Public Preview Version

.NET Memory Dump Analysis Accelerated

Version 2.0

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

WinDbg Commands

We use these boxes to introduce some WinDbg commands used in practice exercises

Basic .NET programming and debugging

Training Goals

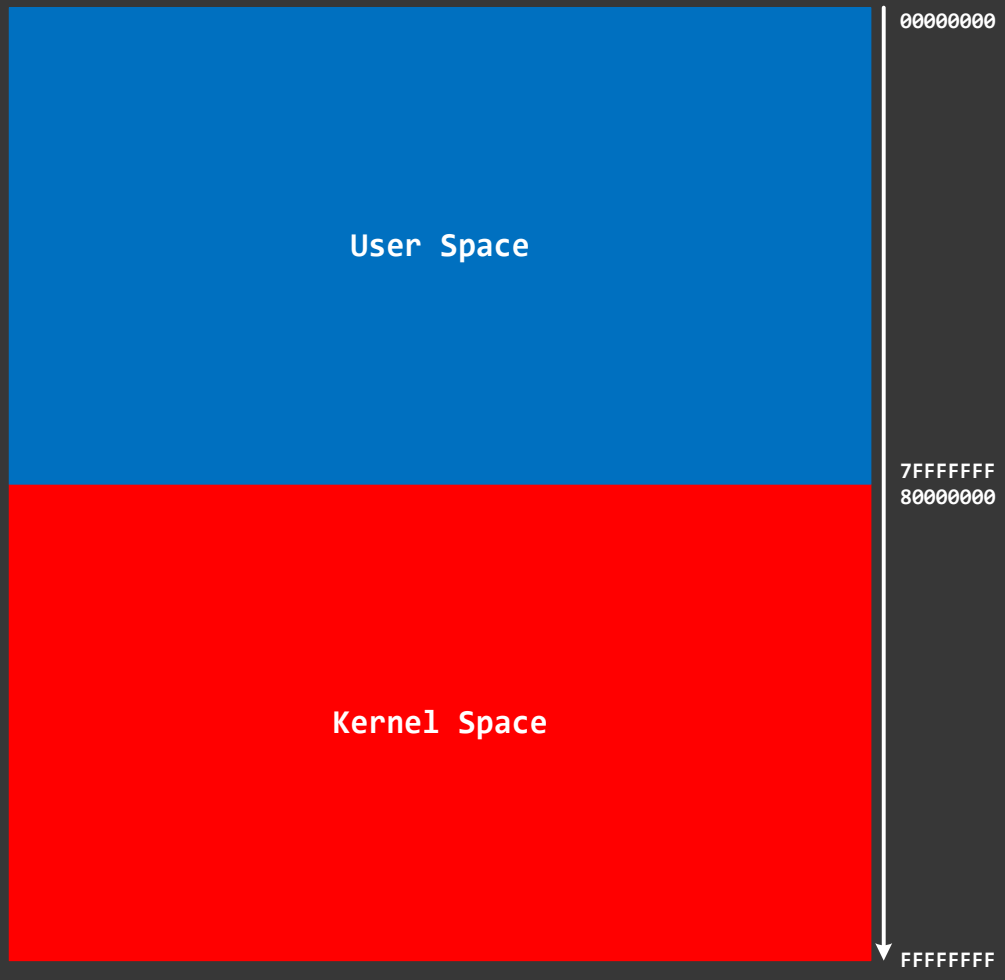
- Review fundamentals
- Learn how to analyze process dumps
- Learn necessary commands in context
- Cover CLR 2 / CLR 4 (x86 and x64)

