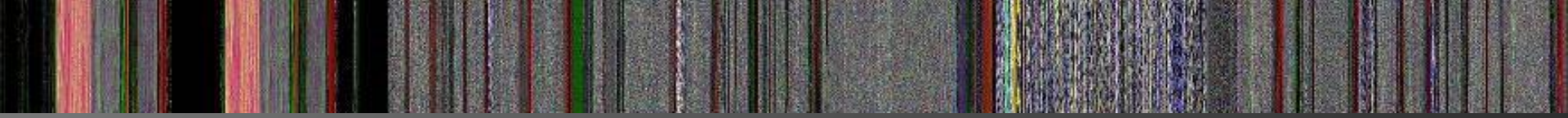


Physical Memory Analysis

Fundamentals

Anniversary Edition

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

To Be Discussed Later

We use these boxes to introduce useful vocabulary to be discussed in later slides

Working knowledge of:

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

Agenda (Summary)

- Basics
- Patterns
- Exercise
- Guide

Agenda (Basics)

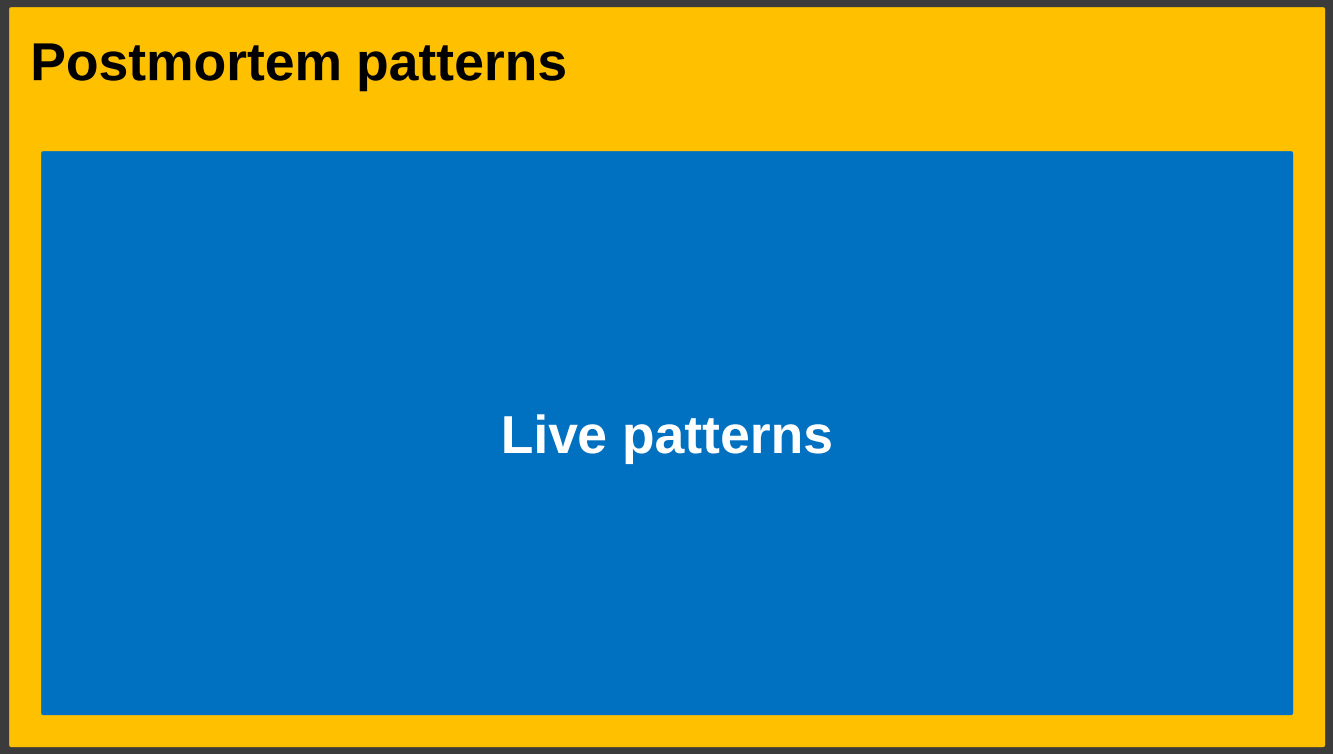
- Dump generation
- Memory spaces
- Major challenges
- Common commands

Platform: Windows

The pattern-oriented approach is applicable to other OS through different memory analysis pattern implementations

