Memory Eye

RECON 2011

Yoann Guillot
Sogeti / ESEC R&D
yoann.guillot(at)sogeti.com

Plan

Memory Eye

- Global analysis of a program
- A tool to analyse the dynamic heap
- Pattern matching

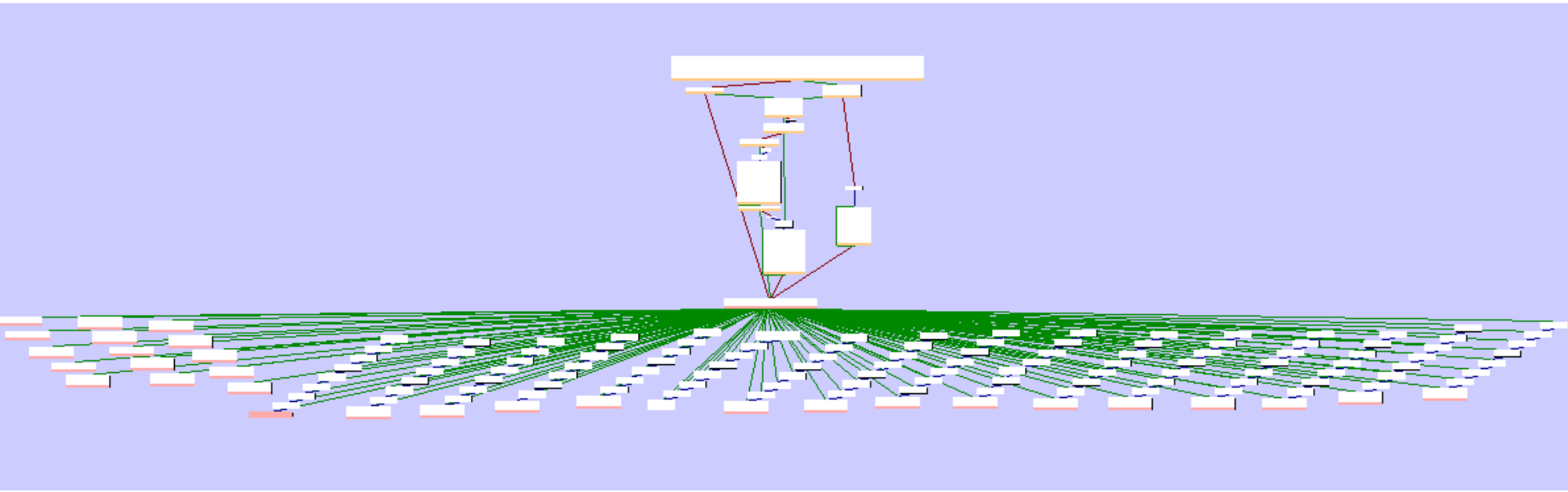
Why ?

Assembly is hard

- Weird instructions
  - xor
  - pmulhrsw
- Unusual architecture
  - x64
  - arm
- Weird algorithms[1]

[1] http://article.gmane.org/gmane.comp.lib.glibc.alpha/15278

wtf.asm



Code analysis

Pros

- Good tools
  - IDA
  - BinNavi[a]
- Large community

[a] M.I.A 01/03/11

Cons

- Bottom-up
- New target binary version == reset
  - BinDiff
- Prison

Plan

Taxinomy

Stack

- Scope == function (+subfunctions)
- Constrained size

Globales

- Compilation-defined
- Life span: permanent
- Included in the binary [a], so big ⇒ big file
- Static

[a] except .bss

Taxinomy (2)

