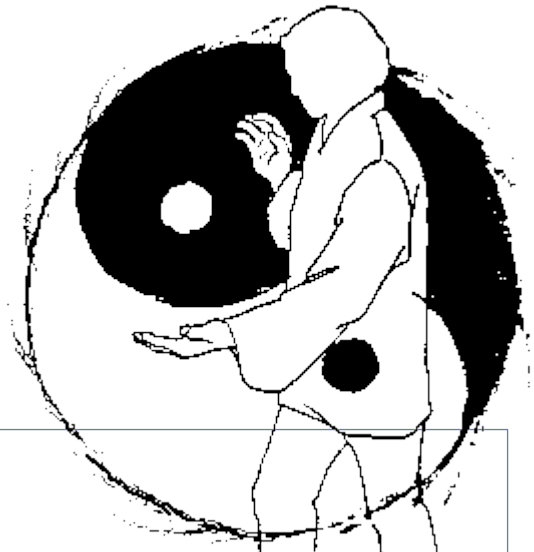


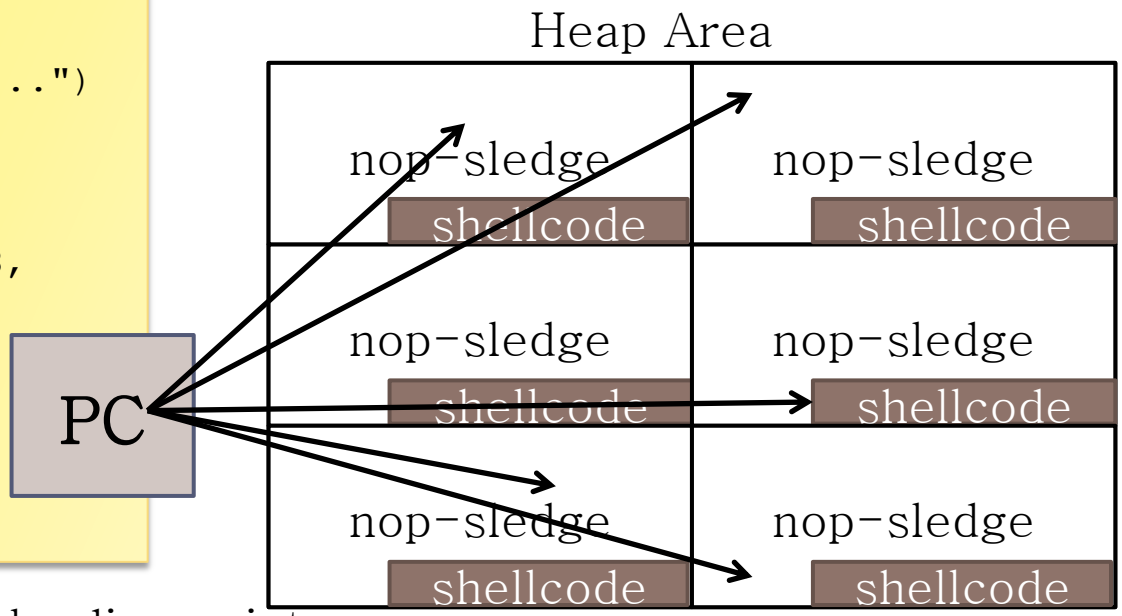
Heap Taichi:
Exploiting Memory Allocation Granularity in Heap-Spraying Attacks

