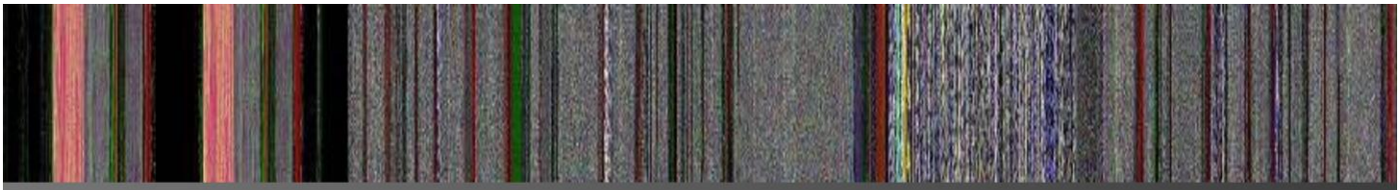


Defect
Detect

Windows Memory Dump Analysis Extended

Extensions
Database and Event Stream Processing
Visualization

Revised Edition

Dmitry Vostokov
Software Diagnostics Services

Published by OpenTask, Republic of Ireland

OpenTask books and magazines are available through booksellers and distributors worldwide. For further information or comments, send requests to press@opentask.com.

A CIP catalog record for this book is available from the British Library.

ISBN-l3: 978-1-912636-68-6 (Paperback)

Revision 1.52 (August 2023)

Contents

About the Author 5

Introduction 7

Practice Exercises 17

- Exercise E0: Download, setup, and verify your WinDbg or Debugging Tools for Windows installation, or Docker Debugging Tools for Windows image 21
- Exercise ES1: Explore Patterns WinDbg Extension 35
- Exercise ES2: Explore MEX WinDbg Extension 41
- Exercise ES3: Explore DbgKit WinDbg Extension 63
- Exercise ES4: Explore win32kext WinDbg Extension 105
- Exercise ES5: Explore SwishDbgExt WinDbg Extension 112
- Exercise ES6: Explore 0cchext WinDbg Extension 130
- Exercise ES7: Explore pykd WinDbg Extension 140
- Exercise EW1: Writing WinDbg Extension (WdbgExts C API) 173
- Exercise EW2: Writing WinDbg Extension (DbgEng COM API) 188
- Exercise EW3: Writing WinDbg Extension (ExtExtension C++ API) 202
- Exercise EP1: Install Kafka Environment 218
- Exercise EP2: Connect WinDbg to Kafka 221
- Exercise ED1: Install MongoDB Environment 227
- Exercise ED2: Connect WinDbg to MongoDB 232
- Exercise EV1: Install Jupyter Notebook Environment 242
- Exercise EV2: Execution Residue Visualization 251

Conclusion 267

Exercise ES1: Explore Patterns WinDbg Extension

Goal: Explore the patterns[4] WinDbg extension

1. Launch WinDbg.

2. Open \EWMDA-Dumps\Complete\x64\MEMORY-W11.DMP.

3. We get the dump file loaded:

```
Microsoft (R) Windows Debugger Version 10.0.25136.1001 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\EWMDA-Dumps\Complete\x64\MEMORY-W11.DMP]
Kernel Bitmap Dump File: Full address space is available


************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
OK                                             C:\AWMDA-Dumps\Symbols
Symbol search path is: srv*;C:\AWMDA-Dumps\Symbols
Executable search path is:
Windows 10 Kernel Version 22000 MP (2 procs) Free x64
Product: WinNt, suite: TerminalServer SingleUserTS Personal
Edition build lab: 22000.1.amd64fre.co_release.210604-1628
Machine Name:
Kernel base = 0xfffff806`61e00000 PsLoadedModuleList = 0xfffff806`62a296b0
Debug session time: Sat Nov 13 23:17:16.607 2021 (UTC + 1:00)
System Uptime: 0 days 0:03:06.813
Loading Kernel Symbols
...............................................................
...............................................................
...............................................................
..
Loading User Symbols
..............................
Loading unloaded module list
........
For analysis of this file, run !analyze -v
nt!KeBugCheckEx:
fffff806`62215590 48894c2408      mov     qword ptr [rsp+8],rcx
ss:0018:ffffbe82`96f64670=000000000000000a
```

4. We load the x64 version of the *patterns* extension:

```
0: kd> .load C:\EWMDA-Dumps\patterns\x64\patterns
```

[4] The latest version can be downloaded from: https://www.patterndiagnostics.com/patterns-extension.

