Structural Analysis of the Windows NT Heap for Memory Forensics

Daniel Uroz, Abraham Díaz-Campo Pinilla, **Ricardo J. Rodríguez**

Universidad Zaragoza

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

March 25, 2026

DFRWS EU 2026
Linköping (Sweden)

Agenda

Universidad Zaragoza

Agenda

Universidad Zaragoza

Introduction

The Shift to User-Space Exploitation

- Modern attackers continuously adapt to OS mitigations
- There is a clear shift towards **user-space memory** to store payloads, obfuscate runtime behavior, and evade traditional EDR detection
- The **Heap** is the primary vehicle for these dynamic allocations
- As a consequence, forensic evidence is now deeply embedded in highly dynamic, ephemeral memory structures

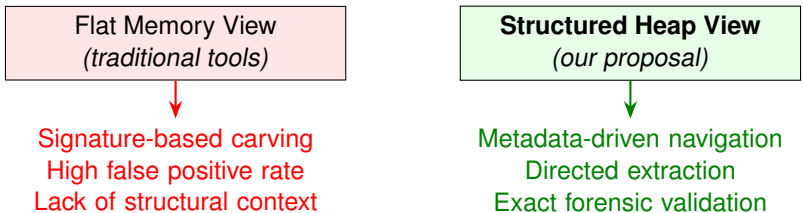
Universidad Zaragoza

Introduction

The Problem: The Forensic Blind Spot

The problem

- Current forensic tools often treat the heap as a flat, unstructured region
- Analysts rely heavily on blind carving or signature scanning



Why is structured heap forensics of interest today?

- **Heap Spraying:** Attackers allocate numerous objects containing shellcode to ensure execution predictability. Carving fails to identify the deployment pattern
- **Use-After-Free (UAF):** Exploiting dangling pointers. Requires exact knowledge of heap chunks and their metadata states (Free vs. Allocated)
- **Heap Feng Shui:** Manipulating the heap layout

Conclusion

You cannot investigate structural attacks with non-structural tools

Universidad Zaragoza

Agenda

Universidad Zaragoza

Background

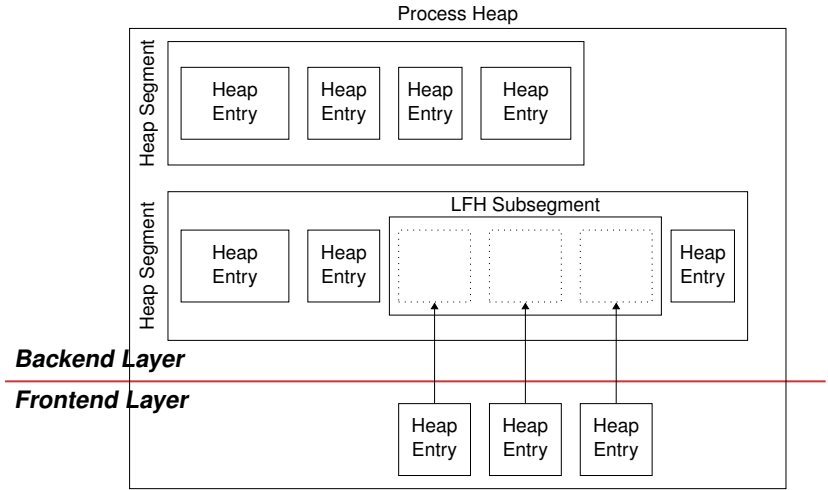
Windows NT Heap Architecture: An Overview

- The **NT Heap** remains predominant in user-space allocations
- It is not a single entity, but a layered architecture:
  - **Backend Layer:** The foundation. Manages heap segments and large memory blocks
  - **Frontend Layer:** Low Fragmentation Heap (LFH). Optimizes small, uniform block allocations to prevent fragmentation
- *Two structural layers → Two distinct traversal algorithms required*

Universidad Zaragoza

Background

Visualizing the Two-Layered Structure



Agenda

Universidad Zaragoza

Technical Contribution: HeapList

- We translate this structural understanding into an actionable forensic tool
- **Introducing HeapList:** An open-source plugin for the Volatility 3 framework
- Designed to map, decode, and extract heap allocations structurally
- **Universal Compatibility:** Fully supports Windows versions from Vista up to Windows 11 (both x86 and x64 architectures)

Available on GitHub!