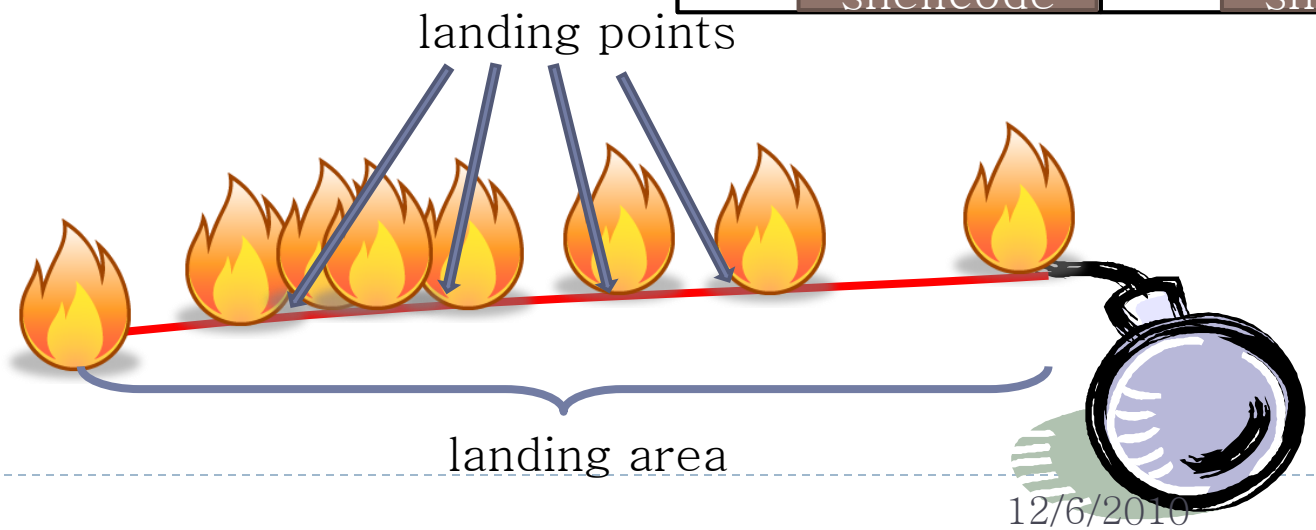
Yu Ding[1], Tao Wei[1], Tielei Wang[1], Zhenkai Liang[2], Wei Zou[1]

[1]Peking University, [2]National University of Singapore

A Typical Heap-Spraying Attack

```
var cc = unescape("%u0C0C");
var sc = unescape("%u785C%u3334...")
while (cc.length * 2 < 0x100000)
    cc+=cc;
cc=cc.substring(sc.length + 0x38,
                0x100000/2);
var m = new Array();
for(i = 0; i < 200; i++)
    m[i] = cc + sc;
```

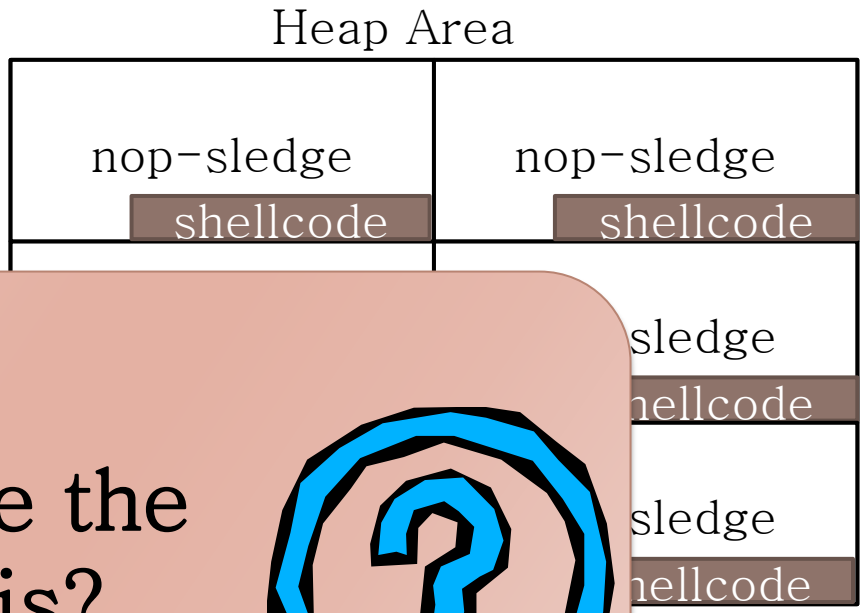




Existing Solutions

Measure how much of the heap can be used as nop-sledge

Because of randomization (ASLR), the large amount of nop instructions is required.



A direct d
measure h
can be use
flows to a

But how reliable the randomization is?