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

Memory Analysis

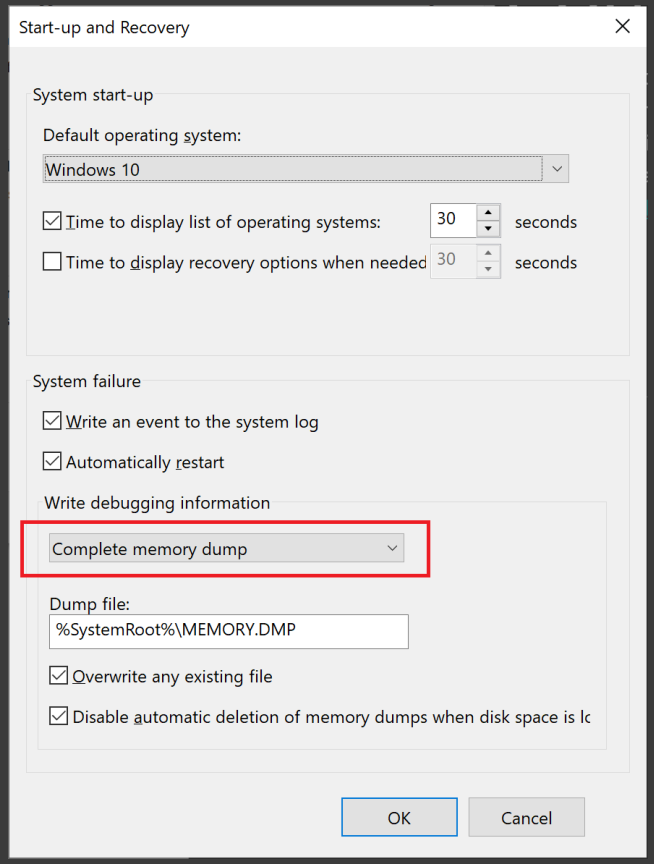


Dump Configuration

To Be Discussed Later

Truncated Dump pattern
Manual Dump pattern

- Control Panel \ System and Security \ System \ Advanced system settings \ Advanced \ Start-up and Recovery
- Page file size should be greater than the amount of physical memory by a few MB
- Configuration for Server Core, small system partitions, or virtual disk systems



Troubleshooting note:

HKLM \ SYSTEM \ CurrentControlSet \ Control \ CrashControl
CrashDumpEnabled = 1 (DWORD)

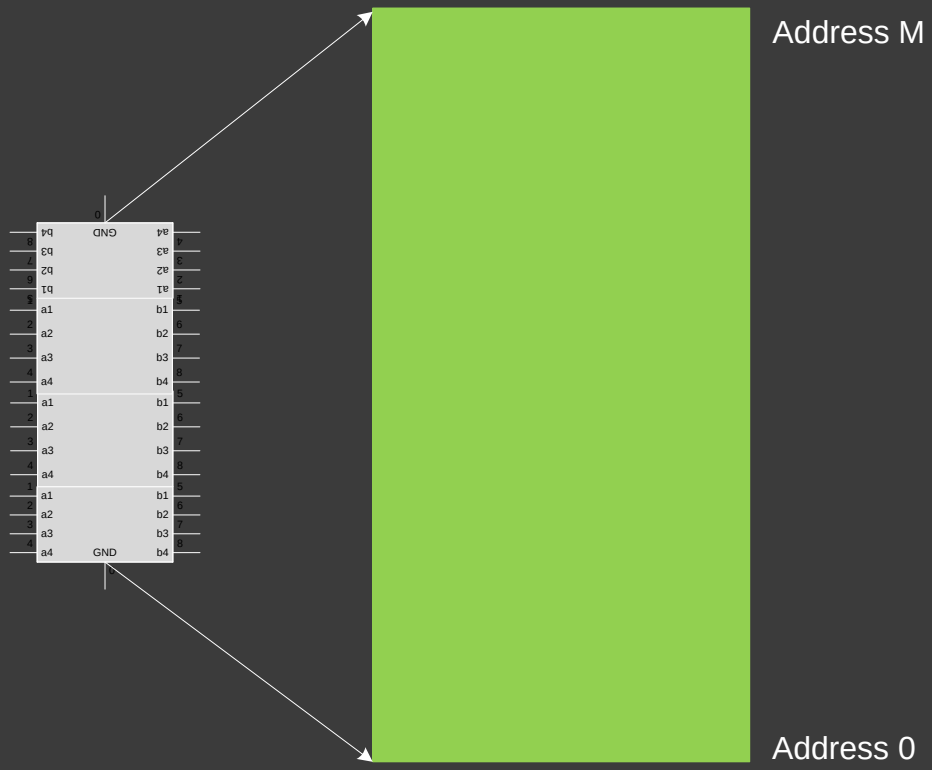
No complete memory dumps saved in older systems