Universidad Zaragoza

Technical Contribution: HeapList

HeapList in Action

- Seamless integration into standard incident response workflows

```
$ python3 vol.py -f memory.raw windows.heaplist.HeapList
```

- **Output capabilities:**
  - Lists all heaps per process
  - Identifies segments, uncommitted ranges, and LFH activation
  - Dumps specific entries, categorized by backend or frontend origin

Universidad Zaragoza

Technical Contribution: HeapList

HeapList in Action – Output Example

```
$ vol3.py -f /path/to/windows-memory-image.raw windows.heaplist --pid 11956 --dump-all
Volatility 3 Framework 2.11.0
Progress: 100.00 PDB scanning finished
PID Name Heap Segment Entry Size Flags State Layer Data File Output

11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b30740 0x20 [01] busy backend PUBLIC=C:\Users\ 11956.heap.22225b30740.dmp
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b30760 0x20 [01] busy backend SESSIONNAME=Cons 11956.heap.22225b30760.dmp
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b30780 0x20 [01] busy backend SystemDrive=C:.. 11956.heap.22225b30780.dmp
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b307a0 0x20 [01] busy backend SystemRoot=C:\Wi 11956.heap.22225b307a0.dmp
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b307c0 0x40 [01] busy backend USERDOMAIN_ROAMINGPROFILE=DESKTOP-H23BTC 11956.heap.22225b30
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b30800 0x20 [01] busy backend USERNAME=User... 11956.heap.22225b30800.dmp
[...redacted...]
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b414c0 0x310 [01] busy backend AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA 11956.heap.22225b4
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b417d0 0x2010 [09] busy internal backend ...%"......%".....,.............. 11956.he
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b41820 0x310 [80] free lfh ..................................... 11956.heap.22225b4182
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b41b30 0x310 [90] busy lfh AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA 11956.heap.22225b41b3
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b41e40 0x310 [90] busy lfh AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA 11956.heap.22225b41e4
[...redacted...]
11956 cmd.exe 0x22225b30000 0x22225b30000 0x22225b437e0 0x4010 [01] busy backend ???? Unavailable
[...redacted...]
```

Universidad Zaragoza

Technical Contribution: HeapList

Methodology 1: Backend Traversal

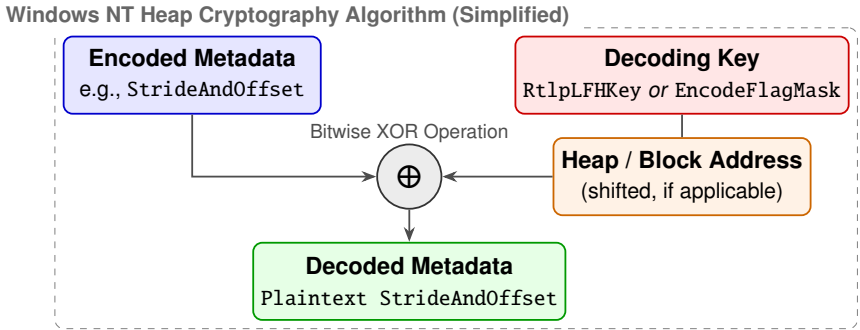
- **Iteration Path:** Process Environment Block (`PEB`) → `ProcessHeaps` array → `SegmentList`
- **Key Challenges Addressed:**
  - *Uncommitted Ranges:* Properly handling uncommitted memory gaps to prevent traversal crashes
  - *Metadata Decoding:* Successfully applying the `EncodeFlagMask` to decode entry headers

Universidad Zaragoza

Technical Contribution: HeapList

Methodology 2: Frontend (LFH) Traversal

- Activated when `FrontEndHeapType = 0x02`
- Parses `SubSegmentZones` to reach the `HEAP_SUBSEGMENT` structures
- **The Obfuscation Hurdle:** Windows 8.1+ introduced XOR encoding for LFH entries
- **Our Solution:** We successfully locate and reverse the XOR encoding using the `RtlpLFHKey`, recovering 100% of the LFH metadata



Agenda

Universidad Zaragoza

Validation

#1: Structure vs. Carving

- To prove reliability, we performed **Cross-Validation with** `WinDbg`
- *Method:* An *entry-by-entry* comparison between HeapList output and live `WinDbg` debugging data (checking address, size, and state)

Result

Structural exactness

Unlike file carving, HeapList is **structurally consistent, effectively eliminating the false positives** inherent to traditional carving heuristics