Surface A sledge)

Normalized ap data)

If the NSA > threshold, generate an alarm.

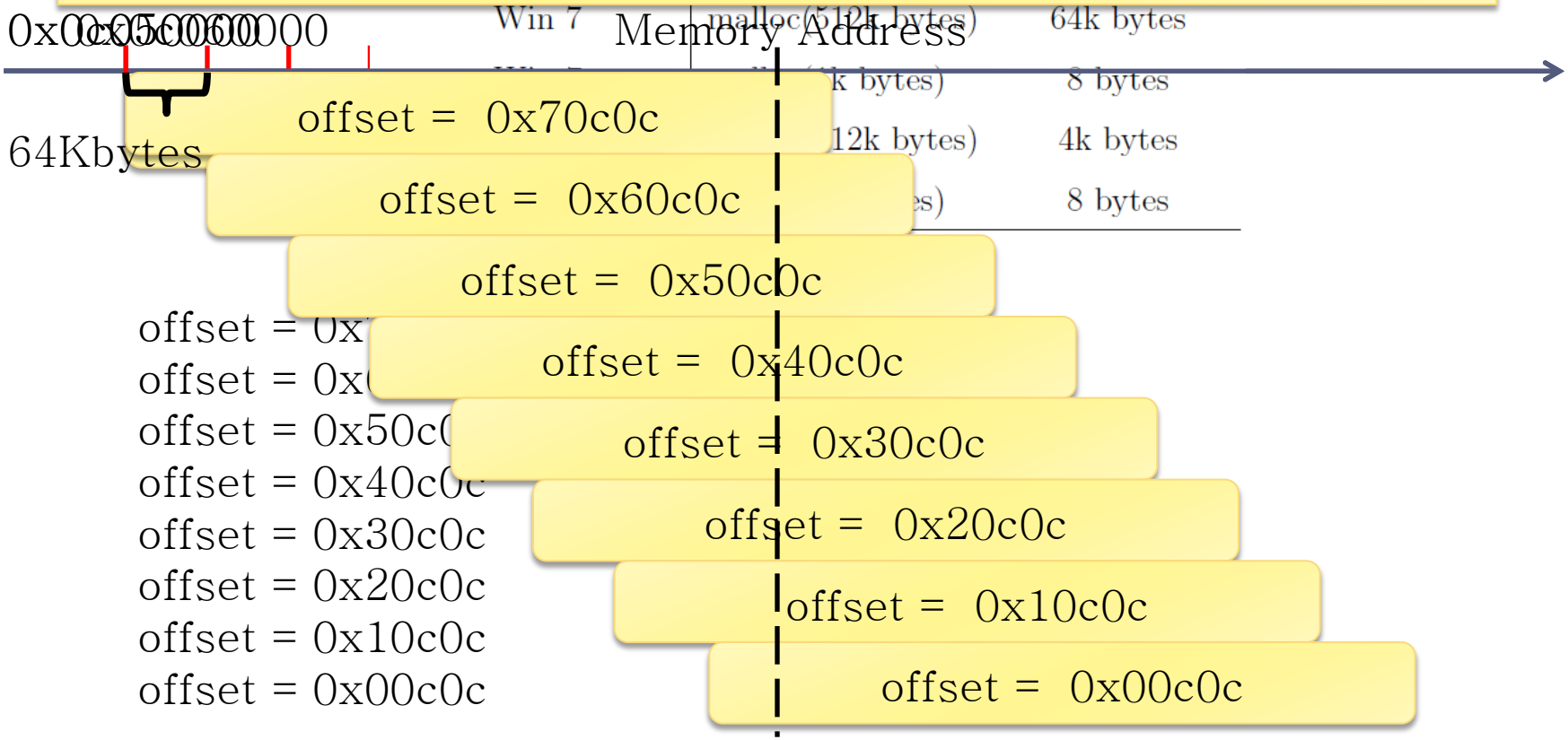
Nozzle [USENIX'09]

