Windows Memory Dump Analysis

Accelerated

Version 2.0

Dmitry Vostokov
Software Diagnostics Services
Prerequisites

Basic Windows troubleshooting
Training Goals

- Review fundamentals
- Learn how to analyze process dumps
- Learn how to analyze kernel dumps
- Learn how to analyze complete dumps
Training Principles

- Talk only about what I can show
- Lots of pictures
- Lots of examples
- Original content and examples
Schedule Summary

Day 1
- Analysis Fundamentals (1 hour)
- Process Memory Dumps (1 hour)

Day 2
- Process Memory Dumps (2 hours)

Day 3
- Kernel Memory Dumps (2 hours)

Day 4
- Complete Memory Dumps (2 hours)
- Remaining Process Memory Dumps
Part 1: Fundamentals
Process Space (x86)
Process Space (x64)

User Space

Kernel Space

© 2013 Software Diagnostics Services
Application/Process/Module

Kernel Space

User Space (PID 102)

Notepad.exe

user32.dll

Notepad.exe

user32.dll
OS Kernel/Driver/Module

User Space

Kernel Space

Driver

nt

Driver.sys

Ntoskrnl.exe
Process Virtual Space

Notepad

User Space (PID 102)

user32

nt

Kernel Space

Driver

00000000 ... FFFFFFFF

© 2013 Software Diagnostics Services
Process Memory Dump

WinDbg Commands

Imv command lists modules and their description
Kernel Memory Dump

WinDbg Commands

```
lmv command lists modules and their description
```

User Space (PID 102)

Kernel Space

Driver

Notepad

user32

nt

MEMORY.DMP

© 2013 Software Diagnostics Services
Complete Memory Dump

WinDbg Commands

```
 procession switches between process virtual spaces (kernel space part remains the same)
```
Process Threads

WinDbg Commands

- Process dumps: `~<n>s` switches between threads

- Kernel/Complete dumps: `~<n>s` switches between processors

- `.thread` switches between threads
System Threads

WinDbg Commands

Kernel/Complete dumps:
~<n>s switches between processors
.thread switches between threads
Thread Stack Raw Data

WinDbg Commands

Process dumps:
!teb

Kernel dumps:
!thread

Complete dumps:
!teb for user space
!thread for kernel space

Data:
dc / dps / dpp / dpa / dpu

© 2013 Software Diagnostics Services
Thread Stack Trace

User Stack for TID 102

FunctionA()
{
    ...
    FunctionB();
    ...
}
FunctionB()
{
    ...
    FunctionC();
    ...
}
FunctionC()
{
    ...
    FunctionD();
    ...
}

WinDbg Commands

0:000> k
Module!FunctionD
Module!FunctionC+130
Module!FunctionB+220
Module!FunctionA+110

Return address Module!FunctionC+130
Return address Module!FunctionB+220
Return address Module!FunctionA+110

Module!FunctionA
Resumes from address Module!FunctionA+110
Saves return address Module!FunctionA+110

Module!FunctionB
Resumes from address Module!FunctionB+220
Saves return address Module!FunctionB+220

Module!FunctionC
Resumes from address Module!FunctionC+130
Saves return address Module!FunctionC+130

Module!FunctionD

© 2013 Software Diagnostics Services
Thread Stack Trace (no PDB)

User Stack for TID 102
- Return address Module+43130
- Return address Module+32220
- Return address Module+22110

Symbol file Module.pdb
- FunctionA 22000 - 23000
- FunctionB 32000 - 33000
- FunctionC 43000 - 44000
- FunctionD 54000 - 55000

No symbols for Module

WinDbg Commands
- 0:000> k
- Module+0
- Module+43130
- Module+32220
- Module+22110
Exceptions (Access Violation)

WinDbg Commands

- address=????????
- Set exception context (process dump): .cxr
- Set trap context (kernel/complete dump): .trap
- Check address: !pte

User Space (PID 306)

ApplicationA

ModuleA

User Stack for TID 102

User Stack for TID 204

User Stack for TID 204

TID 102

TID 204

NTDLL

USER32

Invalid memory access

NULL pointer
Exceptions (Runtime)

ApplicationA

User Space (PID 306)

