Windows Memory Dump Analysis

Extended

Extensions, Database and Event Stream Processing, Visualization

Revised Edition

Dmitry Vostokov
Software Diagnostics Services
Prerequisites

- Basic WinDbg usage
- Coding in a high-level language
- Ideal previous training:
  - Accelerated Windows Memory Dump Analysis
  - Accelerated .NET Core Memory Dump Analysis
  - Advanced Windows Memory Dump Analysis
  - Accelerated Windows Malware Analysis with Memory Dumps
Why Extended Memory Analysis?

- Limitations of existing commands
- Scripts may be slow or not convenient to use
- Different output format
- Get more insight
Training Goals

- Review 3rd-party extensions
- Map to memory analysis patterns
- Compare with traditional techniques
- Write our own extensions
- Use data processing and visualization
Schedule

- Survey of WinDbg extensions
- Writing WinDbg extensions
- Event stream processing
- Database processing
- Visualization
Training Principles

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

- Awesome WinDbg Extensions list
- My experience with Kafka
- My experience with JSON data processing
- My interest in data science and visualization (trace and log analysis)
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](http://www.dumpanalysis.org/windows-memory-analysis-checklist)
Links

- **Applications:**
  Download links are in the exercise E0.

- **Exercise Transcripts:**
  Included in this book.
Exercise E0

- **Goal:** Install WinDbg or Debugging Tools for Windows, or pull Docker image, and check that symbols are set up correctly

- **Memory Analysis Patterns:** Stack Trace; Incorrect Stack Trace

- \EWMDA\Exercise-E0.pdf
Survey of WinDbg Extensions

Exercises ES1 – ES7
Criteria

- General usefulness for dump analysis
- Addresses common manual techniques
- Corresponds to certain analysis patterns
Exercise ES1

- **Goal:** Explore Patterns WinDbg extension

- \EWMDA\Exercise-ES1.pdf
Exercise ES2

- **Goal:** Explore MEX WinDbg extension

- **Memory Analysis Patterns:** Zombie Processes; Instrumentation Information; Blocked Thread (Software); Active Thread; Suspended Thread; Wait Chain (ALPC); Input Thread; Exception Stack Trace; Stack Trace Collection (Predicate); Stack Trace Collection (CPUs); Spiking Thread; Execution Residue (Unmanaged Space)

- `\EWMDA\Exercise-ES2.pdf`
Exercise ES3

- **Goal:** Explore DbgKit WinDbg extension

- **Memory Analysis Patterns:** Module Collection; Historical Information; Driver Device Collection; Stack Trace (I/O devices); Stack Trace Collection (I/O drivers); System Object; Value References; Zombie Processes; Virtualized Process (WOW64); Stack Trace Collection; Environment Hint; Deviant Token; Raw Pointer; Out-of-Module Pointer

- \EWMDA\Exercise-ES3.pdf
Exercise ES4

- **Goal:** Explore `win32kext` WinDbg extension

- **Memory Analysis Patterns:** Handle Limit (GDI, Kernel Space); Wait Chain (Window Messaging)

- `\EWMDA\Exercise-ES4.pdf`
Exercise ES5

- **Goal:** Explore [SwishDbgExt](https://www.swishdbgext.com) WinDbg extension

- **Memory Analysis Patterns:** Historical Information; Missing Thread (Kernel Space); Driver Device Collection; Patched Code; Out-of-Module Pointer; Self-Diagnosis (Registry); System Object; Namespace

- \EWMDA\Exercise-ES5.pdf
Exercise ES6

- **Goal:** Explore Occhext WinDbg extension

- **Memory Analysis Patterns:** Execution Residue (Unmanaged Space); Namespace; Context Pointer; Step Dumps; Evental Dumps

- \EWMDA\Exercise-ES6.pdf
Exercise ES7

- **Goal:** Explore pykd WinDbg extension

- **Memory Analysis Patterns:** Execution Residue (Unmanaged Space)

- \EWMDA\Exercise-ES7.pdf
Raw Stack Analysis

- Symbolic hints at past behavior
- Past stack traces
- Errors, strings, pointers, pointers to pointers
Writing WinDbg Extensions

Exercises EW1 – EW3
Goal

- Survey different ways to write extensions
- Simple clean skeletons for further extension
- Useful functionality for analysis patterns
Exercise EW1

- **Goal:** Write WinDbg extension using [WdbgExts](https://www.microsoft.com) C API

- `\EWMDA\Exercise-EW1.pdf`
Exercise EW2

- **Goal**: Write WinDbg extension using **DbgEng** COM API

- \EWMDA\Exercise-EW2.pdf
Exercise EW3

- **Goal:** Write WinDbg extension using [ExtExtension](#) C++ API

- `\EWMDA\Exercise-EW3.pdf`
Event Stream Processing

Exercises EP1 – EP2
Apache Kafka

- WinDbg log
- log store
- log processing
- log consumers
Command Logs

WinDbg Producer
WinDbg Producer
WinDbg Producer

Kafka
- topic: ~*k output
- topic: !process 0 3f output
- topic: !vm output
- topic: dpS output
- topic: analysis aggregation

Stream Processing App

Consumer
- ELK Consumer
- Database Consumer

© 2023 Software Diagnostics Services
Exercise EP1

- **Goal:** Install Apache Kafka and verify that it works correctly
- \EWMDA\Exercise-EP1.pdf
Exercise EP2