Training Principles

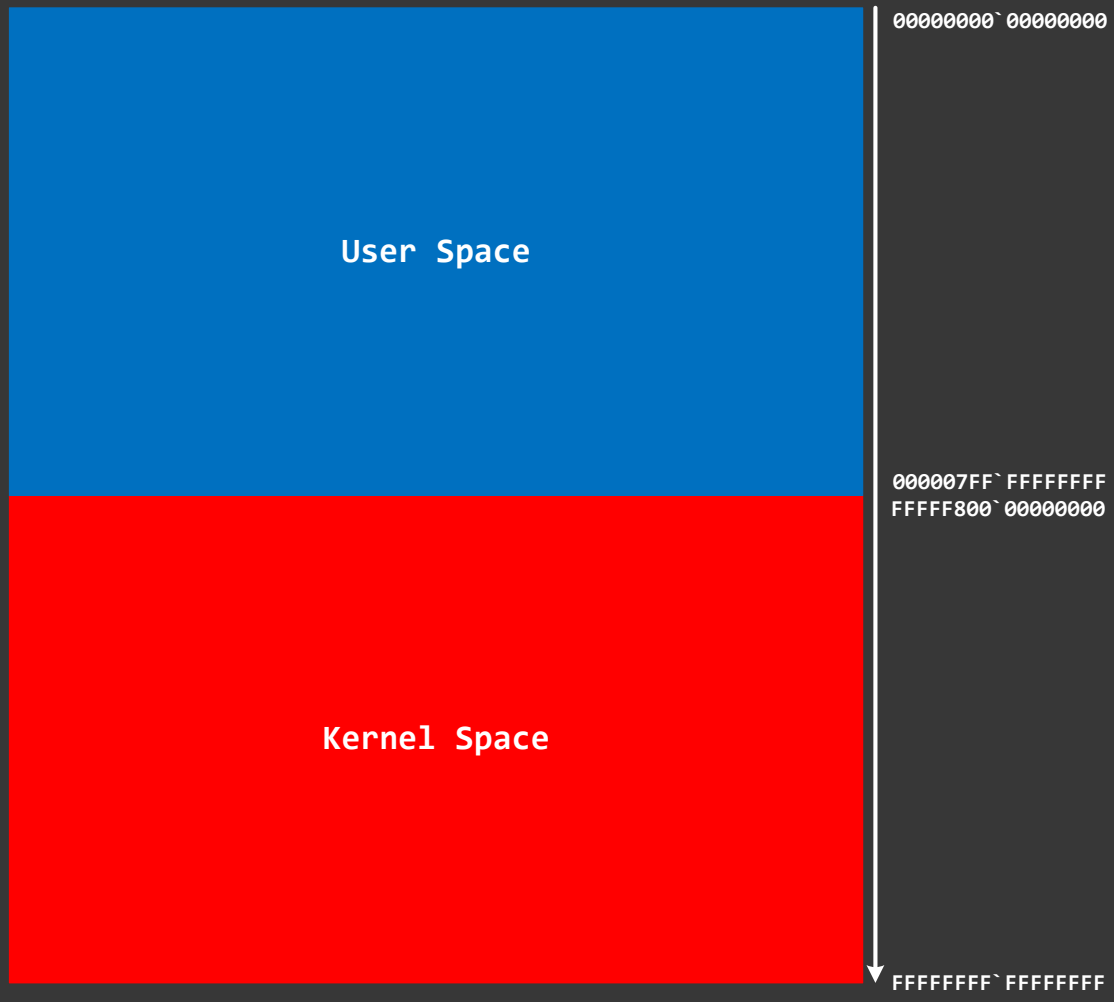
- Talk only about what I can show
- Lots of pictures
- Lots of examples
- Original content and examples

Part 1: Fundamentals

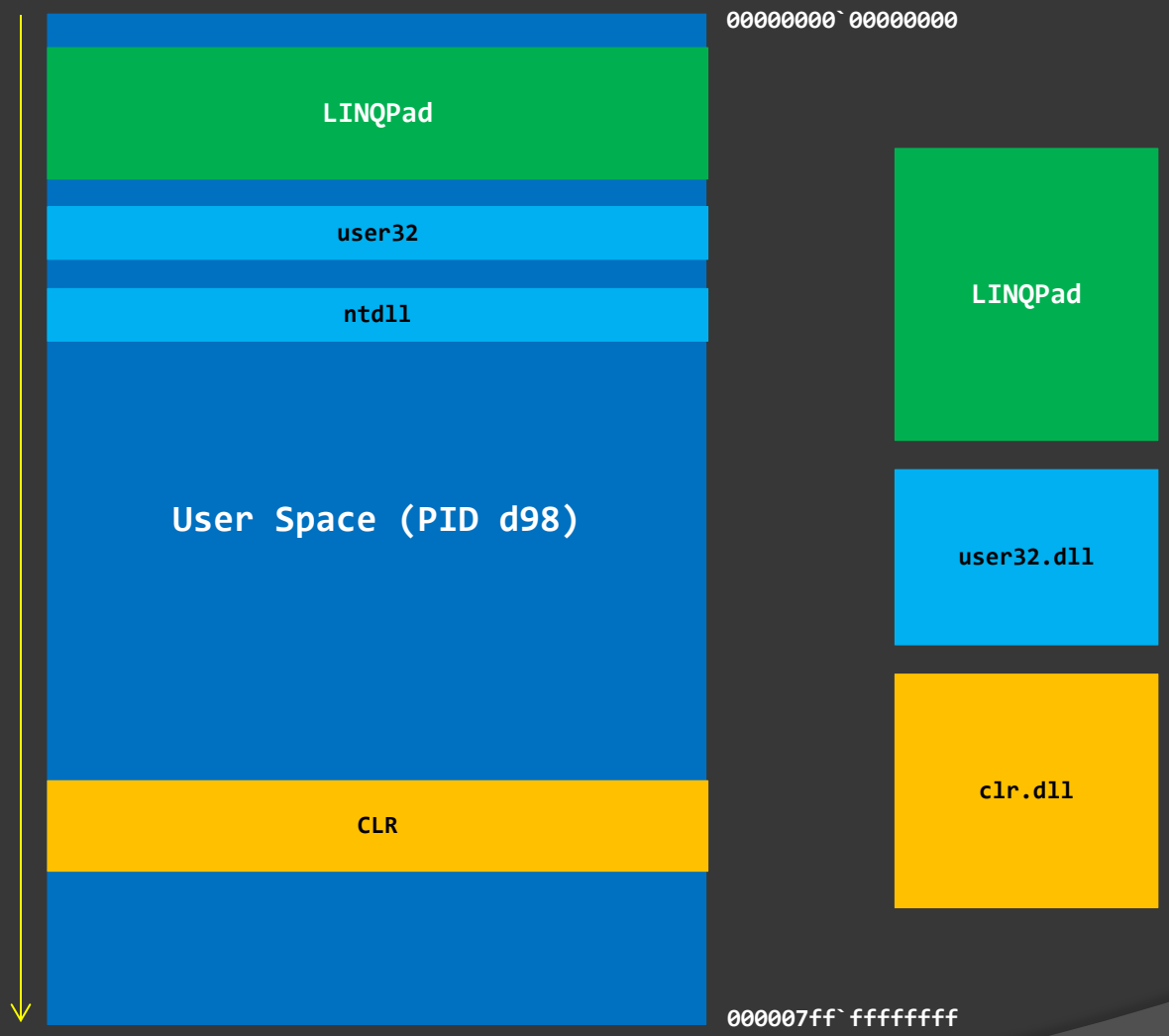
Memory Space (x86)