Observation
Memory Allocation Granularity

OS(32bit)	Allocation	Alignment size
Win 7	malloc(512k bytes)	64k bytes
	k bytes)	8 bytes
	12k bytes)	4k bytes
	es)	8 bytes

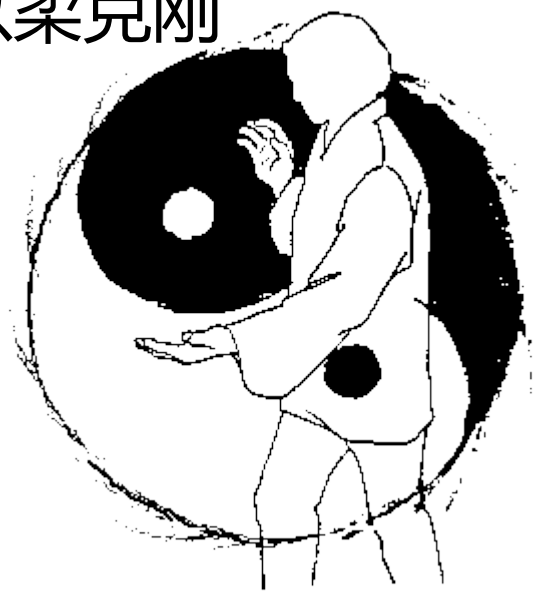
The memory allocation is much more predictable than expected.

- E.g. address 0c0c0c0c can only be 8 possible positions in a 512kbytes block



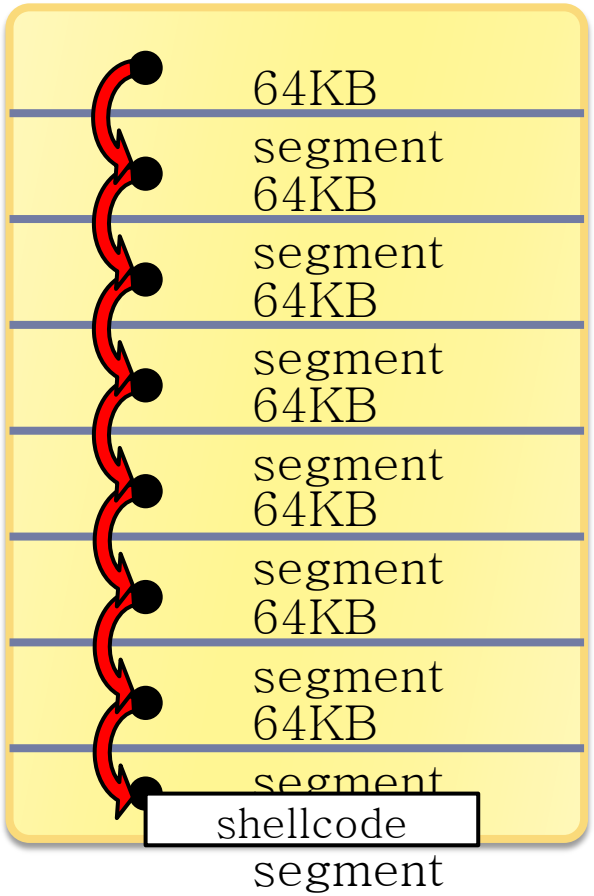
Proof of Concept: Heap Taichi

- Memory allocation granularity makes defense mechanisms less flexible
- We discuss a few transformations to reduce heap-spraying attacks' footprint without loosing its …
- Countering force with agility 以柔克刚