Universidad Zaragoza

Validation
#2: Controlled Testing

- **Heap API Laboratory:**
  - We developed custom C/C++ applications to enforce specific heap behaviors
  - *Goals:* Force the activation of the LFH, fill backend segments, and trigger specific allocation patterns

Result

HeapList flawlessly mapped the expected behavior, decoded the offsets, and retrieved the injected payloads under controlled conditions

Universidad Zaragoza

Agenda

Universidad Zaragoza

Case Study: `Telegram Desktop`

- **Scenario:** Applied HeapList to memory dumps of Telegram Desktop running on Windows 11
- **Goal:** Assess the viability of structured heap forensics to extract user data from a modern app
- **Condition:** The application was running, but the UI was locked by the user

Universidad Zaragoza

Case Study: `Telegram Desktop`

Message & Chat Recovery

- By traversing the structured heap, we extracted highly sensitive communication artifacts without relying on database carving. In particular:

- **Unread Context:** Recovery of messages from chats that were *not recently accessed* by the user
- **Deleted Artifacts:** Successful recovery of both *edited* and *deleted* messages still residing in free heap chunks

Takeaway

The heap acts as an unintentional cache for ephemeral communications

Universidad Zaragoza

Case Study: `Telegram Desktop`

Authentication & Persistence

High-Value Artifact: Plaintext Password

We successfully extracted the local unlock password in **plain text** directly from the heap. This allows analysts to bypass local encryption if a machine is **seized while powered on and locked**

- **Data Persistence:** We observed that data remains highly persistent in memory over time
- The active heap structures are destroyed only upon an explicit *Logout* event

Universidad Zaragoza

Agenda

Universidad Zaragoza

Applications and Limitations

Forensic Applications

1. **Structured Triage:** Rapidly prioritizing critical subsegments during IR instead of dumping all RAM
2. **Assisted Reverse Engineering:** Mapping memory structures dynamically
3. **Heap Layout Visualization:** Detecting *Heap Feng Shui* and exploitation attempts
4. **Payload Recovery:** Extracting dynamic, memory-resident payloads without relying on signatures

Universidad Zaragoza

Applications and Limitations

Methodological Limitations

- **Memory Paging:** Susceptible to swapped-out pages (inherent of user-space memory forensics)
- **Metadata Corruption:** Corruption (e.g., from an aggressive exploit) can break the traversal chains
- **Scope:** Currently lacks support for the Segment Heap (used extensively by Win 10/11 UWP apps)

Universidad Zaragoza

Agenda

Universidad Zaragoza

Related Work

Traditional memory carving & YARA scanning

- Treat user-space memory as a “flat” sequence of bytes
- Highly prone to false positives and completely lack contextual allocation metadata
- Struggle to reconstruct fragmented payloads or encoded memory

Universidad Zaragoza

Related Work

Traditional memory carving & YARA scanning

- Treat user-space memory as a “flat” sequence of bytes
- Highly prone to false positives and completely lack contextual allocation metadata
- Struggle to reconstruct fragmented payloads or encoded memory

Previous heap analysis plugins

- Historically limited to older operating systems (e.g., Windows XP/7)
- Often lack robust support for modern LFH structures in Windows 10/11
- Cannot natively decode modern heap metadata cryptography

Universidad Zaragoza

Related Work

Traditional memory carving & YARA scanning

- Treat user-space memory as a “flat” sequence of bytes
- Highly prone to false positives and completely lack contextual allocation metadata
- Struggle to reconstruct fragmented payloads or encoded memory

Previous heap analysis plugins

- Historically limited to older operating systems (e.g., Windows XP/7)
- Often lack robust support for modern LFH structures in Windows 10/11
- Cannot natively decode modern heap metadata cryptography

HeapList (our contribution)

- From *blind carving* to **structured navigation**
- Exhaustive, cross-version (Vista → Win 11) heap intelligence natively to Volatility 3

Agenda

Universidad Zaragoza

Conclusions and Future Work

Conclusions and Future Work

Future Work

- Extending full structural traversal support to the **Segment Heap**
- Behavioral signatures for automated exploit detection (e.g., detecting Use-After-Free patterns)
- Heuristics to identify anomalous patterns and relationships between isolated heap allocations

Structural Analysis of the Windows NT Heap for Memory Forensics

Daniel Uroz, Abraham Díaz-Campo Pinilla, **Ricardo J. Rodríguez**

Universidad Zaragoza

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

March 25, 2026

DFRWS EU 2026
Linköping (Sweden)