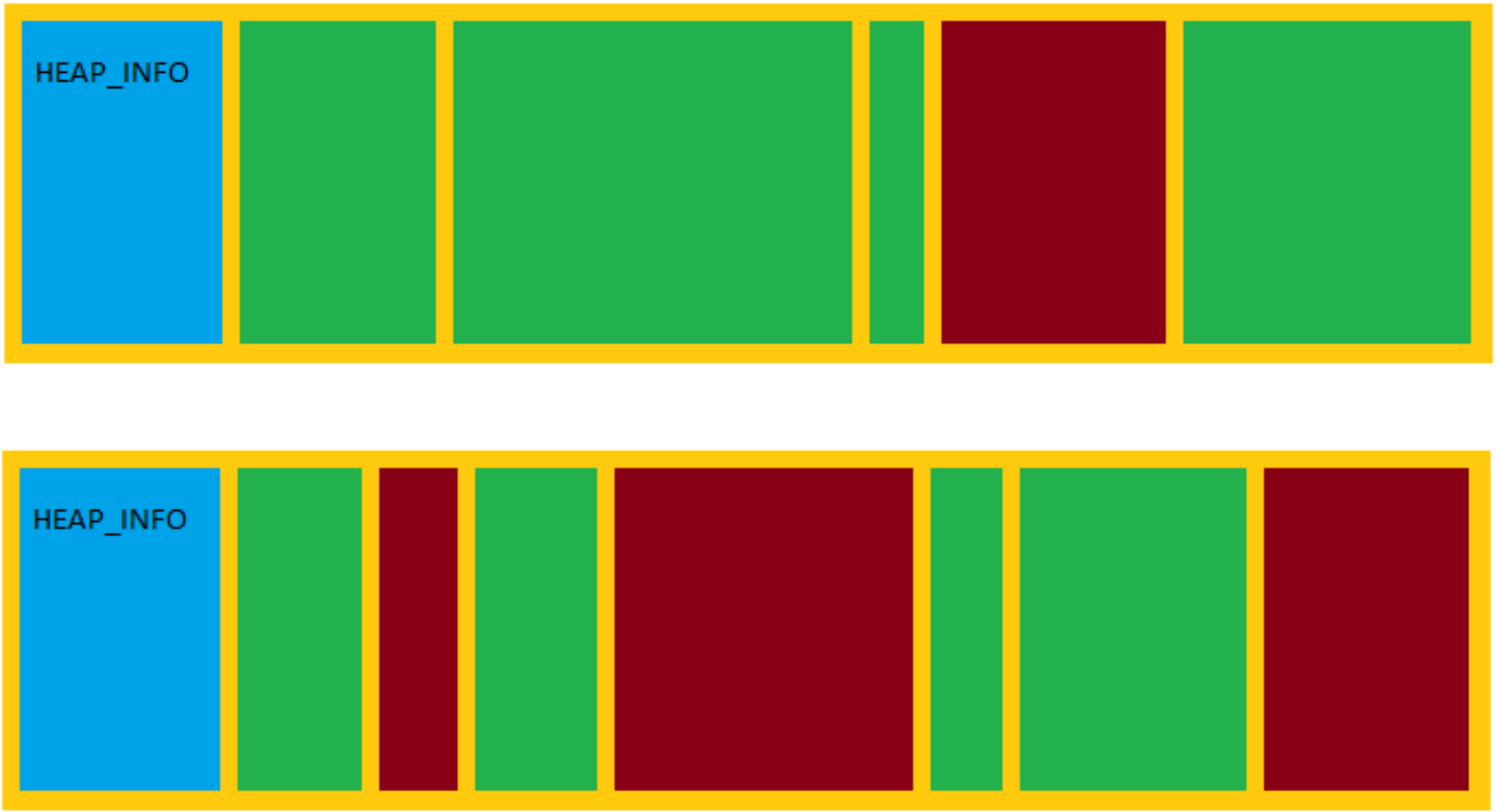
Heap

- Dynamic
  - On-demand (de)allocation
  - Optimal RAM usage
- Custom life span

Heap structure

- "Heap" = 1+ independent areas
  - Per librairy
  - Per thread
- Each one is a sequence of *chunks*
- Chunk = header + data
- Chunks handled through *malloc()* and *free()*
- Optimisations (lookaside)