User Stack for TID 102

throws error ModuleA

TID 102

User Stack for TID 204

TID 204

User Stack for TID 204

User Space (PID 306)

user32

ntdll

© 2013 Software Diagnostics Services
Pattern-Driven Analysis

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


Part 2: Practice Exercises
Links

- Memory Dumps:
  Not available in preview version

- Exercise Transcripts:
  Not available in preview version
Exercise 0

- **Goal:** Install Debugging Tools for Windows and learn how to set up symbols correctly
- **Patterns:** Incorrect Stack Trace
Process Memory Dumps

Exercises P1-P16
Exercise P1

- **Goal:** Learn how to see dump file type and version, get a stack trace, check its correctness, perform default analysis, list modules, check their version information, check process environment

- **Patterns:** Manual Dump; Stack Trace; Not My Version; Environment Hint
Exercise P2

- **Goal:** Learn how to list stack traces, check their correctness, perform default analysis, list modules, check their version information, check process environment; dump module data

- **Patterns:** Manual Dump; Stack Trace; Not My Version; Environment Hint; Unknown Component
Exercise P3

- **Goal:** Learn how to list stack traces, check their correctness, perform default analysis, list modules, check their version information, check thread age and CPU consumption

- **Patterns:** Stack Trace Collection
Exercise P4

- **Goal:** Learn to recognize exceptions in process memory dumps and get their context

- **Patterns:** Exception Thread; Multiple Exceptions; NULL Pointer
Exercise P5

- **Goal:** Learn how to load application symbols, recognize exceptions in process memory dumps and get their context

- **Patterns:** Exception Thread; Multiple Exceptions; NULL Pointer
Exercise P6

- **Goal:** Learn how to recognize heap corruption

- **Patterns:** Exception Thread; Dynamic Memory Corruption
Exercise P7

- **Goal:** Learn how to recognize heap corruption and check error and status codes

- **Patterns:** Exception Thread; Dynamic Memory Corruption
Exercise P8

- **Goal:** Learn how to recognize CPU spikes, invalid pointers and disassemble code

- **Patterns:** Exception Thread; Wild Code; CPU Spike; Multiple Exceptions; NULL Code Pointer; Invalid Pointer
Exercise P9

- **Goal:** Learn how to recognize critical section waits and deadlocks, dump raw stack data and see hidden exceptions
- **Patterns:** Wait Chain; Deadlock; Hidden Exception
Deadlock

Thread 1 (owns)

Critical Section 000000013fd7ef08

Thread 1 (waiting)

Thread 2 (waiting)

Critical Section 000000013fd7eee0

Thread 2 (owns)
Exercise P10

- **Goal:** Learn how to recognize application heap problems, buffer and stack overflow patterns and analyze raw stack data

- **Patterns:** Double Free; Local Buffer Overflow; Stack Overflow
Exercise P11

- **Goal:** Learn how to analyze various patterns, raw stacks and execution residue

- **Patterns:** Divide by Zero; C++ Exception; Multiple Exceptions; Execution Residue
Exercise P12

- **Goal:** Learn how to load the correct .NET WinDbg extension and analyze managed space

- **Patterns:** CLR Thread; Version-Specific Extension; Managed Code Exception; Managed Stack Trace
Exercise P13

- **Goal:** Learn how to analyze 32-process saved as a 64-bit process memory dump

- **Patterns:** Virtualized Process; Message Box; Execution Residue
Exercise P14

- **Goal:** Learn how to analyze process memory leaks

- **Patterns:** Spiking Thread; Thread Age; Memory Leak (process heap)
Parameters and Locals

Debugging TV Frames episode 0x18
Symbol Types

- Exported and imported names
- Function and variable names
- Data types
Exercise P15

- **Goal:** Learn how to navigate function parameters in cases of reduced symbolic information in 32-bit process memory dumps

- **Patterns:** Reduced Symbolic Information
Exercise P16

- **Goal:** Learn how to navigate function parameters in x64 process memory dumps

- **Patterns:** False Function Parameters, Injected Symbols
Pattern Links

- Spiking Thread
- C++ Exception
- Divide by Zero
- Heap Corruption
- Execution Residue
- Invalid Pointer
- Manual Dump
- Managed Stack Trace
- Not My Version
- NULL Code Pointer
- Stack Trace Collection
- Environment Hint
- Unknown Component
- Virtualized Process
- Version-Specific Extension
- False Function Parameters
- Reduced Symbolic Information

