Windows Vista Heap Management Enhancements

Adrian Marinescu
Development Lead
adrmarin at microsoft dot com
Agenda

• Windows NT Heap Management basics and evolution
• Windows Vista heap – major milestone
  – Development principles and guidelines
  – Security features
  – Performance features
• Q & A
Introduction

• Security – industry-wide concern
• TwC driving multiple security initiatives
• The NT Heap
  – Strategic point in defense
  – Improved to respond to industry trends in usage
Part I – Basics
# Heap Evolution

<table>
<thead>
<tr>
<th>Basics</th>
<th>Performance</th>
<th>Opt-in SMP Scalability</th>
<th>Enhanced security</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT 4</td>
<td>NT 4 / SP4</td>
<td>XP / SP2</td>
<td>Windows Vista</td>
</tr>
<tr>
<td>Windows 2000</td>
<td></td>
<td>Windows 2003</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Industry</th>
<th>Workload</th>
<th>Exploitation</th>
<th>Parallelism</th>
</tr>
</thead>
</table>

*Time*
NT Heap Overview

Windows applications

kernel32
Win32 heap interfaces (Heap*, Local*, Global*)

ntdll

Optional Front-End Heap
(lookasides, Low Fragmentation Heap)

Segment Management
(for blocks <512K¹)

Large blocks management

Kernel

The NT Memory Manager

¹ On 32 bit platforms
The NT Heap Core Management

- Free lists
  - 1024
  - > 1K (1 – 512 KBytes)

- Segment array
  - Segment 1
  - Segment 2
  - Segment 3
  - Segment 64
  - > 512K

- Virtual Blocks List
  - > 512K
Heap Segments

Reserved space

Seg.H  B1  B2 Uncommitted space  Bn  ...  By Uncommitted space
# Block Entry in prior Windows NT Versions

<table>
<thead>
<tr>
<th>Previous Block Size</th>
<th>Size</th>
<th>Prev Size</th>
<th>Flags</th>
<th>...</th>
<th>F-Link</th>
<th>B-Link</th>
<th>...</th>
</tr>
</thead>
</table>

| 0:018> dc 04392f80 04392f80 00040002 00000001 | 0442da60 00240e68 |

- **F-Link**
- **B-Link**
- **segment**
- **unused bytes**
- **flags**
- **tag**

Previous block size

Size
Role of Link Entry in Early Exploits

- Arbitrary pointer write

\[
\text{mov eax, DWORD PTR [ecx]} \\
\text{mov ecx, DWORD PTR [ecx+4]} \\
\text{mov DWORD PTR [ecx], eax} \\
\text{mov DWORD PTR [eax+4], ecx}
\]
Lookaside Lists

• Non-blocking single-linked lists

Lookaside lists

<table>
<thead>
<tr>
<th>0-8</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9-16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1016-1024</td>
<td>Header</td>
<td>Link</td>
</tr>
</tbody>
</table>
The Low Fragmentation Heap Architecture

Allocation buckets

| 0-8   | SMP Unit 1 |
| 9-16  | SMP Unit N |
| 1025-1088 | SMP Unit 1 |
| 15873-16384 | |

Segments pool
Low Fragmentation Heap Block Entry

LFH Block

- Sub-Segment
- Flags
- ... Relative link
- User data

Free s-list

Size

Sub-Segment

8 bytes
Early Heap Mitigations

• Safe List Removal
  Entry->FwdLink->BkLink == Entry->BkLink->FwdLink == Entry

• 8-bit cookie tested on free

• LFH block entry encoding
  F (random number, Block address, heap)
Change in Landscape

• New exploiting methods surfaced
• Change in usage outlook
  – Memory usage
  – Increase availability of SMP
  – Increase relevance of 64 bit computing
• Code quality – higher demand in industry
Windows Vista Heap Manager
Key Development Directions

• Performance and reliability
• Security
• Code quality
Windows NT Heap Requirements
Security

• Correctness – like:
  – Guarantees requested sizes
  – Lifetime of allocations
  – Clearing content when requested etc.

• Defense line in heap based exploits:
  – Attempts to mitigate the effect of an attack
  – Makes difficult hiding heap-based exploits
Performance

- Scale from small devices to large servers
- Optimized for varied usage patterns
- Follow the industry trend
  - Memory usage
  - Increase in SMP availability
  - H/W architecture advances
Compatibility

• Applications may rely on things like:
  – Realloc returning same pointer
  – Read/write after releasing a block
  – Double free
  – Overruns over unused structures etc.