Heap



Heap

Custom allocator

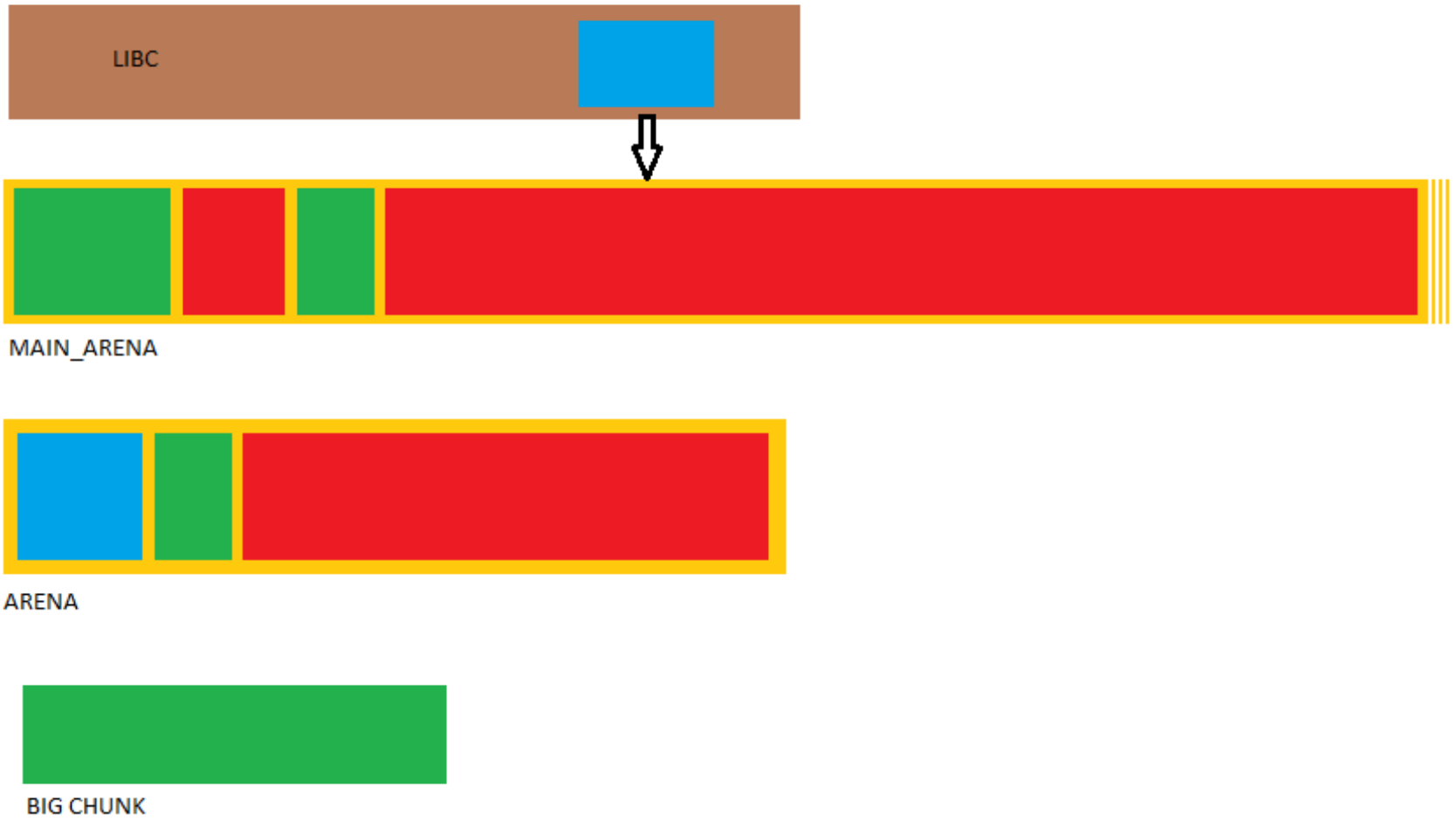
- OS implements do-it-all allocators
- Application may use a custom allocator
  - Tailored for specific chunk sizes
  - Optimized to reduce fragmentation
  - Optimized for speed

Plan

GLibc 2.11

- Independent area called “arena”
- Initial area **main_arena** = *sbrk()*
- Secondary areas = *mmap()*
- Big chunks = *mmap()*
- Cache for small free chunks = *lookaside*

GNU/Linux Heap



GLibc: extracting main_arena

main_arena is not exported.