- CLR Thread
- Critical Section Deadlock
- Double Free
- Exception Stack Trace
- Hidden Exception
- Local Buffer Overflow
- Managed Code Exception
- Multiple Exceptions
- NULL Data Pointer
- Stack Trace
- Stack Overflow
- Wild Code
- Wait Chain
- Message Box
- Memory Leak
- Injected Symbols
Kernel Memory Dumps

Exercises K1-K5
Exercise K1

- **Goal:** Learn how to get various information related to hardware, system, sessions, processes, threads and modules

- **Patterns:** Invalid Pointer; Virtualized System; Stack Trace Collection
Exercise K2

- **Goal:** Learn how to check and compare kernel pool usage
- **Patterns:** Manual Dump; Insufficient Memory (kernel pool)
Exercise K3

- **Goal:** Learn how to recognize pool corruption and check pool data

- **Patterns:** Dynamic Memory Corruption (kernel pool); Execution Residue
Exercise K4

- **Goal:** Learn how to check hooked or invalid code and kernel raw stack

- **Patterns:** Null Pointer; Hooked Functions (kernel space); Execution Residue; Coincidental Symbolic Information
Exercise K5

- **Goal:** Learn how to check I/O requests
- **Patterns:** Blocking File
Pattern Links

- Manual Dump
- Virtualized System
- Insufficient Memory
- Execution Residue
- Hooked Functions
- Blocking File
- Invalid Pointer
- Stack Trace Collection
- Dynamic Memory Corruption
- Null Pointer
- Coincidental Symbolic Information
Additional Pattern Links

ERESOURCE patterns and case studies
Complete Memory Dumps

Exercises C1-C2
Memory Spaces

- Complete memory == Physical memory
- We always see the current process space

User Space
- current process A (NotMyFault.exe)

Kernel Space
- current process A (NotMyFault.exe)

User Space
- current process B (svchost.exe)

Kernel Space
- current process A (NotMyFault.exe)

WinDbg Commands
switching to a different process context:

.process /r /p
Major Challenges

- Multiple processes (user spaces) to examine
- User space view needs to be correct when we examine another thread