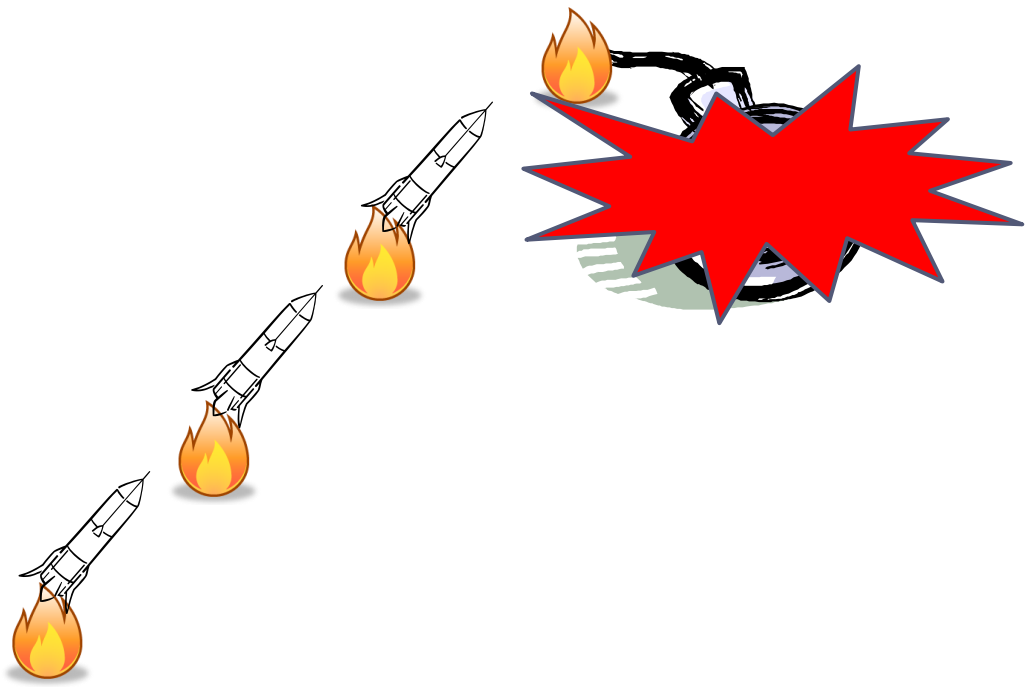


Basic Structure of TypeA (Passing Flower)