Memory Space (x64)



User / Managed Space



WinDbg Commands

lmv command lists all loaded modules (EXE and DLLs)

Types/Assemblies/Modules

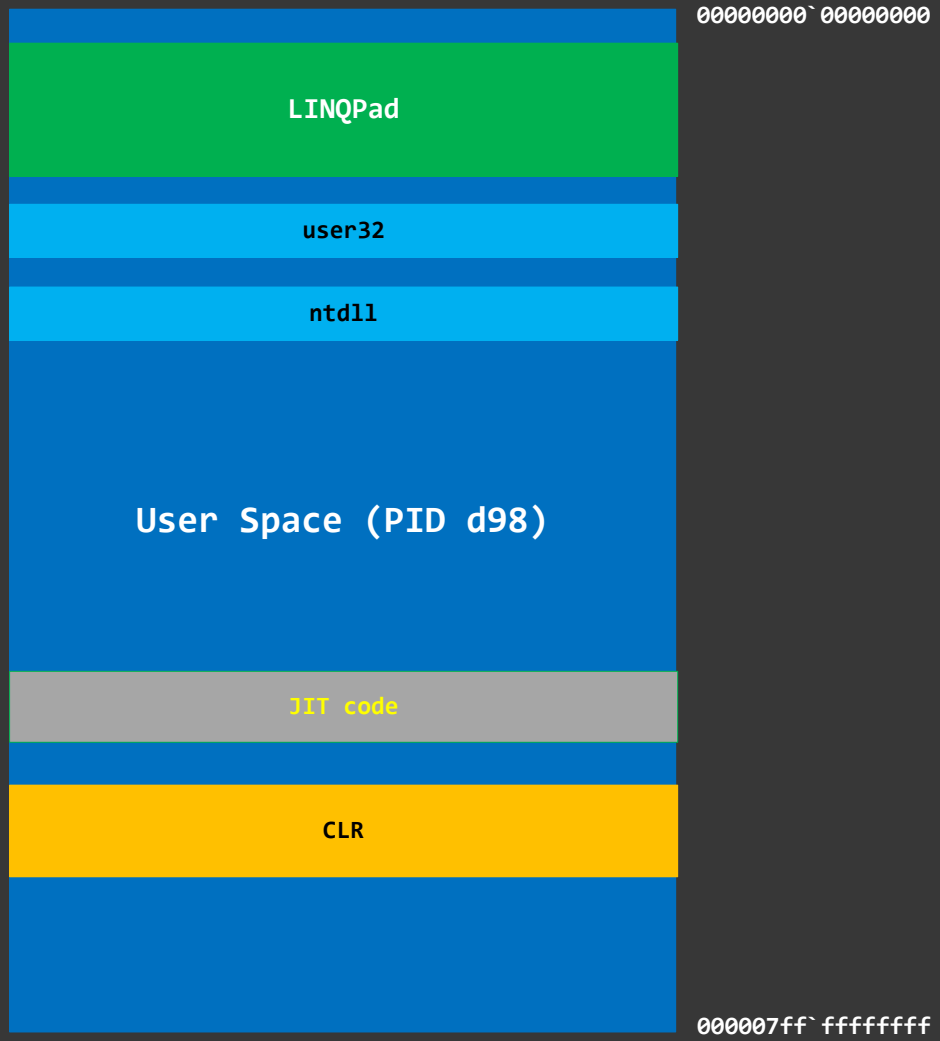
WinDbg Commands

lmv command lists all loaded modules (EXE and DLLs)