WinDbg Commands

dump all stack traces:
!process 0 3f

User Space
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
  ..........................................
  ```
Exercise C1

- **Goal:** Learn how to get various information related to processes, threads and modules

- **Patterns:** Stack Trace Collection
Example: Blocked Thread

THREAD fffffa800451db60  Cid 07f4.0b8c  Teb: 000007ffffffd6000 Win32Thread: fffff900c27c0c30 WAIT: (WrUserRequest) UserMode Non-Abletable  

fffffa800451db60  SynchronizationEvent  
Not impersonating  
DeviceMap  
Owning Process  
Image: ApplicationA.exe  

Stack Init fffff88005b7fbd0 Current fffff88005b7f870  
Base fffff88005b8000 Limit fffff88005b77000 Call 0  
Priority 11 BasePriority 8 UnusualBoost 0 ForegroundBoost 2 IoPriority 2 PagePriority 5  

Child-SP RetAddr Call Site  
fffff880`05b7f8b0 fffff800`01a93992 nt!KiSwapContext+0x7a  
fffff880`05b7f9f0 fffff800`01a95cff nt!KiCommitThreadWait+0x1d2  
fffff880`05b7fa80 fffff900`0011b557 nt!KeWaitForSingleObject+0x19f  
fffff880`05b7fb20 fffff900`0011b5f1 win32k!xxxRealSleepThread+0x257  
fffff880`05b7fbc0 fffff900`0012e22e win32k!xxxSleepThread+0x59  
fffff880`05b7fbf0 fffff800`01a8b993 win32k!NtUserWaitMessage+0x46  
fffff880`05b7fc20 00000000`775cbf5a nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ ffffff880`05b7fc20)  
00000000`022ff7c8 00000000`775d7214 USER32!ZwUserWaitMessage+0x4e  
00000000`022ff7d0 00000000`775d7214 USER32!DialogBox2+0x274  
00000000`022ff860 00000000`776227f0 USER32!InternalDialogBox+0x135  
00000000`022ff880 00000000`77621ae5 USER32!SoftModalMessageBox+0xb4  
00000000`022ff9f0 00000000`7762133b USER32!MessageBoxWorker+0x31d  
00000000`022ffbb0 00000000`77621232 USER32!MessageBoxTimeout+0xb3  
00000000`022ffcc0 00000000`3f3c11fb ApplicationA+0x1089  
00000000`022ffcc0 00000000`3f3c11fb ApplicationA+0x11fb  
00000000`022ffcc0 00000000`3f3c12a5 ApplicationA+0x12a5  
00000000`022ffcc0 00000000`776cf56d ApplicationA+0x12a5  
00000000`022ffcc0 00000000`77803281 kernel32!BaseThreadInitThunk+0xad  
00000000`022ffcc0 00000000`00000000 ntdll!RtlUserThreadStart+0xad  

© 2013 Software Diagnostics Services
Example: Wait Chain

THREAD ffffa8004562b60  Cid 0b34.0858  Teb: 000007ffffffae000  Win32Thread: 0000000000000000  WAIT: (UserRequest) UserMode Non-Alertable

Mutant - owning thread ffffa8004523b60

Not impersonating
DeviceMap ffffa8001e84c00
Owning Process ffffa8005400b30  Image: ApplicationC.exe
Attached Process N/A  Image: N/A
Wait Start TickCount 36004  Ticks: 4286 (0:00:01:06.862)
Context Switch Count 2
UserTime 00:00:00.000
KernelTime 00:00:00.000
Win32 Start Address ApplicationC (0x000000013f7012a0)
Stack Init ffffa88005b1d900  Current ffffa88005b1d900
Base ffffa88005b1e000  Limit ffffa88005b18000  Call 0
Priority 11  BasePriority 8  UnusualBoost 0  ForegroundBoost 2  IoPriority 2  PagePriority 5
Child-SP RetAddr Call Site
ffff880`05b1d940 fffff800`01a93992 nt!KiSwapContext+0x7a
ffff880`05b1da80 ffffe800`01a95ccf nt!KiCommitThreadWait+0x1d2
ffff880`05b1db10 ffffe800`01d871d2 nt!KeWaitForSingleObject+0x19f
ffff880`05b1dbb0 ffffe800`01a8b993 nt!NtWaitForSingleObject+0xb2
ffff880`05b1dc20 00000000`7781fefa nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ ffffa880`05b1dc20)
00000000`00e2f658 0000007fe`fda910ac ntdll!NtWaitForSingleObject+0xa
00000000`00e2f660 00000001`3f70112e KERNELBASE!WaitForSingleObjectEx+0x79
00000000`00e2f700 00000001`3f70128b ApplicationC+0x11e
00000000`00e2f730 00000001`3f701335 ApplicationC+0x12b
00000000`00e2f760 00000000`776cf56d ApplicationC+0x135
00000000`00e2f790 00000000`77803281 kernel32!BaseThreadInitThunk+0x0