5. We can also check if it was loaded successfully:

```
0: kd> .chain
Extension DLL search Path:
[...]
Extension DLL chain:
    C:\EWMDA-Dumps\patterns\x64\patterns: image 2.0.0.0, API 0.1.0, built Mon Aug 15 22:01:39 2022
        [path: C:\EWMDA-Dumps\patterns\x64\patterns.dll]
    wdfkd: image 10.0.25136.1001, API 1.0.0,
        [path: C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2206.19001.0_x64__8wekyb3d8bbwe\amd64\winext\wdfkd.dll]
    MachOBinComposition: image 10.0.25136.1001, API 0.0.0,
        [path: C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2206.19001.0_x64__8wekyb3d8bbwe\amd64\winext\MachOBinComposition.dll]
    ELFBinComposition: image 10.0.25136.1001, API 0.0.0,
        [path: C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2206.19001.0_x64__8wekyb3d8bbwe\amd64\winext\ELFBinComposition.dll]
    dbghelp: image 10.0.25136.1001, API 10.0.6,
        [path: C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2206.19001.0_x64__8wekyb3d8bbwe\amd64\dbghelp.dll]
    exts: image 10.0.25136.1001, API 1.0.0,
        [path: C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2206.19001.0_x64__8wekyb3d8bbwe\amd64\WINXP\exts.dll]
    kext: image 10.0.25136.1001, API 1.0.0,
        [path: C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2206.19001.0_x64__8wekyb3d8bbwe\amd64\winext\kext.dll]
    kdexts: image 10.0.25136.1001, API 1.0.0,
        [path: C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2206.19001.0_x64__8wekyb3d8bbwe\amd64\winxp\kdexts.dll]
```

6. Since the extension is on top, we can use **!help** instead of **!patterns.help**:

```
0: kd> !help
Patterns Debugger Extension DLL (Version 2.0.0.0 [std]). Copyright © 2015-2022 Software Diagnostics
Services. All rights reserved.

Commands:
    lst              - Shows the current list of memory analysis pattern categories
    lst category     - Shows the current list of memory analysis patterns for the specified category
    sdl abbreviation - Opens a pattern description from Software Diagnostics Library
    chk              - Shows the current memory analysis checklist categories
    chk category     - Shows the current memory analysis checklist for the specified category
    eula             - Shows license terms
```

7. The extension supports DML hyperlinks when possible (for example, in the GUI debugger version such as WinDbg, so the console-based Docker environment doesn't have it). **!chk** command outputs Memory Analysis Checklist[5] categories:

```
0: kd> !chk
The following memory analysis checklist categories are available:

General [G]
Application [A]
System [S]
BSOD [B]
.NET [.]
```

8. In the DML version, we can click on a category and also use the Back link, but for non-DML environments, we need to specify a category for the **!chk** command:

```
0: kd> !chk S
Memory Analysis Checklist for System Hang:

Default analysis (!analyze -v -hang)
ERESOURCE contention (!locks)
Processes and virtual memory including session space (!vm 4)
Important services are present and not hanging (for example, terminal or IMA services for
Citrix environments)
```

[5] https://www.dumpanalysis.org/windows-memory-analysis-checklist

```
Pools (!poolused)
Waiting threads (!stacks)
Critical system queues (!exqueue f)
I/O (!irpfind)
The list of all thread stack traces (!process 0 3f)
LPC/ALPC chain for suspected threads (!lpc message or !alpc /m after search for "Waiting for
reply to LPC" or "Waiting for reply to ALPC" in !process 0 3f output)
Mutants (search for "Mutants - owning thread" in !process 0 3f output)
Critical sections for suspected processes (!cs -l -o -s)
Sessions, session processes (!session, !sprocess)
Processes (size, handle table size) (!process 0 0)
Running threads (!running)
Ready threads (!ready)
DPC queues (!dpcs)
The list of APCs (!apc)
Internal queued spinlocks (!qlocks)
Computer name (dS srv!srvcomputername)
File cache, VACB (!filecache)
File objects for blocked thread IRPs (!irp -> !fileobj)
Network (!ndiskd.miniports and !ndiskd.pktpools)
Disk (!scsikd.classext -> !scsikd.classext class_device 2)
Modules rdbss, mrxdav, mup, mrxsmb in stack traces
Functions Ntfs!Ntfs* and nt!Fs* in stack traces

Back
```

Note: The extension uses the currently published checklist at the time of this writing. It may be amended in the future, so please download the most recent version of the extension for your practical analysis needs.