!IP2MD command shows type method and module address

!DumpModule command shows module name

Process Threads



WinDbg Commands

.load *<a path to SOS>*
Loads SOS WinDbg extension

~<n>s command switches between threads

k command shows unmanaged stack trace

!Threads command shows managed threads

!CLRStack command shows managed stack trace

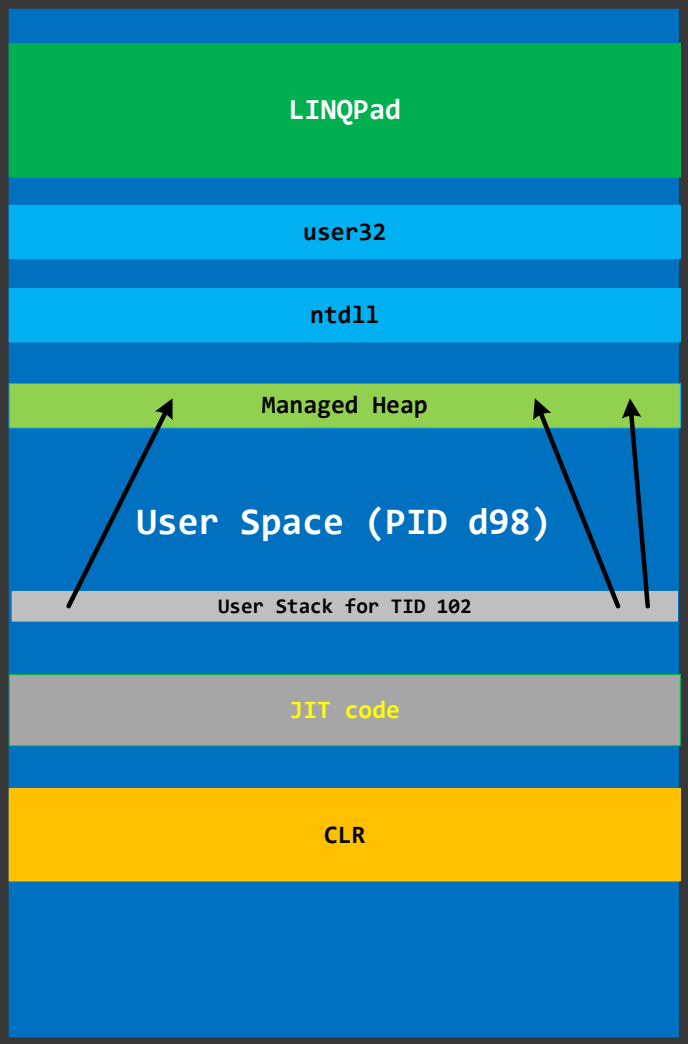
Example

```
0:000> kL
ChildEBP RetAddr
0020e95c 7720b5b4 ntdll!KiFastSystemCallRet
0020e960 68e6737a user32!NtUserWaitMessage+0xc
0020e9f4 68e66e2c System_Windows_Forms_ni+0x22737a
0020ea50 68e66c81 System_Windows_Forms_ni+0x226e2c
0020ea80 68df366d System_Windows_Forms_ni+0x226c81
0020ea98 002b31fd System_Windows_Forms_ni+0x1b366d
0020eb7c 002b1515 0x2b31fd
0020ee24 6ce421db 0x2b1515
0020ee34 6ce64a2a clr!CallDescrWorker+0x33
0020eeb0 6ce64bcc clr!CallDescrWorkerWithHandler+0x8e
0020efe8 6ce64c01 clr!MethodDesc::CallDescr+0x194
0020f004 6ce64c21 clr!MethodDesc::CallTargetWorker+0x21
0020f01c 6cf2ce82 clr!MethodDescCallSite::Call+0x1c
0020f180 6cf2cf90 clr!ClassLoader::RunMain+0x24c
0020f3e8 6cf2cda4 clr!Assembly::ExecuteMainMethod+0xc1
0020f8cc 6cf2d199 clr!SystemDomain::ExecuteMain...
0020f920 6cf2d09a clr!ExecuteEXE+0x58
0020f96c 6cfaaf00 clr!_CorExeMainInternal+0x19f
0020f9a4 6e1e55ab clr!_CorExeMain+0x4e
0020f9b0 6e187f16 mscoreei!_CorExeMain+0x38
0020f9c0 6e184de3 mscoree!ShellShim__CorExeMain+0x99
0020f9c8 77563833 mscoree!_CorExeMain_Exported+0x8
0020f9d4 77b7a9bd kernel32!BaseThreadInitThunk+0xe
0020fa14 00000000 ntdll!_RtlUserThreadStart+0x23
```

```
0:000> !IP2MD 0x2b1515
MethodDesc:   000e934c
Method Name:  LINQPad.Program.Go(...)
Class:        002c05dc
MethodTable:  000e954c
mdToken:      06000272
Module:       000e2e9c
IsJitted:     yes
CodeAddr:     002b05c0
Transparency: Critical

0:000> !DumpModule 000e2e9c
Name:       C:\LINQPad4\LINQPad.exe
Attributes: PEFile
Assembly:   00317418
[...]
```