Page file preservation

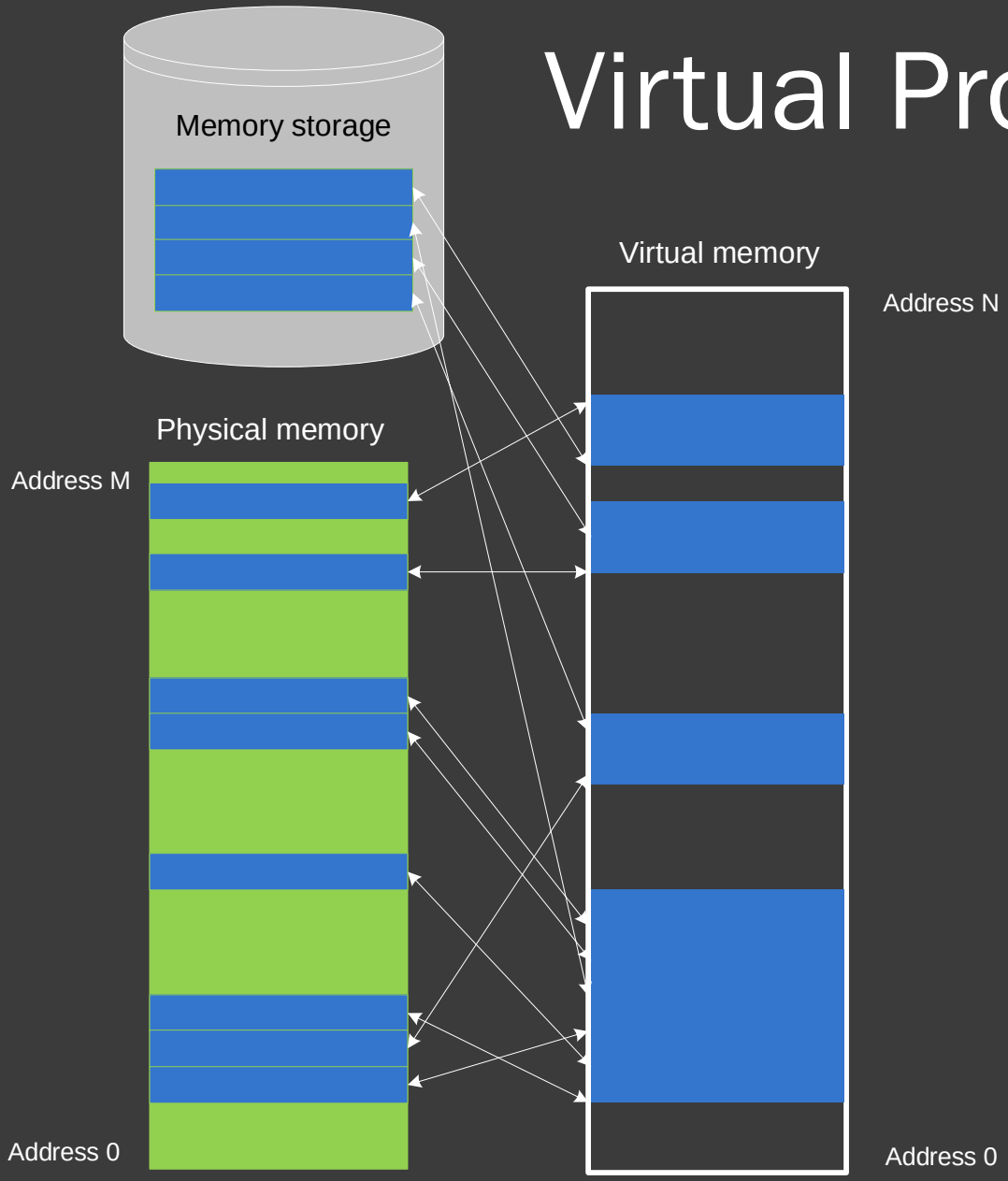
Dump and Memory Acquisition

- General
- Killing a system process like csrss.exe (-W8.1)
- LiveKd (options for more consistency)
- Live debugging (.dump)
- Memory forensic tools

Physical Memory



Virtual Process Memory



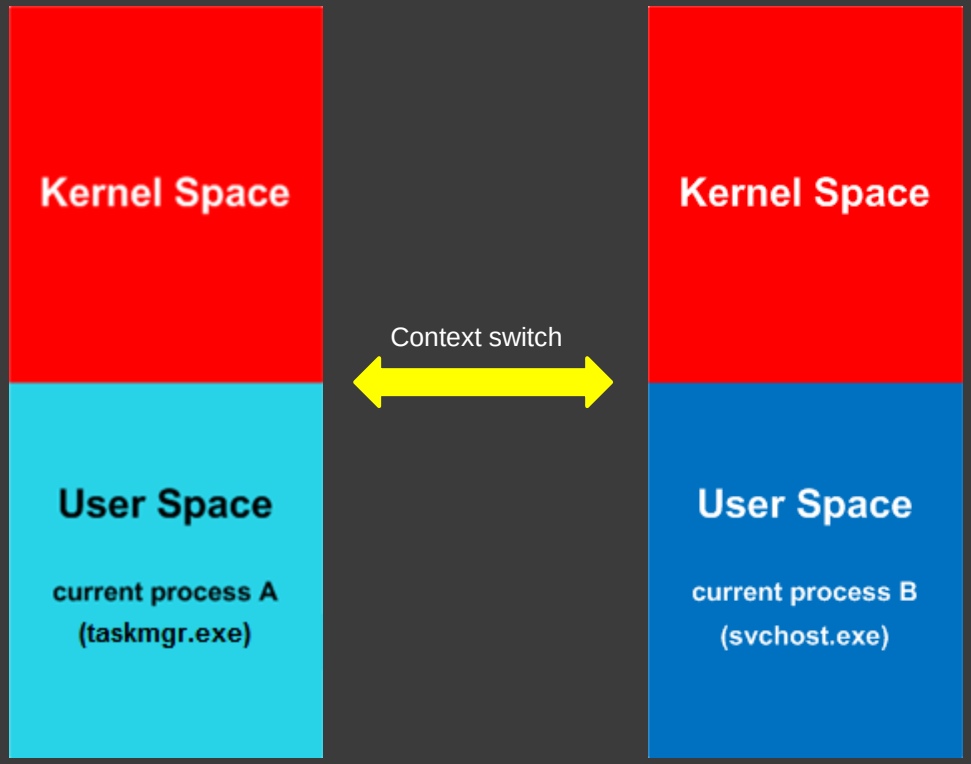
Memory Spaces

- 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



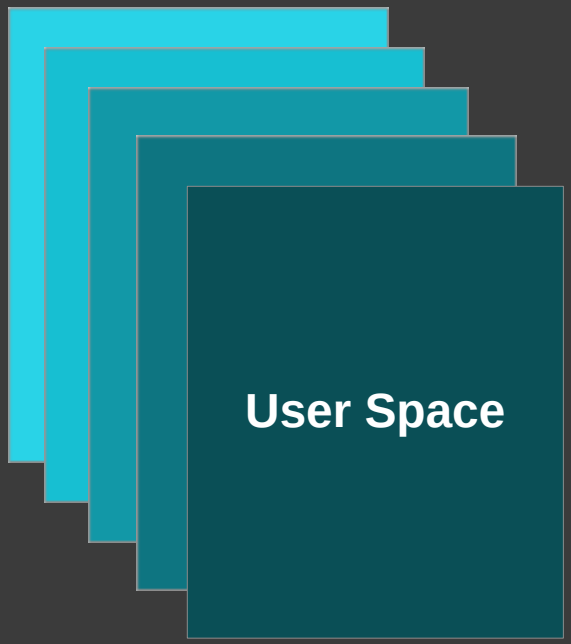
Major Challenges

To Be Discussed Later

WinDbg extension command to dump all stack traces:

!process 0 3f

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



To Be Discussed Later

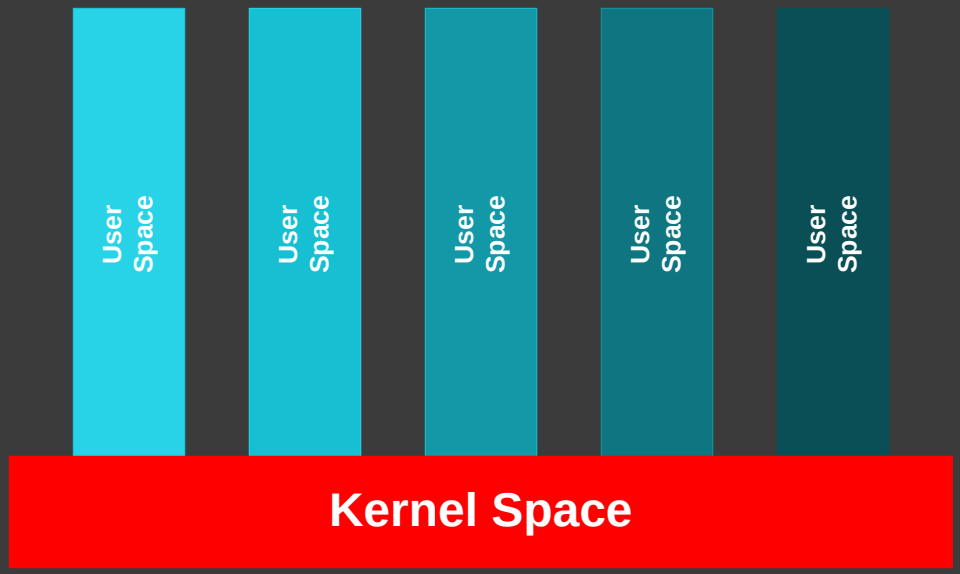
Wait Chain patterns

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

To Be Discussed Later

x86 stack trace from WOW64 process:

.thread /w

- **.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
...............................
```

Agenda (Patterns)

- Pattern-oriented analysis
- Pattern classification
- Pattern examples
- Common mistakes

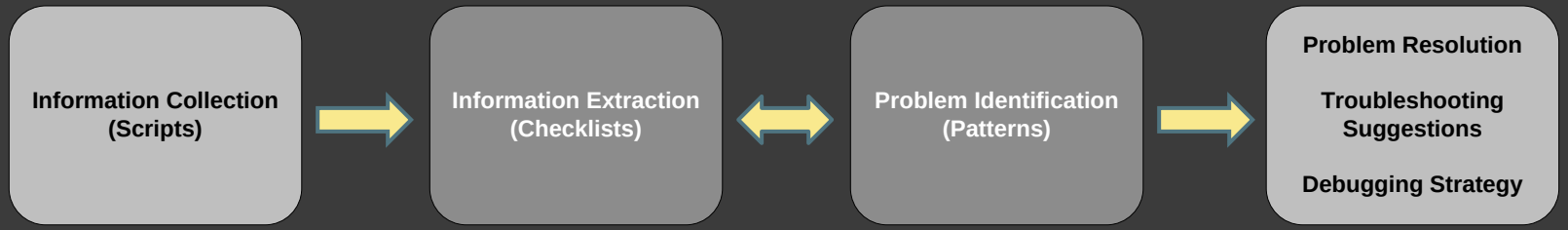
Pattern-Oriented Diagnostic Analysis

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

Diagnostic Problem: a set of indicators (symptoms, signs) describing a problem.

Diagnostic Analysis Pattern: a common recurrent analysis technique and method of diagnostic pattern identification in a specific context.

Diagnostics Pattern Language: common names of diagnostic and diagnostic analysis patterns. The same language for any operating system: Windows, Mac OS X, Linux, ...



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

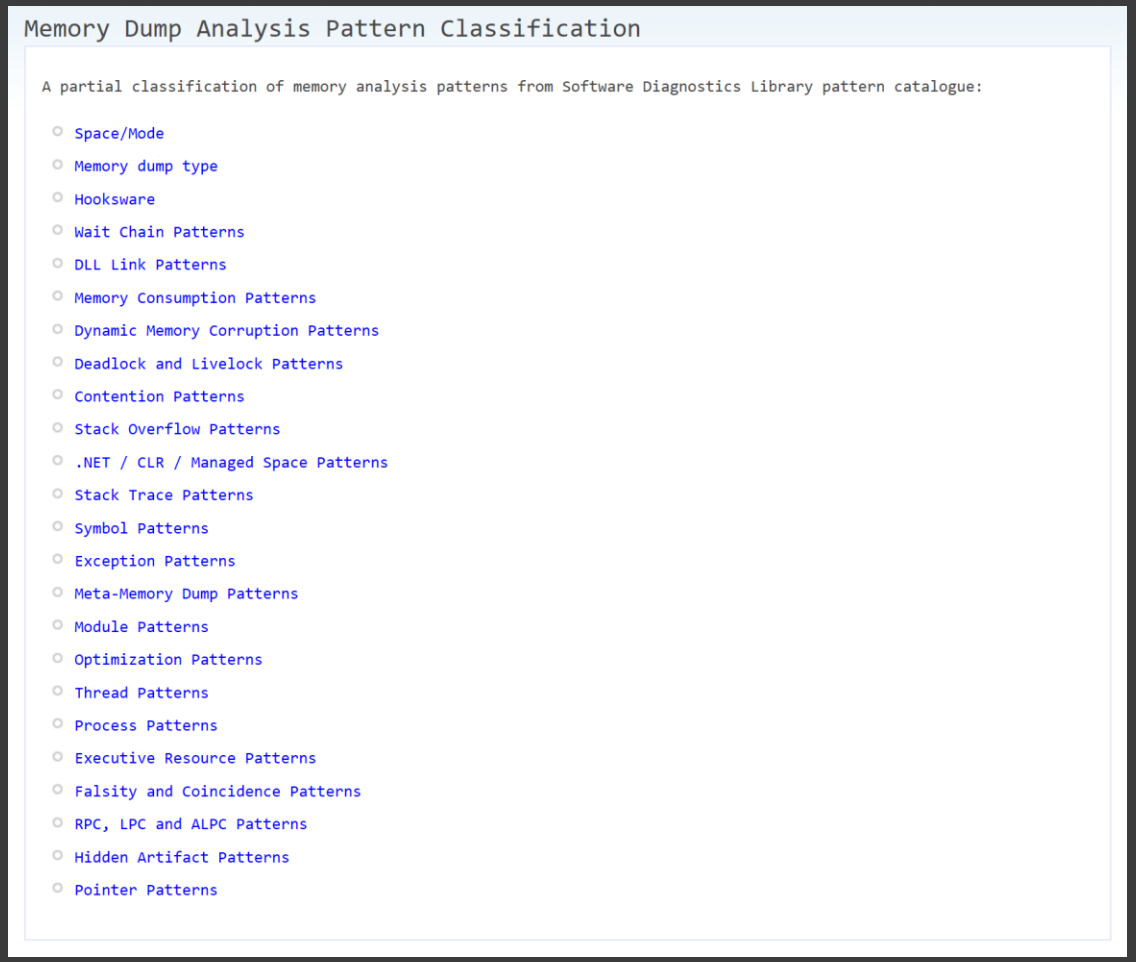
Patterns: http://www.dumpanalysis.org/blog/index.php/crash-dump-analysis-patterns/

Pattern Classes

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

Pattern Classification

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



Memory Dump Analysis Pattern Classification

A partial classification of memory analysis patterns from Software Diagnostics Library pattern catalogue:

- Space/Mode
- Memory dump type
- Hooksware
- Wait Chain Patterns
- DLL Link Patterns
- Memory Consumption Patterns
- Dynamic Memory Corruption Patterns
- Deadlock and Livelock Patterns
- Contention Patterns
- Stack Overflow Patterns
- .NET / CLR / Managed Space Patterns
- Stack Trace Patterns
- Symbol Patterns
- Exception Patterns
- Meta-Memory Dump Patterns
- Module Patterns
- Optimization Patterns
- Thread Patterns
- Process Patterns
- Executive Resource Patterns
- Falsity and Coincidence Patterns
- RPC, LPC and ALPC Patterns
- Hidden Artifact Patterns
- Pointer Patterns

Example: Blocked Thread