Option 1

- Ignore the problem
- Find sbrk in /proc/*pid*/maps (`[heap]` tag)
- Pb: no secondary arenas, no lookaside

Option 2

1. Find a function using the structure
2. Extract structure address
3. Walk the linked list of arenas

In any case, we need to scan mmaped zones for big chunks.

GLibc: extracting main_arena

```
dasm.disassemble_fast('malloc_trim')
b = dasm.block_at('malloc_trim')
if b.list.last.opcode.name == 'call'
  # x86 getip()
  dasm.disassemble b.to_normal.first
end
# mutex_lock(&main_arena.mutex) gives us the addr
cmpxchg = dasm.decoded.values.find { |di|
  di.kind_of? DecodedInstruction and
  di.opcode.name == 'cmpxchg'
}
raise 'no_cmpxchg' if not cmpxchg
indir = cmpxchg.instruction.args.first.symbolic
arena_ptr = d.backtrace(indir.pointer, cmpxchg.address)
if arena_ptr.length == 1
        arena_ptr = arena_ptr.first.reduce
end
raise "cant_find_mainarena" if not arena_ptr.kind_of? Integer
arena_ptr += libc_base
```

Chunks enumeration

- Arena ⇒ top chunk + len - system_mem
- Walk in sequence
- Remove from lookaside
- Scan mmaped big chunks