Thread Stack Raw Data



WinDbg Commands

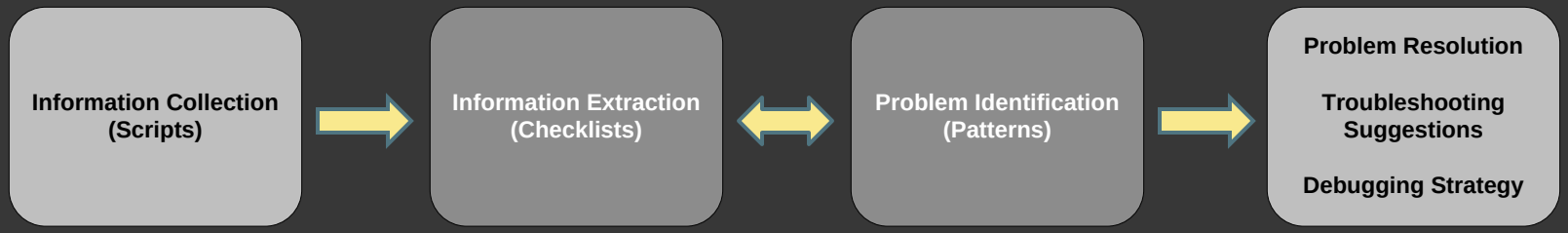
Get stack range:
!teb

Dump raw data:
dc / **dps** / **dpp** / **dpa** / **dpu**

Dump managed references:
!DumpStackObjects

Pattern-Driven Analysis

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



Checklist: http://www.dumpanalysis.org/windows-memory-analysis-checklist

Patterns: http://www.dumpanalysis.org/blog/index.php/crash-dump-analysis-patterns/

.NET Patterns: http://www.dumpanalysis.org/blog/index.php/2011/04/22/net-clr-managed-space-patterns/

Part 2: Practice Exercises

Links

- Memory Dumps:

Not available in preview version

- Exercise Transcripts:

Not available in preview version

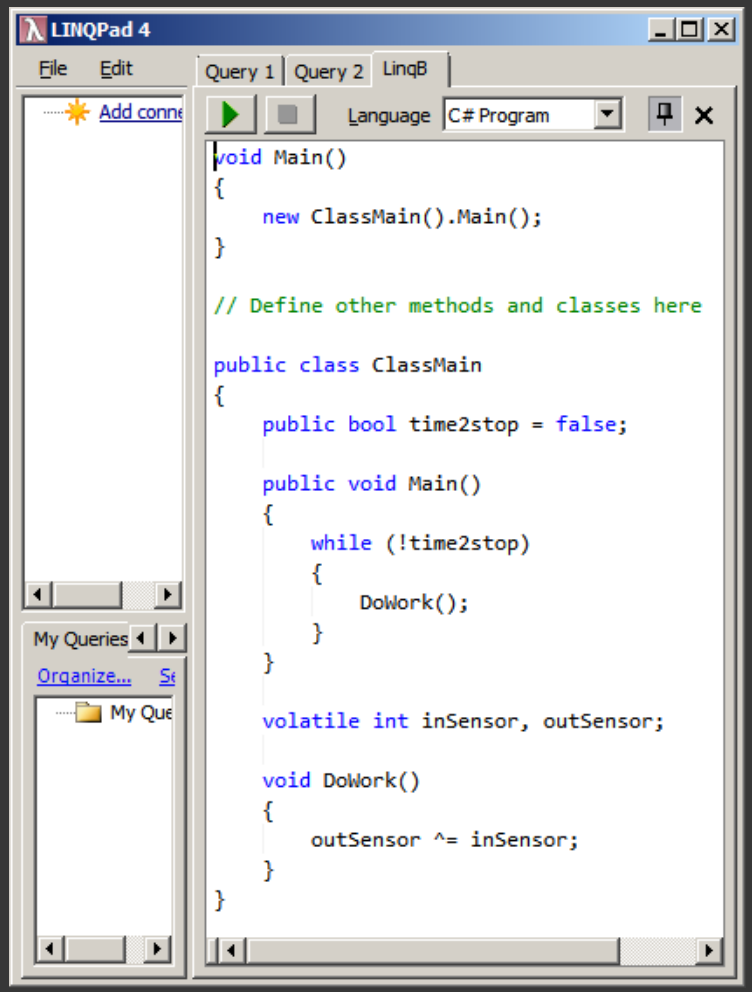
Exercise 0

- **Goal:** Install Debugging Tools for Windows and learn how to set up symbols correctly
- **Patterns:** Incorrect Stack Trace
- **Commands:** .symfix, .reload, k