```
THREAD ffff930ac49d0080  Cid 1ffc.109c  Teb: 0000003c7ecd1000 Win32Thread: ffff930ac62b44b0 WAIT: (WrUserRequest) UserMode Non-Alertable
           ffff930ac621fc80  QueueObject
      Not impersonating
      DeviceMap                 ffffcf8978c103a0
      Owning Process            ffff930ac55de080       Image:         ApplicationA.exe
      Attached Process          N/A            Image:         N/A
      Wait Start TickCount      49071          Ticks: 976 (0:00:00:15.250)
      Context Switch Count      548            IdealProcessor: 1
      UserTime                  00:00:00.031
      KernelTime                00:00:00.015
      Win32 Start Address ApplicationA (0x00007ff64ed42c2c)
      Stack Init ffffef8637e84c90 Current ffffef8637e84490
      Base ffffef8637e85000 Limit ffffef8637e7f000 Call 0000000000000000
      Priority 10 BasePriority 8 PriorityDecrement 0 IoPriority 2 PagePriority 5
      Child-SP          RetAddr           Call Site
      ffffef86`37e844d0 fffff800`1151507d nt!KiSwapContext+0x76
      ffffef86`37e84610 fffff800`11513f04 nt!KiSwapThread+0xbfd
      ffffef86`37e846b0 fffff800`115136a5 nt!KiCommitThreadWait+0x144
      ffffef86`37e84750 fffff800`114dea6e nt!KeWaitForSingleObject+0x255
      ffffef86`37e84830 ffffdfa3`9b92962e nt!KeWaitForMultipleObjects+0x54e
      ffffef86`37e84940 ffffdfa3`9b929c55 win32kfull!xxxRealSleepThread+0x2be
      ffffef86`37e84a70 ffffdfa3`9b91c225 win32kfull!xxxSleepThread2+0xb5
      ffffef86`37e84ac0 fffff800`115d3c15 win32kfull!NtUserWaitMessage+0x65
      ffffef86`37e84b00 00007ffc`3fb71224 nt!KiSystemServiceCopyEnd+0x25 (TrapFrame @ ffffef86`37e84b00)
      0000003c`7f3ff748 00007ffc`4083bf90 win32u!NtUserWaitMessage+0x14
      0000003c`7f3ff750 00007ffc`4083bcff USER32!DialogBox2+0x260
      0000003c`7f3ff7f0 00007ffc`40882f99 USER32!InternalDialogBox+0x11b
      0000003c`7f3ff850 00007ffc`408819d5 USER32!SoftModalMessageBox+0x7e9
      0000003c`7f3ff9a0 00007ffc`40882712 USER32!MessageBoxWorker+0x319
      0000003c`7f3ffb50 00007ffc`4088279e USER32!MessageBoxTimeoutW+0x192
>>>   0000003c`7f3ffc50 00007ffc`3d2b23ff USER32!MessageBoxW+0x4e
      0000003c`7f3ffc90 00007ff6`4ed41299 apphelp!MbHook_MessageBoxW+0x2f
      0000003c`7f3ffce0 00007ff6`4ed42c89 ApplicationA+0x1299
      0000003c`7f3ffd10 00007ffc`41937bd4 ApplicationA+0x2c89
      0000003c`7f3ffd40 00007ffc`425cce51 KERNEL32!BaseThreadInitThunk+0x14
      0000003c`7f3ffd70 00000000`00000000 ntdll!RtlUserThreadStart+0x21
```

To Be Discussed Later

Complete Dump Analysis Exercise

Example: Wait Chain

To Be Discussed Later

Complete Dump Analysis Exercise

```
 THREAD ffff930ac2a850c0  Cid 1da4.0aa0  Teb: 0000005d75b4d000 Win32Thread: 0000000000000000 WAIT: (UserRequest) UserMode Non-Alertable
>>>         ffff930ac4f05ad0  Mutant - owning thread ffff930ac230f080
        Not impersonating
        DeviceMap                 ffffcf8978c103a0
        Owning Process            ffff930ac236e080       Image:         ApplicationC.exe
        Attached Process          N/A            Image:         N/A
        Wait Start TickCount      42255          Ticks: 7792 (0:00:02:01.750)
        Context Switch Count      6              IdealProcessor: 0
        UserTime                  00:00:00.000
        KernelTime                00:00:00.000
        Win32 Start Address ApplicationC (0x00007ff7b8f62ce0)
        Stack Init ffffef8637ebcc90 Current ffffef8637ebc6e0
        Base ffffef8637ebd000 Limit ffffef8637eb7000 Call 0000000000000000
        Priority 9 BasePriority 8 PriorityDecrement 0 IoPriority 2 PagePriority 5
        Child-SP          RetAddr           Call Site
        ffffef86`37ebc720 fffff800`1151507d nt!KiSwapContext+0x76
        ffffef86`37ebc860 fffff800`11513f04 nt!KiSwapThread+0xbfd
        ffffef86`37ebc900 fffff800`115136a5 nt!KiCommitThreadWait+0x144
        ffffef86`37ebc9a0 fffff800`11abd2bb nt!KeWaitForSingleObject+0x255
        ffffef86`37ebca80 fffff800`115d3c15 nt!NtWaitForSingleObject+0x10b
        ffffef86`37ebcb00 00007ffc`425fc0f4 nt!KiSystemServiceCopyEnd+0x25 (TrapFrame @ ffffef86`37ebcb00)
        0000005d`763ffdb8 00007ffc`3f8a8b03 ntdll!NtWaitForSingleObject+0x14
        0000005d`763ffdc0 00007ff7`b8f6136c KERNELBASE!WaitForSingleObjectEx+0x93
        0000005d`763ffe60 00007ff7`b8f62d3d ApplicationC+0x136c
        0000005d`763ffea0 00007ffc`41937bd4 ApplicationC+0x2d3d
        0000005d`763ffed0 00007ffc`425cce51 KERNEL32!BaseThreadInitThunk+0x14
        0000005d`763fff00 00000000`00000000 ntdll!RtlUserThreadStart+0x21
```

Example: Consumption

To Be Discussed Later

Complete Dump Analysis Exercise

```
0: kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS ffff930abce80040
    SessionId: none  Cid: 0004    Peb: 00000000  ParentCid: 0000
    DirBase: 001ad002  ObjectTable: ffffcf896e606580  HandleCount: 3423.
    Image: System

PROCESS ffff930abcee2080
    SessionId: none  Cid: 0058    Peb: 00000000  ParentCid: 0004
    DirBase: 00222002  ObjectTable: ffffcf896e60ca80  HandleCount:   0.
    Image: Registry

PROCESS ffff930ac005a040
    SessionId: none  Cid: 0144    Peb: 8ed0d35000  ParentCid: 0004
    DirBase: 1006ed002  ObjectTable: ffffcf896ec2ab00  HandleCount:  53.
    Image: smss.exe

PROCESS ffff930ac015f080
    SessionId: 0  Cid: 01a0    Peb: e57797b000  ParentCid: 0198
    DirBase: 1056b0002  ObjectTable: ffffcf896ec2b7c0  HandleCount: 512.
    Image: csrss.exe

[...]

