads
- Wait chains
- Resource consumption
- Corruption signs
- Special processes
Pattern Classification

http://www.dumpanalysis.org/memory-dump-analysis-pattern-classification

<table>
<thead>
<tr>
<th>Memory Dump Analysis Pattern Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is under development and here is a partial classification from Software Diagnostics Library pattern catalogue:</td>
</tr>
<tr>
<td>- Space/Node</td>
</tr>
<tr>
<td>- Memory dump type</td>
</tr>
<tr>
<td>- Hookware</td>
</tr>
<tr>
<td>- Wait Chain Patterns</td>
</tr>
<tr>
<td>- DLL Link Patterns</td>
</tr>
<tr>
<td>- Insufficient Memory Patterns</td>
</tr>
<tr>
<td>- Dynamic Memory Corruption Patterns</td>
</tr>
<tr>
<td>- Deadlock and Livelock Patterns</td>
</tr>
<tr>
<td>- Contention Patterns</td>
</tr>
<tr>
<td>- Stack Overflow Patterns</td>
</tr>
<tr>
<td>- .NET / CLR / Managed Space Patterns</td>
</tr>
<tr>
<td>- Stack Trace Patterns</td>
</tr>
<tr>
<td>- Symbol Patterns</td>
</tr>
<tr>
<td>- Exception Patterns</td>
</tr>
<tr>
<td>- Meta-Memory Dump Patterns</td>
</tr>
<tr>
<td>- Module Patterns</td>
</tr>
<tr>
<td>- Optimization Patterns</td>
</tr>
<tr>
<td>- Thread Patterns</td>
</tr>
<tr>
<td>- Process Patterns</td>
</tr>
<tr>
<td>- Executive Resource Patterns</td>
</tr>
</tbody>
</table>
Example: Blocked Thread

THREAD fffffa80097f1660  Cid 154c.1570  Teb: 000007ffffff4000  Win32Thread: fffff900c06a83f0  WAIT: (WrUserRequest) UserMode Non-Alertable
  fffffa800447e8f0  SynchronizationEvent
  Not impersonating
  DeviceMap  fffff8a001ce6b90
  Owning Process  fffffa8004451060  Image: ApplicationA.exe
  Attached Process  N/A  Image: N/A
  Wait Start TickCount  22248  Ticks: 47 (0:00:00:00.733)
  Context Switch Count  340  IdealProcessor: 1  LargeStack
  UserTime  00:00:00.015
  KernelTime  00:00:00.000
  Win32 Start Address ApplicationA (0x000000013f2c1210)
  Stack Init fffff8800ec25c70  Current fffff8800ec25730
  Base fffff8800ec26000  Limit fffff8800ec1d000  Call 0
  Priority 11 BasePriority 8  UnusualBoost 0  ForegroundBoost 2  IoPriority 2  PagePriority 5
  Child-SP  RetAddr  Call Site
  fffff8800ec25770  fffff80002ee6f32  nt!KiSwapContext+0x7a
  fffff8800ec25b00  fffff80002ee974f  nt!KiCommitThreadWait+0x1d2
  fffff8800ec25940  fffff9600013bc97  nt!KeWaitForSingleObject+0x19f
  fffff8800ec259e0  fffff9600013bd39  win32k!xxxRealSleepThread+0x257
  fffff8800ec25a00  fffff9600014e7a6  win32k!xxxSleepThread+0x59
  fffff8800ec25a00  fffff80002ee0ed3  win32k!NtUserWaitMessage+0x46
  fffff8800ec25ae0  000000007709933a  nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ fffff8800ec25ae0)
  00000000275f308  00000000770a4bc4  USER32!ZwUserWaitMessage+0x9a
  00000000275f310  00000000770a4edd  USER32!DialogBox+0x274
  00000000275f33a0  00000000770f2920  USER32!InternalDialogBox+0x135
  00000000275f400  00000000770f1c15  USER32!SoftModalMessageBox+0x9b4
  00000000275f530  00000000770f146b  USER32!MessageBoxWorker+0x31d
  00000000275f6f0  00000000770f1362  USER32!MessageBoxTimeoutW+0xb3
  >>> 00000000275f7c0  000000013f2c1089  USER32!MessageBoxW+0x4e
  00000000275f800  000000002135fd0  ApplicationA+0x1089
  00000000275f808  00000000000000  0x2135fd0

To Be Discussed Later

Complete Dump Analysis Exercise

© 2013 Software Diagnostics Services
Example: Wait Chain

THREAD fffffa8004514060  Cid 07f4.1470  Teb: 000007fffffae000  Win32Thread: 0000000000000000  WAIT: (UserRequest) UserMode Non-Alertable
>>> fffffa80044c53c0  Mutant - owning thread fffffa8004569750