alignment size = 64Kbytes
blocksize = 512Kbytes

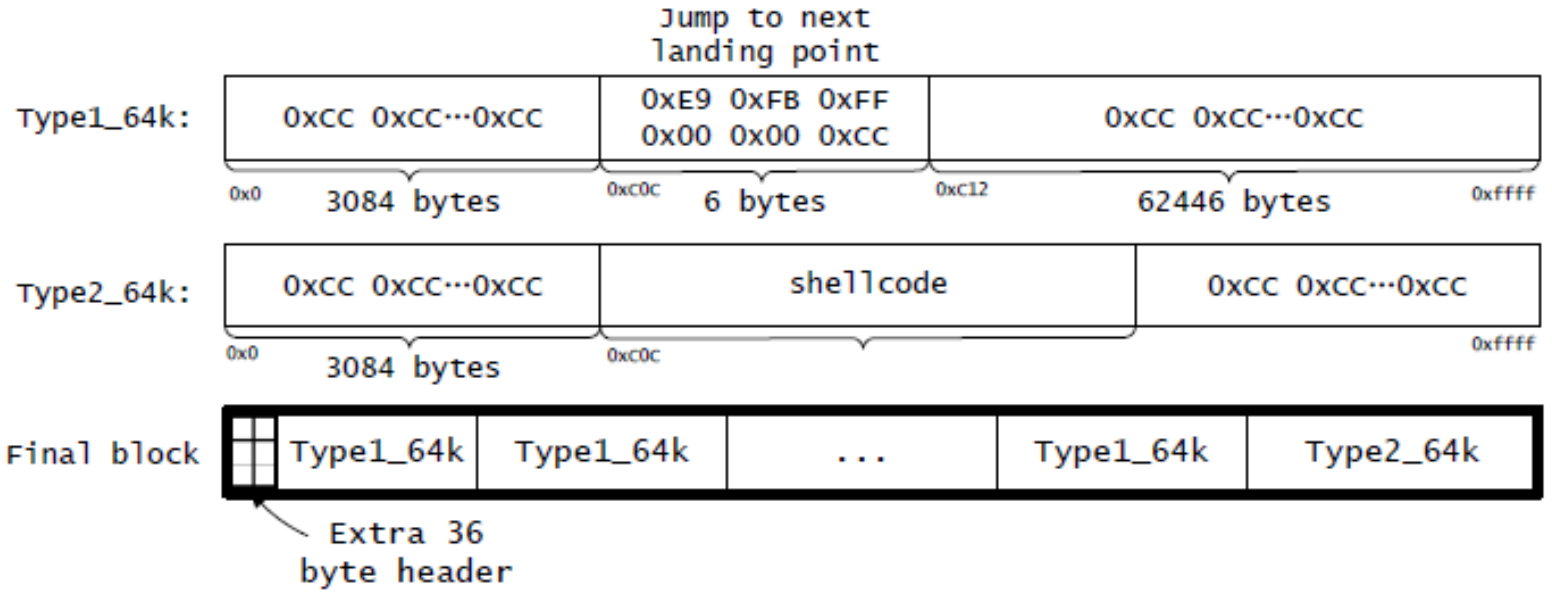


NSA(sledge) ≈ 7 / 512K ≈ **0%**



The Structure of TypeA Attack (Detail)

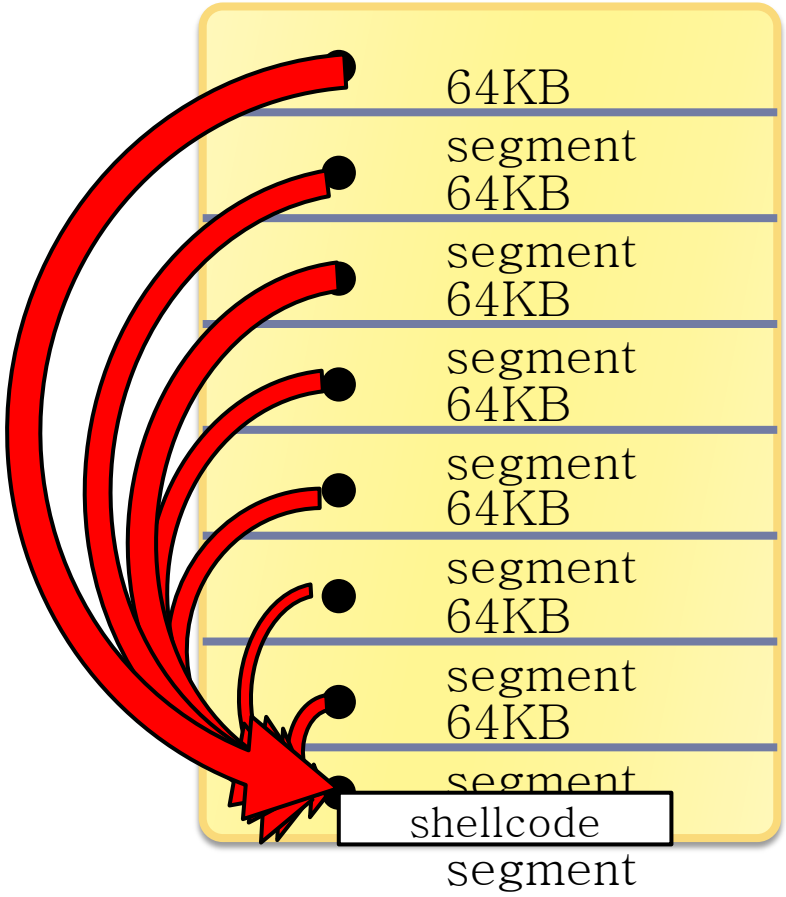
- A sample JavaScript code creating Type A heap objects



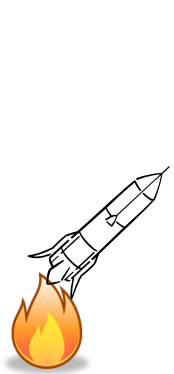
  - The size of each heap memory block is 1Mbytes.
  - Try to allocate 200 heap memory blocks to cover address `0x0c0c0c0c`

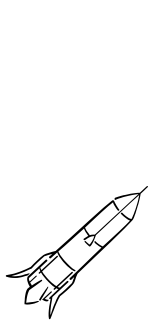
Basic Structure of TypeB (Jumping together)