PROCESS ffff930ac2be5080
    SessionId: 1  Cid: 0c58    Peb: 56ece5a000  ParentCid: 1600
>>> DirBase: 86166002  ObjectTable: ffffcf897a694bc0  HandleCount: 20055.
    Image: ApplicationE.exe

[...]
```

Example: Corruption

To Be Discussed Later

Complete Dump Analysis Exercise

```
THREAD ffff930ac4dda500  Cid 1df8.0714  Teb: 0000000000712000 Win32Thread: 0000000000000000 WAIT: (UserRequest) UserMode Alertable
           ffff930ac268bb60  NotificationEvent
           ffff930ac61f7080  ProcessObject
       Not impersonating
       DeviceMap                 ffffcf8978c103a0
       Owning Process            ffff930ac63230c0       Image:         ApplicationD.exe
       Attached Process          N/A            Image:         N/A
       Wait Start TickCount      42613          Ticks: 7434 (0:00:01:56.156)
       Context Switch Count      16             IdealProcessor: 0
       UserTime                  00:00:00.000
       KernelTime                00:00:00.000
       Win32 Start Address ApplicationD (0x00007ff625ec1318)
       Stack Init ffffef8637f6bc90 Current ffffef8637f6af30
       Base ffffef8637f6c000 Limit ffffef8637f66000 Call 0000000000000000
       Priority 9 BasePriority 8 PriorityDecrement 0 IoPriority 2 PagePriority 5
       Child-SP          RetAddr           Call Site
       ffffef86`37f6af70 fffff800`1151507d nt!KiSwapContext+0x76
       ffffef86`37f6b0b0 fffff800`11513f04 nt!KiSwapThread+0xbfd
       ffffef86`37f6b150 fffff800`114de7a7 nt!KiCommitThreadWait+0x144
       ffffef86`37f6b1f0 fffff800`11a90659 nt!KeWaitForMultipleObjects+0x287
       ffffef86`37f6b300 fffff800`11a90375 nt!ObWaitForMultipleObjects+0x2a9
       ffffef86`37f6b800 fffff800`115d3c15 nt!NtWaitForMultipleObjects+0x105
       ffffef86`37f6ba90 00007ffc`425fcbc4 nt!KiSystemServiceCopyEnd+0x25 (TrapFrame @ ffffef86`37f6bb00)
       [...]
       00000000`00f9e7a0 00007ffc`425c9fc3 ntdll!RtlDispatchException+0x219
       00000000`00f9eeb0 00007ffc`42659229 ntdll!RtlRaiseException+0x153
       00000000`00f9f6a0 00007ffc`426591f3 ntdll!RtlReportFatalFailure+0x9
       00000000`00f9f6f0 00007ffc`426615e2 ntdll!RtlReportCriticalFailure+0x97
       00000000`00f9f7e0 00007ffc`426618ea ntdll!RtlpHeapHandleError+0x12
       00000000`00f9f810 00007ffc`4266a8a9 ntdll!RtlpHpHeapHandleError+0x7a
       00000000`00f9f840 00007ffc`425a080d ntdll!RtlpLogHeapFailure+0x45
       00000000`00f9f870 00007ffc`4259fb91 ntdll!RtlpFreeHeapInternal+0x80d
       00000000`00f9f920 00007ff6`25ec1274 ntdll!RtlFreeHeap+0x51
       00000000`00f9f960 00007ff6`25ec10c3 ApplicationD+0x1274
       [...]
```

Example: Special Process

```
0: kd> !vm

[...]