Process Memory Dumps

Exercises PN1 - PN8

Modeling with LINQPad



http://www.linqpad.net/

Exercise PN1

- **Goal:** Learn how to load the correct .NET SOS WinDbg extension and analyze managed space
- **Patterns:** Stack Trace Collection; CLR Thread; Version-Specific Extension; Managed Code Exception; Managed Stack Trace
- **Commands:** .logopen, ~*k, !analyze -v, !pe, ~*e, lmv, .chain, .unload, .load, .logclose

Exercise PN2

- **Goal:** Compare CLR 2 with CLR 4. Manual stack reconstruction (*Advanced*).
- **Patterns:** Stack Trace Collection; CLR Thread; Version-Specific Extension; Managed Code Exception; Managed Stack Trace; Truncated Stack Trace; Incorrect Stack Trace (*Advanced*)
- **Commands:** .kframes, kc, dps, k L=<>

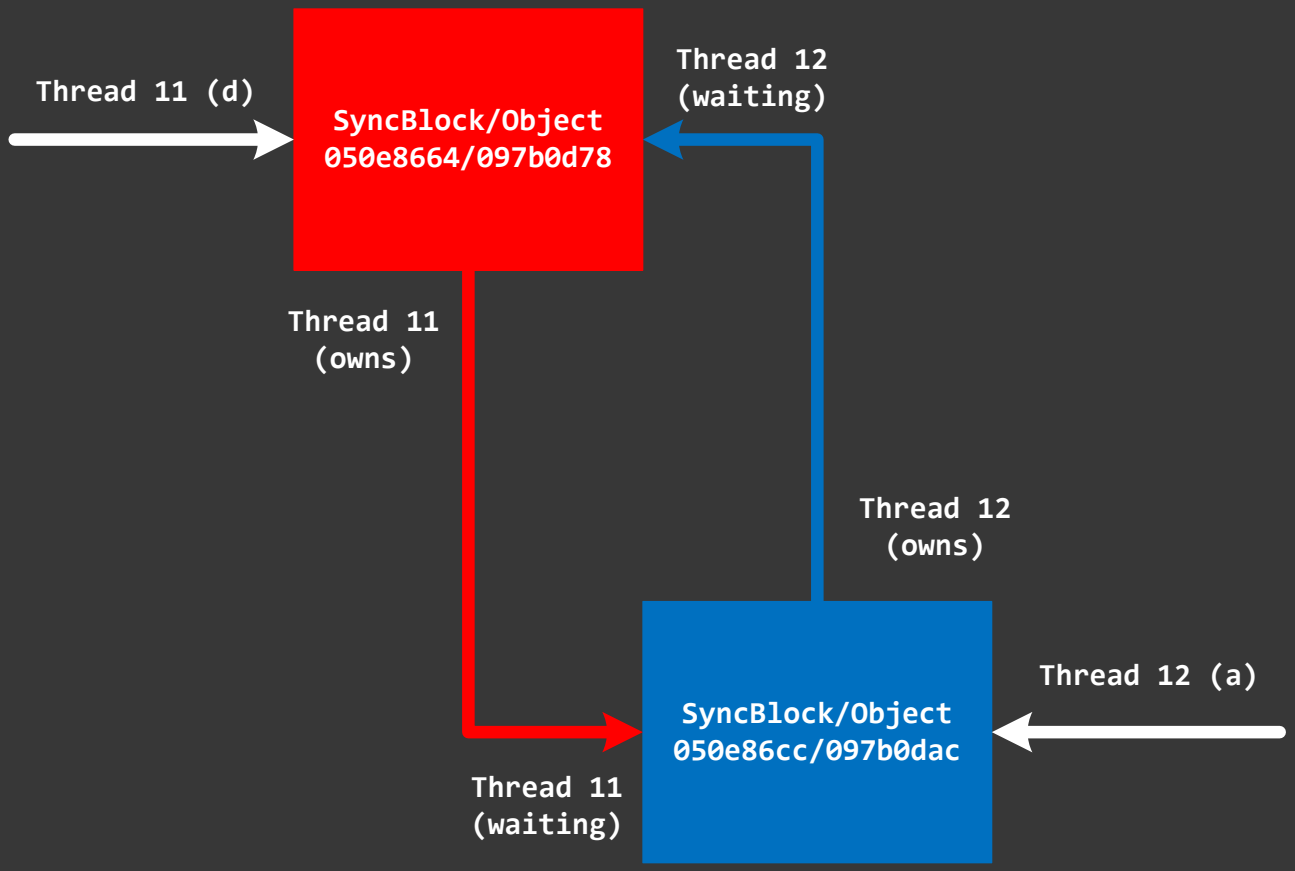
Exercise PN3

- **Goal:** Learn how to find problem assemblies, modules, classes and methods, disassemble code, analyze CPU spikes
- **Patterns:** Stack Trace Collection; CLR Thread; Version-Specific Extension; Duplicate Extension; JIT Code; Spiking Thread; Annotated Disassembly
- **Commands:** !analyze -v -hang, !CLRStack, !IP2MD, !runaway, ~<>s, ~<>k, !U, !DumpMD, !DumpClass, !DumpMT, !DumpModule, !DumpAssembly, !DumpDomain