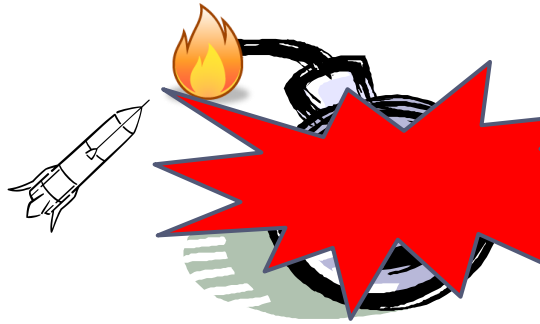
alignment size = 64Kbytes
blocksize = 512Kbytes



NSA(sledge) ≈ 7 / 512K ≈ **0%**

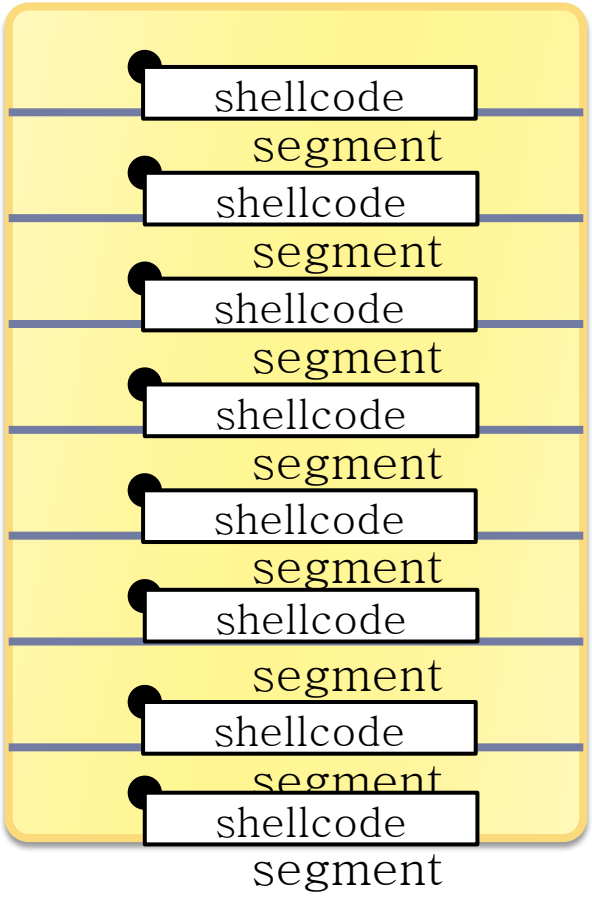




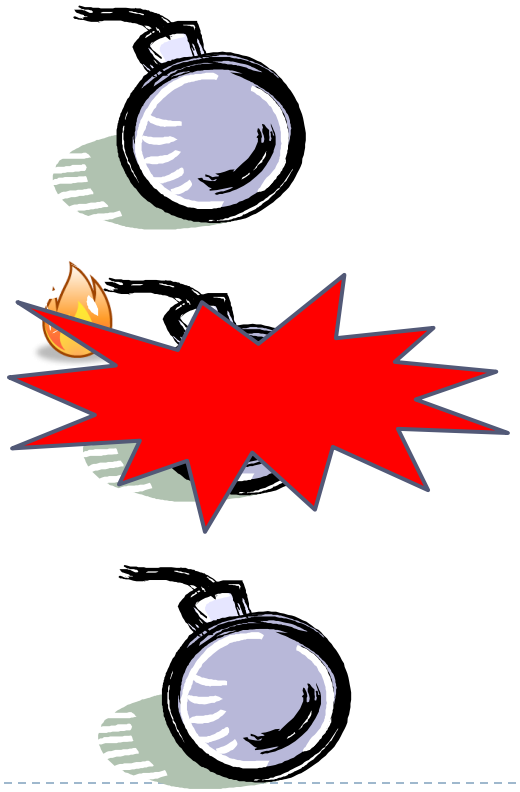


Basic Structure of TypeC (Returning Home)