9. The **!lst** command lists analysis pattern categories, and we can navigate back and forth in the DML version. In the non-DML version, we need to specify category as a parameter:

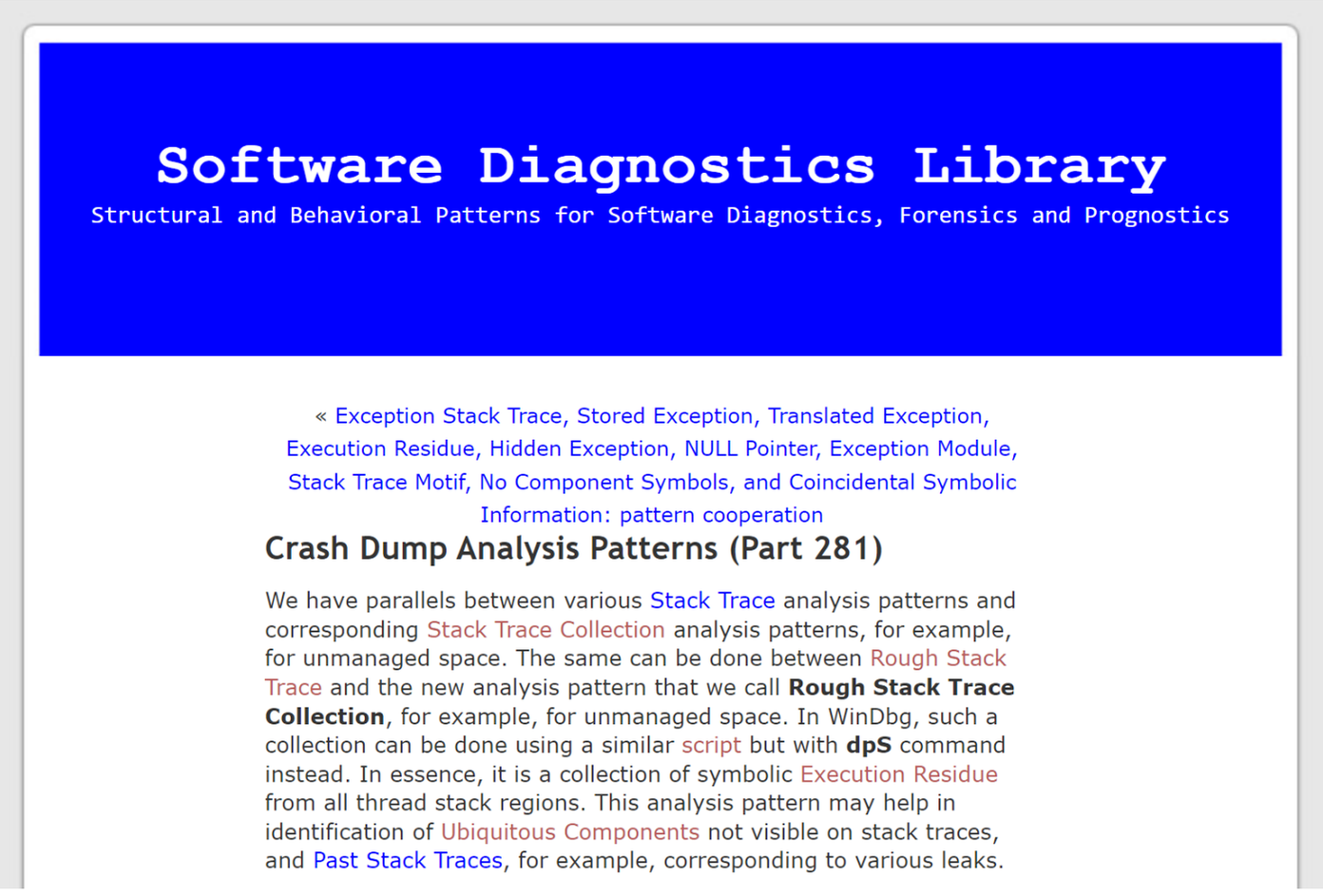
```
0: kd> !lst
Memory Analysis Pattern Categories:

Hooksware Patterns [H]
Wait Chain Patterns [W]
DLL Link Patterns [L]
Memory Consumption Patterns [M]
Dynamic Memory Corruption Patterns [C]
Deadlock and Livelock Patterns [D]
Contention Patterns [N]
Stack Overflow Patterns [O]
.NET / CLR / Managed Space Patterns [.]
Stack Trace Patterns [S]
Symbol Patterns [Y]
Exception Patterns [E]
Meta-Memory Dump Patterns [-]
Module Patterns [!]
Optimization Patterns [I]
Thread Patterns [T]
Process Patterns [P]
Executive Resource Patterns [X]
Falsity and Coincidence Patterns [F]
RPC, LPC and ALPC Patterns [R]
Hidden Artifact Patterns [A]
Pointer Patterns [*]
Frame Patterns [+]
```

```
CPU Consumption Patterns [^]
Malware Analysis Patterns [@]

0: kd> !lst S
Stack Trace Patterns:

Stack Trace [STTR]
Stack Trace Collection (unmanaged space) [STCU]
Special Stack Trace [SSTR]
Exception Stack Trace [ESTR]
Dual Stack Trace [DSTR]
Truncated Stack Trace [TSTR]
Managed Stack Trace [MSTR]
Incorrect Stack Trace [ISTR]
Stack Trace Set [STSE]
Stack Trace Collection (managed space) [STCM]
Stack Trace Collection (predicate) [STCP]
Empty Stack Trace [EMST]
Stack Trace Collection (I/O requests) [STCI]
Stack Trace Change [STCH]
First Fault Stack Trace [FFST]
Critical Stack Trace [CSTR]
RIP Stack Trace [RSTR]
Glued Stack Trace [GSTR]
Rough Stack Trace (unmanaged space) [RSTU]
Past Stack Trace [PSTR]
Stack Trace (I/O request) [STIO]