Not impersonating
DeviceMap fffff8a001ce6b90
Owning Process fffffa8004546060  Image: ApplicationC.exe
Attached Process N/A  Image: N/A
Wait Start TickCount 14474  Ticks: 7821 (0:00:02:02.008)
Context Switch Count 2  IdealProcessor: 1
UserTime 00:00:00.000
KernelTime 00:00:00.000
Win32 Start Address ApplicationC (0x000000013f6c12a0)
Stack Init fffff8800d196c70 Current fffff8800d1967c0
Base fffff8800d197000 Limit fffff8800d191000 Call 0
Priority 11 BasePriority 8 UnusualBoost 0 ForegroundBoost 2 IoPriority 2 PagePriority 5
Child-SP RetAddr Call Site
fffff880'0d196800 fffff800'02ee6f32 nt!KiSwapContext+0x7a
fffff880'0d196940 fffff800'02ee974f nt!KiCommitThreadWait+0x1d2
fffff880'0d1969d0 fffff800'031d844e nt!KeWaitForSingleObject+0x19f
fffff880'0d196a70 fffff800'02ee0ed3 nt!NtWaitForSingleObject+0xde
fffff880'0d196ae0 00000000 772f135a nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ fffff80'0d196ae0)
00000000'0107f08 000007fe fd9e10dc ntdll!NtWaitForSingleObject+0xa
00000000'0107f10 00000001 3f6c112e KERNELBASE!WaitForSingleObjectEx+0x79
00000000'0107f70 00000000 00586570 ApplicationC+0x112e
00000000'0107f7b8 00000000 00000000 0x586570

To Be Discussed Later
Complete Dump Analysis
Exercise

© 2013 Software Diagnostics Services
Example: Consumption

1: kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS fffffffa80042d5400
  SessionId: none  Cid: 0004  Peb: 00000000  ParentCid: 0000
  DirBase: 00187000  ObjectTable: ffffa80000017e0  HandleCount: 785.
  Image: System

PROCESS fffffffa8006fa8750
  SessionId: none  Cid: 0144  Peb: 7fffffffda000  ParentCid: 0004
  DirBase: 107226000  ObjectTable: fffffff8a0002dcf90  HandleCount: 32.
  Image: smss.exe

PROCESS fffffffa80083c93b30
  SessionId: 0  Cid: 01ec  Peb: 7fffffffde000  ParentCid: 01a8
  DirBase: 918c1000  ObjectTable: fffffff8a0013b98a0  HandleCount: 679.
  Image: csrss.exe

[...]

PROCESS fffffffa800442ab30
  SessionId: 1  Cid: 1418  Peb: 7fffffffde000  ParentCid: 0840
  DirBase: 11e2c5000  ObjectTable: fffffff8a004cc3b50  HandleCount: 20014.
  Image: ApplicationE.exe

[...]
Example: Corruption

THREAD ffffa8004e2720  Cid 06fc.15c0  Teb: 000007fffae000 Win32Thread: 0000000000000000 WAIT: (UserRequest) UserMode Alertable

fffafa8009a5bb30  ProcessObject
Not impersonating
DeviceMap  ffffa80001ce6b90
Owning Process  ffffa8004364060  Image: ApplicationD.exe

Win32 Start Address ApplicationD (0x000000013f061318)
Stack Init ffffa8000ec10c70 Current ffffa8000ec107c0
Base ffffa8000ec11000 Limit ffffa8000ec00000 Call 0
Priority 11 BasePriority 8 UnusualBoost 0 ForegroundBoost 2 IoPriority 2 PagePriority 5
Child-SP RetAddr Call Site
fffafa800'0ec10800 ffffa800'02ee6f32 nt!KiSwapContext+0x7a
fffafa800'0ec10940 ffffa800'02ee974f nt!KiCommitThreadWait+0x1d2
fffafa800'0ec109d0 ffffa800'031d844e nt!KeWaitForSingleObject+0x19f
fffafa800'0ec10a70 ffffa800'02ee0ed3 nt!NTWaitForSingleObject+0x1d2
fffafa800'0ec10ae0 00000000'772f135a nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ ffffa800'0ec10ae0)
00000000'071e7f8 00000000'77363062 ntdll!NtWaitForSingleObject+0xa
00000000'071e800 00000000'773632a5 ntdll!RtlReportExceptionEx+0x1d2
00000000'071e8f0 00000000'7736330a ntdll!RtlReportExceptionEx+0x19f
00000000'071e970 00000000'77364145 ntdll!RtlpTerminateFailureFilter+0x1a
00000000'071e9a0 00000000'772b85a8 ntdll!RtlReportCriticalFailure+0x96
[...]
00000000'071ea70 00000000'772b97a8 ntdll!RtlDispatchException+0x45a
00000000'071f150 00000000'77364f2 ntdll!RtlRaiseException+0x12f
00000000'071fbd0 00000000'77364736 ntdll!RtlReportCriticalFailureEx+0x62
00000000'071fd90 00000000'77365942 ntdll!RtlpReportHeapFailureFilter+0x26
00000000'071fcb0 00000000'773675f4 ntdll!RtlpHeapHandleError+0x12
00000000'071fc30 00000000'7730dc8f ntdll!RtlpLogHeapFailure+0x44
00000000'071fc60 00000000'771a301a ntdll! ?? ::FNODOBFM::`string'+0x10c54

>> 00000000'071fcb0 00000000'3f061274 kernel32!HeapFree+0xa
00000000'071fdd0 00000000'3f0610c3 ApplicationD+0x1274
[...]