Exercise PN4

- **Goal:** Learn how to recognize and analyze deadlocks using SOS(EX), execution residue, handled exceptions, dump object references
- **Patterns:** CLR Thread; Special Thread; Blocked Thread; Annotated Disassembly; Deadlock; Caller-n-Callee; Execution Residue; Handled Exception
- **Commands:** !Threads -special, kL, !syncblk, !DumpObject, ub, !U, dp, !dlk, !DumpStack, !DumpStackObjects, !teb, dps

Deadlock



Exercise PN5

- **Goal:** Learn how to analyze multiple managed exceptions, diagnose heap and handle leaks
- **Patterns:** CLR Thread; Managed Stack Trace Collection; Managed Code Exception; Handle Leak; Multiple Exceptions; Nested Exceptions; Annotated Disassembly; Execution Residue; Exception Thread; Hidden Exception; NULL Pointer
- **Commands:** kv, .cxr, !DumpHeap, ?, !eeheap, !GCHandles, !FinalizeQueue

Exercise PN6

- **Goal:** Learn how to recognize and analyze heap corruption
- **Patterns:** CLR Thread; Exception Thread; Invalid Pointer; Managed Heap Corruption; Execution Residue
- **Commands:** !VerifyHeap, dc

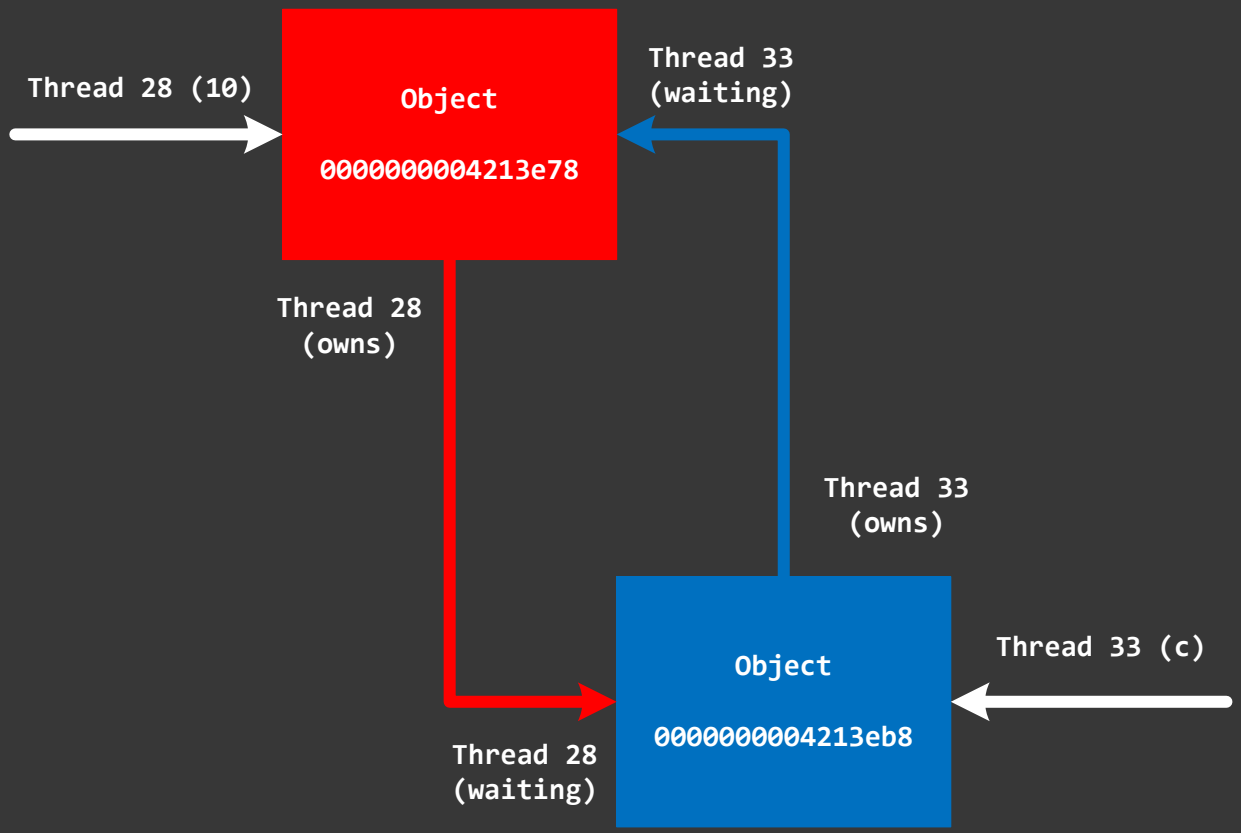
Exercise PN7 (x64)