00000000`00e2f7c0 00000000`0000000 ntdll!RtlUserThreadStart+0x1d

© 2013 Software Diagnostics Services
Example: Handle Leak

1: kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS fffffa8003baa890
  SessionId: none  Cid: 0004    Peb: 00000000  ParentCid: 0000
  DirBase: 00187000  ObjectTable: fffff8a000001a80  HandleCount: 558.
  Image: System

PROCESS fffffa8004277870
  SessionId: none  Cid: 011c    Peb: 7fffffd000  ParentCid: 0004
  DirBase: 133579000  ObjectTable: fffff8a00000f3d0  HandleCount: 35.
  Image: smss.exe

PROCESS fffffa80048f3950
  SessionId: 0  Cid: 016c    Peb: 7fffffd000  ParentCid: 0154
  DirBase: 128628000  ObjectTable: fffff8a001d62f90  HandleCount: 387.
  Image: csrss.exe

[...]

PROCESS fffffa800541a060
  SessionId: 1  Cid: 0b94    Peb: 7fffffde000  ParentCid: 06ac
  DirBase: a6ba9000  ObjectTable: fffff8a0098efaf0  HandleCount: 20013.
  Image: ApplicationE.exe

[...]
Example: Corruption

THREAD ffffffff8004514060  Cid 0abc.087c  Teb: 000007fffffff000 Win32Thread: 0000000000000000 WAIT: (UserRequest) UserMode
Alertable

fffffa800518fb30  ProcessObject

Child-SP    RetAddr           Call Site
fffff880`05a6c940 fffff800`01a93992 nt!KiSwapContext+0x7a
fffff880`05a6ca80 fffff800`01a95cff nt!KiCommitThreadWait+0x1d2
fffff880`05a6cb10 fffff800`01d871d2 nt!KewaitForSingleObject+0x19f
fffff880`05a6cb00 fffff800`01a8b993 nt!NtWaitForSingleObject+0x1b2
fffff880`05a6cc20 00000000`7781fefa nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ fffff880`05a6cc20)
00000000`00ddef928 00000000`77895ce2 ntdll!NtWaitForSingleObject+0x1a
00000000`00ddef930 00000000`77895e85 ntdll!NtRtlReportExceptionEx+0x1d2
00000000`00ddefea20 00000000`77895eaa ntdll!NtRtlReportException+0x1b5
00000000`00ddefea0 00000000`77896d25 ntdll!RtlpTerminateFailureFilter+0x1a
00000000`00ddefea0 00000000`7789e1a8 ntdll!RtlReportCriticalFailure+0x96
00000000`00ddefea0 00000000`7789554d ntdll!_C_specific_handler+0x8c
00000000`00ddefea0 00000000`77895d1c ntdll!RtlpExecuteHandlerForException+0xd
00000000`00ddefa0 00000000`777e62ee ntdll!RtlDispatchException+0x3cb
00000000`00ddefb0 00000000`777e6cd2 ntdll!RtlRaiseException+0x221
00000000`00ddefba0 00000000`777e7396 ntdll!RtlReportCriticalFailure+0x62
00000000`00ddefba0 00000000`777e8bc2 ntdll!RtlpReportHeapFailure+0x1a
00000000`00ddefba0 00000000`777e9ac4 ntdll!RtlpHeapHandleError+0x12
00000000`00ddefba0 00000000`7783c1cd ntdll!RtlpLogHeapFailure+0xa4
00000000`00ddefba0 00000000`776d27c7 ntdll! ?? ::FDODBFM::`string'+0x123b4

>>> 00000000`00ddefa0 00000001`3fa71274 kernel32!HeapFree+0xa
00000000`00ddefe0 00000001`3fa710c3 ApplicationD+0x124
00000000`00ddefe0 00000001`3fa71303 ApplicationD+0x103
00000000`00ddefe0 00000001`3fa71ad3 ApplicationD+0x1303
00000000`00ddefe0 00000000`776cf56d ApplicationD+0x13ad
00000000`00ddefe0 00000000`77883281 kernel32!BaseThreadInitThunk+0x1d
00000000`00ddefe0 00000000`00000000 ntdll!RtlUserThreadStart+0x1d
Example: Special Process

1: kd> !vm

[...]

<table>
<thead>
<tr>
<th>PID</th>
<th>Process Name</th>
<th>ID</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0744</td>
<td>svchost.exe</td>
<td>19725</td>
<td>78900 Kb</td>
</tr>
<tr>
<td>06ac</td>
<td>explorer.exe</td>
<td>11444</td>
<td>45776 Kb</td>
</tr>
<tr>
<td>0920</td>
<td>iexplore.exe</td>
<td>8828</td>
<td>35312 Kb</td>
</tr>
<tr>
<td>0354</td>
<td>svchost.exe</td>
<td>5589</td>
<td>22356 Kb</td>
</tr>
<tr>
<td>040c</td>
<td>audiodg.exe</td>
<td>4003</td>
<td>16012 Kb</td>
</tr>
<tr>
<td>0334</td>
<td>svchost.exe</td>
<td>3852</td>
<td>15408 Kb</td>
</tr>
<tr>
<td>04e4</td>
<td>spoolsv.exe</td>
<td>3230</td>
<td>12920 Kb</td>
</tr>