© 2013 Software Diagnostics Services
Example: Special Process

1: kd> !vm

[...]

0458 svchost.exe     1922 (7688 Kb)
0bf4 msseces.exe     1863 (7452 Kb)
0fc8 NisSrv.exe      1845 (7380 Kb)
0584 StageRemoteSer  1776 (7104 Kb)
05dc spoolsv.exe     1729 (6916 Kb)
11e8 mscorsvw.exe    1620 (6480 Kb)
06c6 vmware-authd.e  1593 (6372 Kb)
02d8 services.exe    1560 (6240 Kb)
0584 StageRemoteSer  1558 (6232 Kb)
0ea4 WmiPrvSE.exe    1411 (5644 Kb)
0350 svchost.exe     1283 (5132 Kb)
02e0 lsass.exe      1218 (4872 Kb)
03b8 svchost.exe     1214 (4856 Kb)
09b0 SftService.exe  1171 (4684 Kb)
13fc daemonu.exe     1117 (4468 Kb)
055c igfxpers.exe    1080 (4320 Kb)
088c DTLite.exe      1051 (4204 Kb)
1194 DTShellHlp.exe  1080 (4032 Kb)
04c8 taskhost.exe    975 (3900 Kb)
0b34 STService.exe   933 (3732 Kb)
0270 csrss.exe       855 (3420 Kb)
029c winlogon.exe    845 (3380 Kb)

To Be Discussed Later

Complete Dump Analysis Exercise
Common Mistakes

- Not switching to the appropriate context
- Not looking at full stack traces
- Not looking at all stack traces
- Not using checklists
- Not looking past the first found evidence
- Not comparing to the reference debugger output
- Not doing explicit symbol qualification: module!symbol

**Note:** Listing both x86 and x64 stack traces ([windbg.org](http://windbg.org))

```
.load wow64exts
!for_each_thread "!thread @#Thread 16;.thread /w @#Thread; .reload; kv 256; .effmach AMD64"
```
Agenda (Exercise)

- Run processes that model abnormal behavior
- Generate a complete memory dump
- Analyze the memory dump

Note: Due to security concerns I’m not making a complete memory dump downloadable. You can generate your own complete memory dump after downloading and running model applications
Exercise: Run Processes

These processes model specific patterns:

ApplicationA, ApplicationB, ApplicationC, ApplicationD, ApplicationE

For demonstration I run x64 versions plus x86 version of ApplicationA

Note: Run applications in alphabetical order

Can be downloaded from this location:
http://www.patterndiagnostics.com/Training/Webinars/FCMDA-Examples.zip

There are x86 and x64 versions
Exercise: Force A Dump

The system is x64 Windows 7

Note: Wait at least 10 seconds after running model applications to have them properly initialize their dependencies.
Exercise: Dump Analysis

Now I switch to a WinDbg session...
Agenda (Guide)

- Patterns related to complete memory dumps
- Pattern cooperation case studies from complete memory dumps
- Pattern Map
Pattern Examples

Some patterns that are relevant to complete memory dumps:

- Incorrect Symbolic Information
- Semantic Split
- Paged Out Data
- Wait Chain (thread objects)
- Wait Chain (LPC/ALPC)
- Last Error Collection
- Suspended Thread
- Coupled Processes (strong)
- Truncated Dump
- Spiking Thread
- Deadlock (critical sections)
- Problem Vocabulary
- Semantic Structures
- Virtualized System

- No System Dumps
- Message Box
- Inconsistent Dump
- Wait Chain (critical sections)
- Wait Chain (process objects)
- Special Process
- Historical Information
- Stack Trace Collection
- Insufficient Memory (handle leak)
- Main Thread
- Suspended Thread
- Pleiades
- Dual Stack Trace
Case Studies

17 pattern interaction case studies using complete memory dumps:

http://www.dumpanalysis.org/blog/index.php/category/complete-memory-dump-analysis/
WinDbg Command Map

Pattern <-> WinDbg command

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug</td>
<td>!c</td>
</tr>
<tr>
<td>Continue</td>
<td>!n</td>
</tr>
<tr>
<td>Stepping</td>
<td>!s</td>
</tr>
<tr>
<td>Break</td>
<td>!b</td>
</tr>
<tr>
<td>Continue</td>
<td>!n</td>
</tr>
<tr>
<td>Stepping</td>
<td>!s</td>
</tr>
<tr>
<td>Break</td>
<td>!b</td>
</tr>
</tbody>
</table>
Reference Resources

- WinDbg.org
- DumpAnalysis.org
- Software Diagnostics Library
- Debugging TV
- Memory Dump Analysis Anthology (volumes 1 – 6)

Volume 7 is planned for January 2014
Training Resources

- **Accelerated Windows Memory Dump Analysis, 3rd edition**
- **Advanced Windows Memory Dump Analysis with Data Structures, 2nd edition**
- **Accelerated Windows Malware Analysis with Memory Dumps**
- **Accelerated Windows Debugging**
Q&A

Please send your feedback using the contact form on PatternDiagnostics.com
Thank you for attendance!