• Heap changes may have unintended effects, such as:
  – Crashes, leaks or broken functionality in poorly written applications
  – Severe performance regressions
Part II - Windows Vista Heap
Windows Vista Heap Security Features

• Block metadata randomization
• Integrity check on block entry
• Algorithm variation in response to usage pattern
• Random rebasing
• Function pointer randomization
• Abrupt application termination on error
Block Metadata Randomization

• A part of the header is XORd with a random value
• Low performance impact
• Should make guessing the right value impractical
• Flexible and contained algorithm and implementation
• Agile in updates
Entry Integrity Check

• Previous 8-bit cookie has been repurposed to validate a larger part of the header
• Value may be randomized along with the other fields
• Validated during internal operations too
Demo – Heap Header Layout
Runtime Algorithm Variation

• Automatic tuning
  – Shift to LFH allocations at arbitrary points on runtime
  – Triggers on various patterns
  – Involves also de-commit / commit policies
More Heap Randomizations

• Heap base randomization – things to consider:
  – Fragmentation of the application address space affecting large server applications
  – Possible performance issues if higher randomization is used

• Heap function pointer randomization
  – Takes away a known place to facilitate the code execution along with rebasing
Demo
Abrupt Termination on Error

• Any data inconsistency or invalid heap function usage detected may trigger it
• The scope is process-wide (any heap in the process has the same behavior)
• The process is terminated via Windows Error Reporting
• Detailed info is available in the dump file
• No function provided to disable it
• On by default for 64 bit platforms & apps
Termination on Errors (cont.)

• Programmatic opt-In method
  (new `HeapEnableTerminationOnCorruption` class defined)

BOOLO HeapSetInformation(
  HANDLE HeapHandle,
  HEAP_INFORMATION_CLASS HeapInformationClass,
  PVOID HeapInformation,
  SIZE_T HeapInformationLength);

• Large number of components with Windows Vista are opted in
• The information is available in a debugger extension
Demo
NT Heap Manager – Improves Code Quality

Benefits to app developers
• Early error detection
• Improved debugging aid to reduce cost of investigating corruptions
• Reduced tolerance to misusage
• Windows Vista apps will be more resilient to future heap changes
Known Attack Vectors & Windows Vista

- Removed lookaside list and array of lists targeted by previous exploits
- Integrity check on block metadata significant obstacle to brute force attacks
- Most Windows processes terminate on memory errors
- Dynamic (runtime) change in heap algorithms obstacle to consistent exploits
- Heap structures and memory mgmt changes limit portability of exploits
Security enhancements are a journey

- Mitigations are not substitute for good development practices
- Windows Vista is just a milestone in continual heap improvements
Windows Vista Heap Perf & Reliability

• Improved scenarios by default for:
  • SMP scalability
  • External fragmentation
  • Large heaps
• Improved reference locality on 64 bit platforms
• Reduced Virtual Address exhaustion
• Increased resilience to patterns involving long-term allocations
Key Performance Enhancements

- Automatic tuning
- Lower granularity of control policies to switch to the Low Fragmentation Heap
- Use of lazy initialization
- Redesigned segment management
- Improved internal lookup algorithms
- Addressed fragmentation in problematic scenarios
- Lower overhead on 64 bit
Fragmentation Test (512 blocks / 80 bytes)

**Virtual address Cost**

- **Bytes per leaked block**
  - Run time (sec)
  - Recent Windows Vista
  - Windows Server 2003

**Committed memory cost**

- **Bytes per leaked block**
  - Run time (sec)
  - Recent Windows Vista
  - Windows Server 2003

**Heap performance on heap expansion pattern 1 (512 blocks) (70x)**

- **Ops./sec.**
  - Run time (sec)
  - Recent Windows Vista
  - Windows Server 2003
# Fragmentation Scenario II

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Ops/sec (Recent Windows Vista)</th>
<th>Ops/sec (Windows Server 2003 SP 1)</th>
<th>Improvement (x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>2576004</td>
<td>388</td>
<td>6639</td>
</tr>
<tr>
<td>512</td>
<td>927709</td>
<td>151</td>
<td>6144</td>
</tr>
<tr>
<td>1024</td>
<td>403774</td>
<td>51</td>
<td>7917</td>
</tr>
<tr>
<td>2048</td>
<td>194180</td>
<td>25</td>
<td>7767</td>
</tr>
<tr>
<td>4096</td>
<td>82534</td>
<td>12</td>
<td>6878</td>
</tr>
</tbody>
</table>
Memory footprint on 2 GBytes heap expansion

- **Reserved Memory In Recent Windows Vista**
- **Committed Memory In Recent Windows Vista**
- **Reserved Memory In Windows Server 2003**
- **Committed Memory In Windows Server 2003**
Summary

• Attacks get more sophisticated ...
• But so does the heap management – and not only for security
• We laid the foundation for increased agility in heap improvements with reduced compatibility risks
• Improved scenarios for SMP and large memory usage
• Designed to enhance the code quality for applications
• We are not yet done ... we are looking forward for further enhancements as needed
• Come see me with your ideas!
Resources

- Feedback on Heap: heapext@microsoft.com

- Debugging tools: http://www.microsoft.com/whdc/devtools/debugging/debugstart.mspx

Still to Come!

16:45 – 18:30  Case Study: The Secure Development Lifecycle and Internet Explorer 7  TONY CHOR
secure@microsoft.com

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.