<tr>
<td>012c</td>
<td>svchost.exe</td>
<td>2802</td>
<td>11208 Kb</td>
</tr>
<tr>
<td>0168</td>
<td>iexplore.exe</td>
<td>2186</td>
<td>8424 Kb</td>
</tr>
<tr>
<td>0384</td>
<td>svchost.exe</td>
<td>2090</td>
<td>8360 Kb</td>
</tr>
<tr>
<td>042c</td>
<td>svchost.exe</td>
<td>1938</td>
<td>7752 Kb</td>
</tr>
<tr>
<td>0218</td>
<td>lsass.exe</td>
<td>1314</td>
<td>5256 Kb</td>
</tr>
<tr>
<td>03d4</td>
<td>svchost.exe</td>
<td>1128</td>
<td>4512 Kb</td>
</tr>
<tr>
<td>***</td>
<td>WerFault.exe</td>
<td>1107</td>
<td>4428 Kb</td>
</tr>
<tr>
<td>0210</td>
<td>services.exe</td>
<td>1106</td>
<td>4424 Kb</td>
</tr>
<tr>
<td>0288</td>
<td>svchost.exe</td>
<td>980</td>
<td>3928 Kb</td>
</tr>
<tr>
<td>02d8</td>
<td>svchost.exe</td>
<td>891</td>
<td>3564 Kb</td>
</tr>
<tr>
<td>0438</td>
<td>msdtc.exe</td>
<td>851</td>
<td>3404 Kb</td>
</tr>
<tr>
<td>071c</td>
<td>mscorsvw.exe</td>
<td>821</td>
<td>3284 Kb</td>
</tr>
<tr>
<td>0378</td>
<td>taskhost.exe</td>
<td>795</td>
<td>3180 Kb</td>
</tr>
<tr>
<td>01a8</td>
<td>psxss.exe</td>
<td>685</td>
<td>2740 Kb</td>
</tr>
<tr>
<td>08a0</td>
<td>jusched.exe</td>
<td>667</td>
<td>2668 Kb</td>
</tr>
<tr>
<td>09e0</td>
<td>jucheck.exe</td>
<td>621</td>
<td>2484 Kb</td>
</tr>
<tr>
<td>0828</td>
<td>mscorsvw.exe</td>
<td>600</td>
<td>2400 Kb</td>
</tr>
<tr>
<td>0538</td>
<td>mdm.exe</td>
<td>595</td>
<td>2380 Kb</td>
</tr>
<tr>
<td>0220</td>
<td>lsm.exe</td>
<td>595</td>
<td>2380 Kb</td>
</tr>
</tbody>
</table>

[...]
Exercise C2

- **Goal:** Learn how to recognize various abnormal software behavior patterns

- **Patterns:** Special Process; Handle Leak; Spiking Thread; Stack Trace Collection; Message Box; Wait Chain; Exception Thread
Wait Chain

- Critical Section 00a9b7c0
  - Thread 83336a00 (owns)
  - Thread 886ee030 (waiting)

- Process ApplicationC
  - Thread 830f9990 (waiting)

- Critical Section 00a9b7a8
  - Thread 886ee030 (owns)

- Process ApplicationB
  - Thread 832be6d8
  - Mutant 00a9b7c0
  - Thread 83336a00 (waiting)
  - Thread 832be6d8 (owns)
Pattern Links

Special Process
Handle Leak
Spiking Thread
Stack Trace Collection
Message Box
Wait Chain (critical sections)
Exception Stack Trace

Also other patterns are present in C2 memory dump (not shown in exercise transcript):

Wait Chain (window messaging)  Paged Out Data
Wait Chain (LPC/ALPC)
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

**Note:** Listing both x86 and x64 stack traces

http://www.dumpanalysis.org/blog/index.php/2010/02/09/complete-stack-traces-from-x64-system/
Kernel Minidumps

Memory Dump Analysis Anthology, Volume 1
pp. 43 - 67
Pattern Classification

Space/Mode
Hooksware
DLL Link Patterns
Contention Patterns
Stack Trace Patterns
Exception Patterns
Module Patterns
Thread Patterns
Dynamic Memory Corruption Patterns
.NET / CLR / Managed Space Patterns

Memory dump type
Wait Chain Patterns
Insufficient Memory Patterns
Stack Overflow Patterns
Symbol Patterns
Meta-Memory Dump Patterns
Optimization Patterns
Process Patterns
Deadlock and Livelock Patterns
Executive Resource Patterns
Pattern Case Studies

70 multiple pattern case studies:

http://www.dumpanalysis.org/blog/index.php/pattern-cooperation/

Pattern Interaction chapters in Memory Dump Analysis Anthology
Resources

- WinDbg Help / WinDbg.org (quick links)
- DumpAnalysis.org
- Debugging.TV
- Windows Internals, 6th ed.
- Windows Debugging: Practical Foundations
- x64 Windows Debugging: Practical Foundations
- Advanced Windows Debugging
- Windows Debugging Notebook: Essential User Space WinDbg Commands
- Memory Dump Analysis Anthology
Q&A

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