- **Goal:** Connect WinDbg to Kafka for logging to various topics

- **Memory Analysis Patterns:** Structure Sheaf; Stack Trace (Command); Stack Trace Collection (Commands)

- \EWMDA\Exercise-EP2.pdf
Database Processing

Exercises ED1 – ED2
MongoDB

- WinDbg logs as NoSQL data
- Collections of command output, for example, !analyze -v or ~*k
- Command output with added metadata as a document
Exercise ED1

- **Goal:** Install MongoDB and verify that it works correctly

- \EWMDA\Exercise-ED1.pdf
Exercise ED2

- **Goal:** Connect WinDbg to MongoDB for storing analysis documents

- `\EWMDA\Exercise-ED2.pdf`
Visualization

Exercises EV1 – EV2
Pandas

- Tabular raw stack or heap data
- Thousands of rows per thread and millions per heap
- Hundreds of threads
Exercise EV1

- **Goal:** Install Jupyter Notebook and verify that it works correctly
- \\EWMDA\\Exercise-EV1.pdf
Exercise EV2

- **Goal:** Explore various execution residue visualization opportunities using Pandas and Matplotlib

- **Memory Analysis Patterns:** Execution Residue (Unmanaged Space); Region Profile; Region Clusters; Namespace

- \EWMDA\Exercise-EV2.pdf
Diagnostics Presentation Patterns

- Introduced in:
  
  Pattern-Oriented Debugging Process

- Include visualization

- New forthcoming pattern catalog
<table>
<thead>
<tr>
<th>Memory Analysis Pattern Links</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execution Residue (Unmanaged Space, User)</strong></td>
</tr>
<tr>
<td><strong>Execution Residue (Unmanaged Space, Kernel)</strong></td>
</tr>
<tr>
<td><strong>Instrumentation Information</strong></td>
</tr>
<tr>
<td><strong>Driver Device Collection</strong></td>
</tr>
<tr>
<td><strong>Stack Trace Collection (I/O drivers)</strong></td>
</tr>
<tr>
<td><strong>Stack Trace Collection (Predicate)</strong></td>
</tr>
<tr>
<td><strong>Stack Trace Collection (CPUs)</strong></td>
</tr>
<tr>
<td><strong>Handle Limit (GDI, Kernel Space)</strong></td>
</tr>
<tr>
<td><strong>Virtualized Process (WOW64)</strong></td>
</tr>
<tr>
<td><strong>Stack Trace Collection (Unmanaged Space)</strong></td>
</tr>
<tr>
<td><strong>Missing Thread (Kernel Space)</strong></td>
</tr>
<tr>
<td><strong>Wait Chain (Window Messaging)</strong></td>
</tr>
<tr>
<td><strong>Self-Diagnosis (Registry)</strong></td>
</tr>
<tr>
<td><strong>Out-of-Module Pointer</strong></td>
</tr>
<tr>
<td><strong>Deviant Token</strong></td>
</tr>
<tr>
<td><strong>Patched Code</strong></td>
</tr>
<tr>
<td><strong>Step Dumps</strong></td>
</tr>
<tr>
<td><strong>Region Profile</strong></td>
</tr>
<tr>
<td><strong>Stack Trace Collection (Commands)</strong></td>
</tr>
<tr>
<td><strong>Stack Trace (Command)</strong></td>
</tr>
<tr>
<td><strong>Zombie Processes</strong></td>
</tr>
<tr>
<td><strong>Module Collection</strong></td>
</tr>
<tr>
<td><strong>Historical Information</strong></td>
</tr>
<tr>
<td><strong>Stack Trace (I/O devices)</strong></td>
</tr>
<tr>
<td><strong>Exception Stack Trace</strong></td>
</tr>
<tr>
<td><strong>Blocked Thread (Software)</strong></td>
</tr>
<tr>
<td><strong>Active Thread</strong></td>
</tr>
<tr>
<td><strong>Suspended Thread</strong></td>
</tr>
<tr>
<td><strong>Wait Chain (ALPC)</strong></td>
</tr>
<tr>
<td><strong>Spiking Thread</strong></td>
</tr>
<tr>
<td><strong>System Object</strong></td>
</tr>
<tr>
<td><strong>Input Thread</strong></td>
</tr>
<tr>
<td><strong>Value References</strong></td>
</tr>
<tr>
<td><strong>Environment Hint</strong></td>
</tr>
<tr>
<td><strong>Raw Pointer</strong></td>
</tr>
<tr>
<td><strong>Context Pointer</strong></td>
</tr>
<tr>
<td><strong>Evental Dumps</strong></td>
</tr>
<tr>
<td><strong>Region Clusters</strong></td>
</tr>
<tr>
<td><strong>Namespace</strong></td>
</tr>
<tr>
<td><strong>Structure Sheaf</strong></td>
</tr>
</tbody>
</table>
Resources

- WinDbg Help / WinDbg.org (quick links and some extensions)
- DumpAnalysis.org / SoftwareDiagnostics.Institute / PatternDiagnostics.com
- Software Diagnostics Library
- Comprehensive WinDbg extension collection
- Kafka in Action / Kafka: The Definitive Guide / Apache Kafka
- MongoDB
- Pandas / ydata-profiling / Matplotlib / Polars
- Memory Dump Analysis Anthology (Diagnomicon)
Q&A

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