Stack Trace (file system filters) [STFS]
Stack Trace (database) [STDB]
Variable Subtrace [VASU]
Technology-Specific Subtrace (COM interface invocation) [TSCI]
Technology-Specific Subtrace (dynamic memory) [TSDM]
Technology-Specific Subtrace (JIT .NET code) [TSJN]
Technology-Specific Subtrace (COM client call) [TSCC]
Internal Stack Trace [INST]
Stack Trace Collection (CPUs) [STCC]
Stack Trace Surface [STSU]
Hidden Stack Trace [HSTR]
Constant Subtrace [COSU]
Stack Trace Signature [STSI]
Quotient Stack Trace [QSTR]
Module Stack Trace [MOST]
Coincidental Frames [COFR]
Least Common Frame [LCFR]
Foreign Module Frame [FMFR]
Unified Stack Trace [USTR]
Aggregated Frames [AGFR]
Stack Trace (I/O devices) [SIOD]
Stack Trace Motif [STMO]
Stack Trace Race [STRA]
Source Stack Trace [SRCS]
Hidden Stack [HIST]
Interrupt Stack [INTS]
Frame Trace [FRTR]
False Frame [FAFR]
Procedure Call Chain [PCCH]
Rough Stack Trace (managed space) [RSTM]
Rough Stack Trace Collection (unmanaged space) [RSCU]
Caller-n-Callee [CNCA]
```

```
Back
```

10. If we click on an individual analysis pattern in the DML-based version, the default web browser should open the appropriate link from Software Diagnostics Library[6] (which requires prior login). In the non-DML version, we need to use the **!sdl** command with the corresponding four-letter pattern tag:

```
0: kd> !sdl RSCU
```



11. We can unload the extension anytime if we no longer need it:

```
0: kd> .unload C:\EWMDA-Dumps\patterns\x64\patterns
```

Note: We recommend exiting WinDbg after each exercise to avoid possible confusion and glitches.

[6] https://www.dumpanalysis.org/blog