Plan

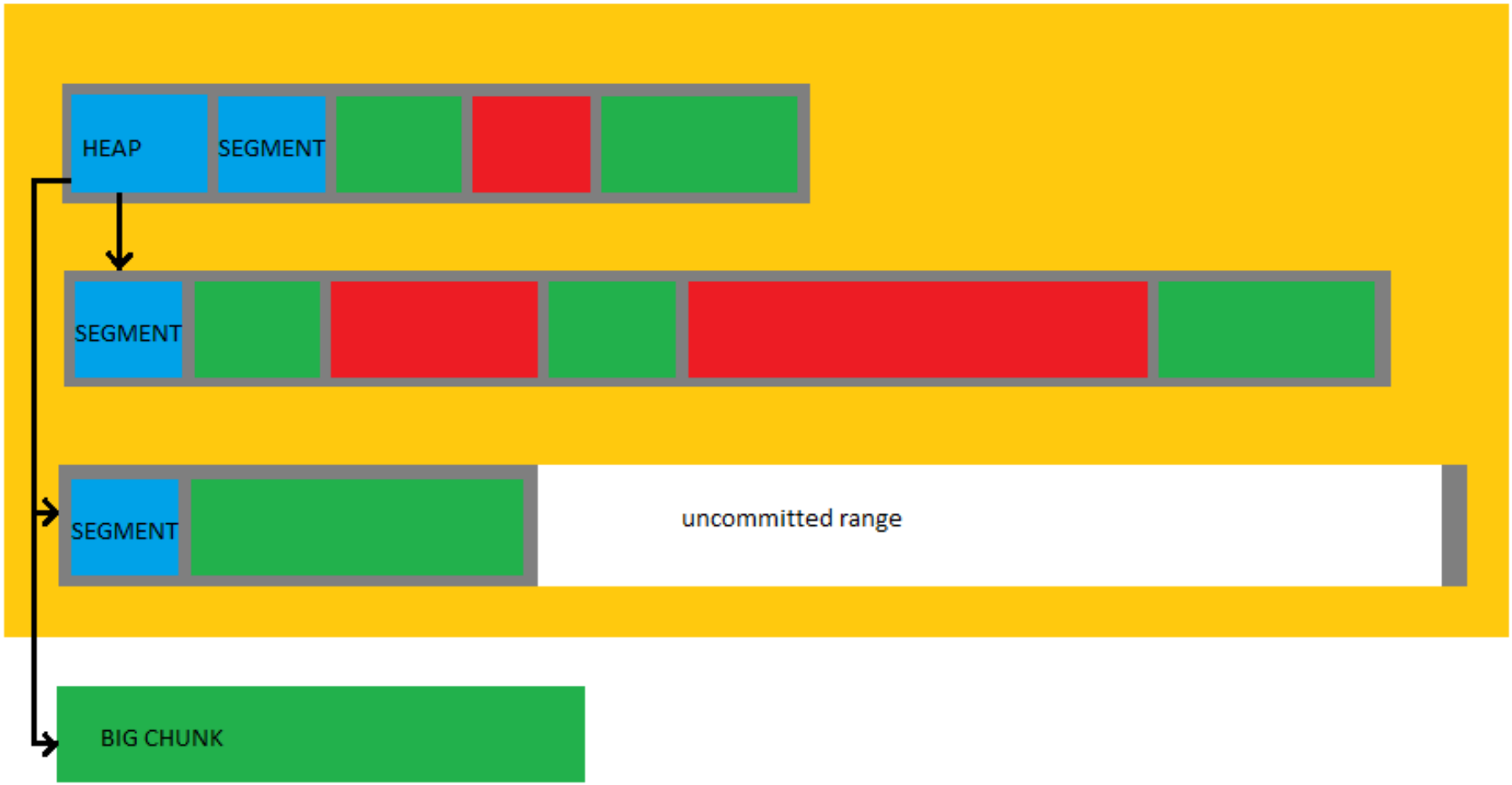
Windows XP

- Independant area called "_HEAP"
- Big chunks = *VirtualAlloc()*
- Free chunk cache = *LookAside*
- *ToolhelpSnapshot(HEAPLIST)*

Windows heap

- Heaps divided in Segments
- One segment size = twice the previous'
  - FirstEntry
  - LastEntryInSegment
  - UncommittedRange

Heap Windows



Windows 7

- Rewrote _**HEAP**
- Suppressed LookAside
- Use “Low Fragmentation Heap”
- Encode chunk headers

Windows 7 LFH

- Built on top of the backend (segments etc)
- Store complex structures in backend chunks
- HEAP $\Rightarrow$ LFH_HEAP $\Rightarrow$ HEAP_LOCAL_DATA $\Rightarrow$ LOCAL_SEGMENT_INFO $\Rightarrow$ HEAP_SUBSEGMENT $\Rightarrow$ USERDATA

Plan

So what ?

Building cross-reference graph

- Scan each chunk content
- Pointer to *chunk* $\Rightarrow$ new graph edge

Equivalence classes

- Tables
  - 1 *chunk* linking many same-sized *chunks*
- Linked lists
  - *chunk* linking a same-sized chunk
  - next one does the same
  - pointer offset is the same

Plan

Dwarf Fortress

- `http://www.bay12games.com/dwarves/`
- Massively Singleplayer Offline Role-Playing Game
- Fantasy universe simulation
- Active hacking community
- Very detailed world
- C++ code
- Regular new versions

$\Rightarrow$ Ideal application

Demo

Demo

Other use cases

- Most program manipulating lots of differenciated data

Questions ?

References

- http://www.blackhat.com/presentations/bh-usa-09/MCDONALD/BHUSA09-McDonald-WindowsHeap-PAPER.pdf
- http://lateralsecurity.com/downloads/hawkes_ruxcon-nov-2008.pdf
- http://www.eglibc.org/cgi-bin/viewcvs.cgi/branches/eglibc-2_13/libc/malloc/malloc.c?rev=12752&view=markup
- http://illmatics.com/Understanding_the_LFH.pdf
- http://metasm.cr0.org/