     564 svchost.exe                  6264 Kb        1980 Kb        0 Kb
     8c8 svchost.exe                  6060 Kb        2692 Kb        0 Kb
     a74 spoolsv.exe                  5868 Kb        1988 Kb        0 Kb
     be4 svchost.exe                  5700 Kb        2068 Kb        0 Kb
    10ac svchost.exe                  5672 Kb        2232 Kb        0 Kb
>>>  bd8 WerFault.exe                 5384 Kb        4944 Kb        0 Kb
    1128 svchost.exe                  4968 Kb        2264 Kb        0 Kb
     274 services.exe                 4916 Kb         356 Kb        0 Kb
     c28 svchost.exe                  4860 Kb        2260 Kb        0 Kb
      b0 cmd.exe                      4692 Kb         356 Kb        0 Kb
    1290 browser_broker.exe           4520 Kb        2564 Kb        0 Kb
    1fbc MicrosoftEdgeSH.exe          4480 Kb        5052 Kb        0 Kb
     6dc svchost.exe                  4456 Kb        1936 Kb        0 Kb
     84c svchost.exe                  4292 Kb        1952 Kb        0 Kb
     e5c NisSrv.exe                   4288 Kb        2000 Kb        0 Kb
    1c44 svchost.exe                  4276 Kb        1984 Kb        0 Kb
     c5c svchost.exe                  4164 Kb        1980 Kb        0 Kb
    12f4 backgroundTaskHost.exe       4060 Kb        2812 Kb        0 Kb
     e94 dllhost.exe                  4012 Kb        1976 Kb        0 Kb
    16b8 svchost.exe                  3980 Kb        2692 Kb        0 Kb
    1ce8 ctfmon.exe                   3728 Kb        3512 Kb        0 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)

```
.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: I did not make 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 10

We use the following command:
C:\Tools>notmyfault64.exe /crash

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 basic analysis 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

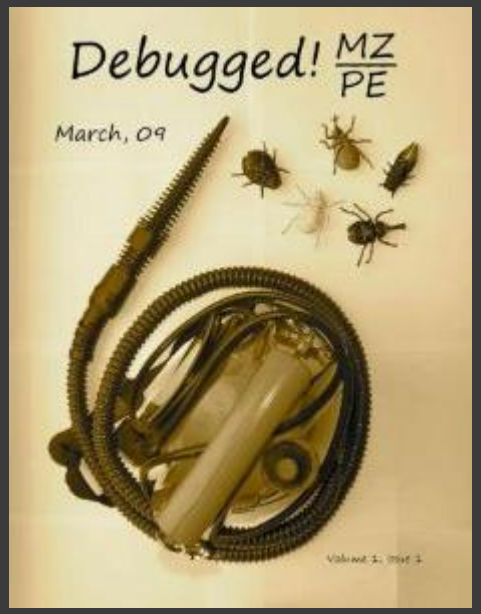
No System Dumps
Message Box
Inconsistent Dump
Wait Chain (critical sections)
Wait Chain (process objects)
Special Process
Historical Information
Stack Trace Collection
Insufficient Memory (handle leak)
Main Thread
Suspended Thread
Pleiades
Dual Stack Trace

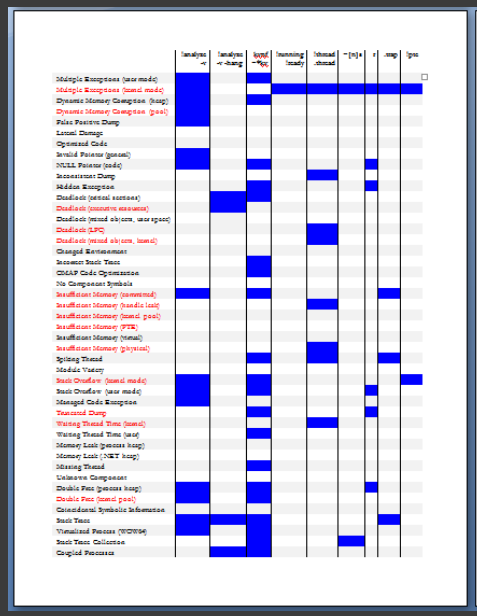
Case Studies

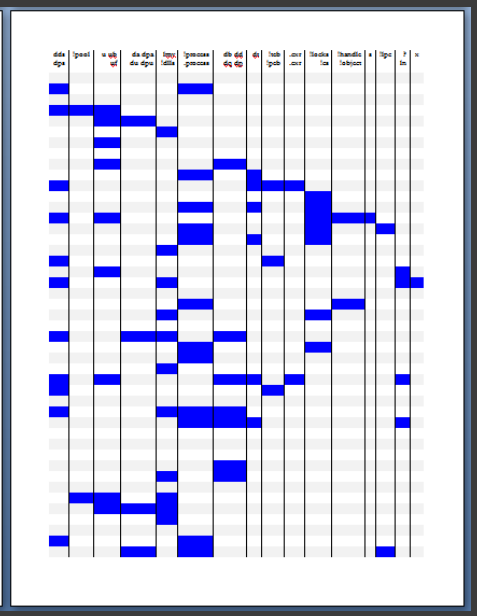
17 pattern interaction case studies using complete memory dumps:

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

WinDbg Command Map







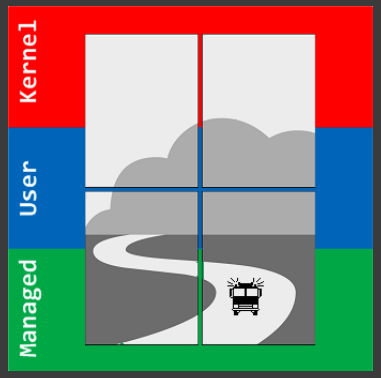
[Pattern <-> WinDbg command]

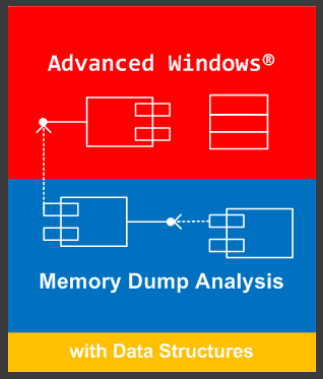
Reference Resources

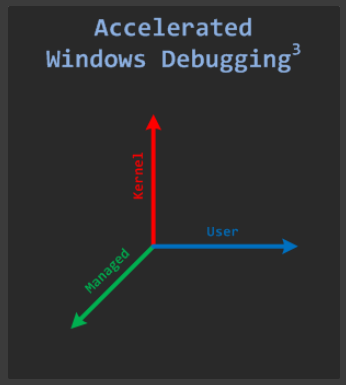
- WinDbg Help / WinDbg.org (quick links)
- DumpAnalysis.org / SoftwareDiagnostics.Institute / PatternDiagnostics.com
- Debugging.TV / YouTube.com/DebuggingTV / YouTube.com/PatternDiagnostics
- Encyclopedia of Crash Dump Analysis Patterns, 2nd edition
- Memory Dump Analysis Anthology (Volume 13 is forthcoming in 2020)

Training Resources

- Accelerated Windows Memory Dump Analysis, 4th + 5th editions
- Advanced Windows Memory Dump Analysis with Data Structures, 3rd edition
- Accelerated Windows Malware Analysis with Memory Dumps, 2nd edition
- Accelerated Windows Debugging^3, 2nd edition







Q&A

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

Thank you for attendance!

Facebook LinkedIn Twitter