- **Goal:** Learn how to find problem assemblies, modules, classes and methods, disassemble code, analyze CPU spikes
- **Patterns:** Stack Trace Collection; CLR Thread; Version-Specific Extension; Duplicate Extension; JIT Code; Spiking Thread; Annotated Disassembly
- **Commands:** !analyze -v -hang, !CLRStack, !IP2MD, !runaway, ~<>s, ~<>k, !U, !DumpMD, !DumpClass, !DumpMT, !DumpModule, !DumpAssembly, !DumpDomain

Exercise PN8 (x64)

- **Goal:** Learn how to recognize and analyze deadlocks using SOS(EX), execution residue, handled exceptions, dump object references
- **Patterns:** CLR Thread; Special Thread; Blocked Thread; Deadlock; Execution Residue; Handled Exception
- **Commands:** !Threads -special, kL, !syncblk, !DumpObject, ub, dp, !dlk, !DumpStack, !DumpStackObjects, !teb, dps

Deadlock (x64)



Pattern Links

CLR Thread
Managed Code Exception
Nested Exceptions
Mixed Exception
Memory Leak
JIT Code
Managed Stack Trace
Multiple Exceptions
Version-Specific Extension
Caller-n-Callee

Deadlock
Duplicate Extension
Stack Trace Collection
Dynamic Memory Corruption
Special Thread
Execution Residue
Handled Exception
Annotated Disassembly
Technology-Specific Subtrace

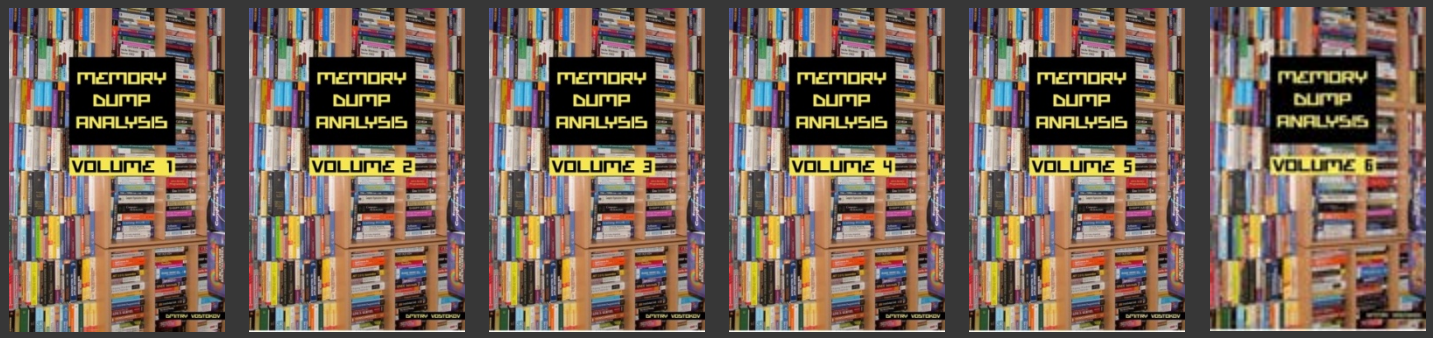
Incorrect Stack Trace Paged Out Data
NULL Pointer Handle Leak
Truncated Stack Trace Special Process
Spiking Thread Hidden Exception

SOS Checklist

- CLR module and SOS extension versions (lmv and .chain)
- Managed exceptions (~*e !pe -nested)
- Managed threads (!Threads -special)
- Managed stack traces (~*e !CLRStack)
- Managed execution residue (~*e !DumpStackObjects)
- Managed heap (!VerifyHeap, !DumpHeap -stat and !eeheap -gc)
- GC handles (!GCHandles)
- Finalizer queue (!FinalizeQueue)
- Sync blocks (!syncblk)

Resources

- WinDbg Help / WinDbg.org (quick links)
- DumpAnalysis.org
- C# 4.0 in a Nutshell
- Advanced .NET Debugging
- Debugging Microsoft .NET 2.0 Applications
- Shared Source CLI 2.0 Internals
- Accelerated Windows Memory Dump Analysis, 2nd edition
- Memory Dump Analysis Anthology, volumes 1 - 6



Q&A

Please send your feedback using the contact form on PatternDiagnostics.com

Thank you for attendance!