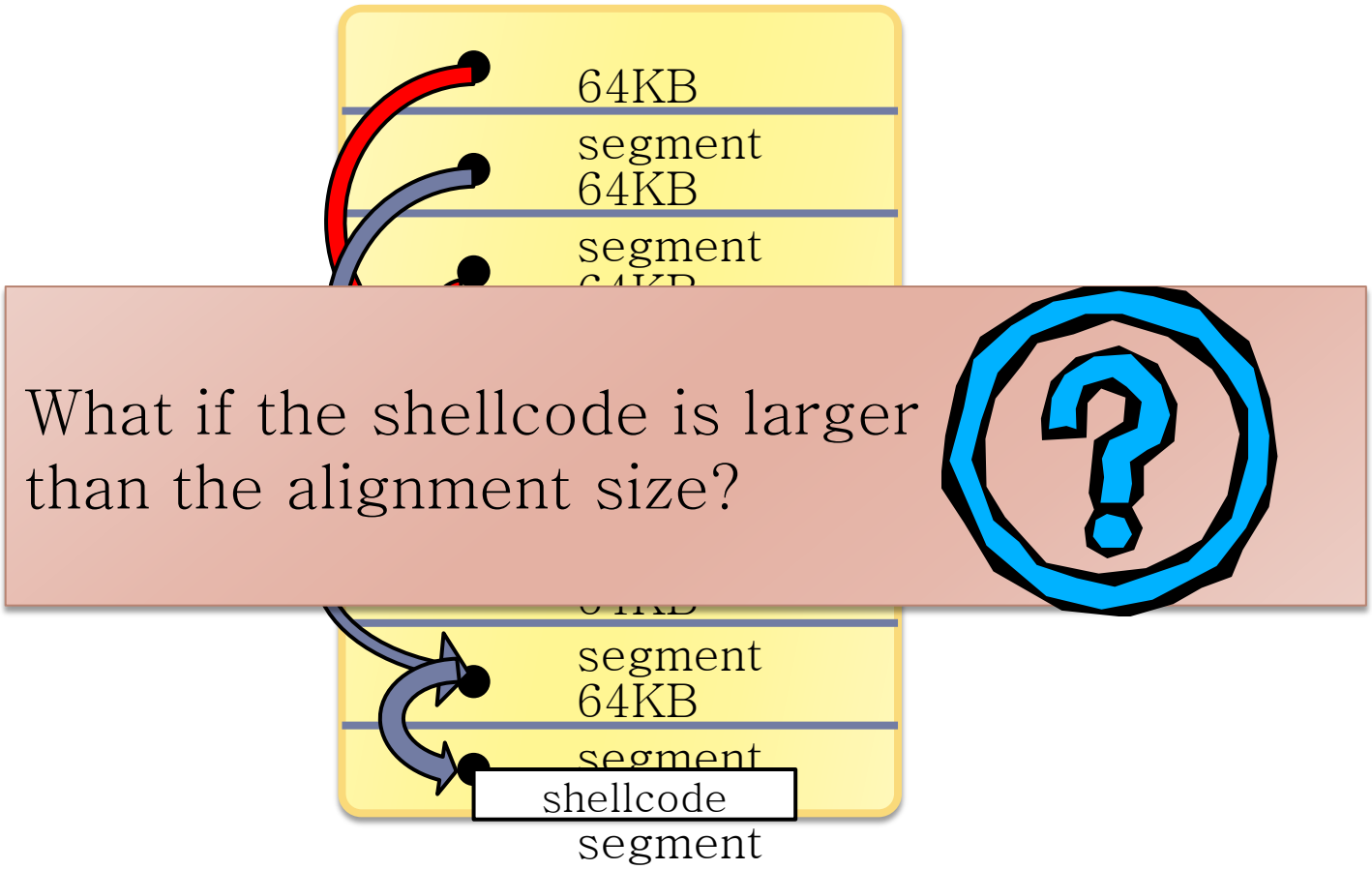
alignment size = 64Kbytes
blocksize = 512Kbytes



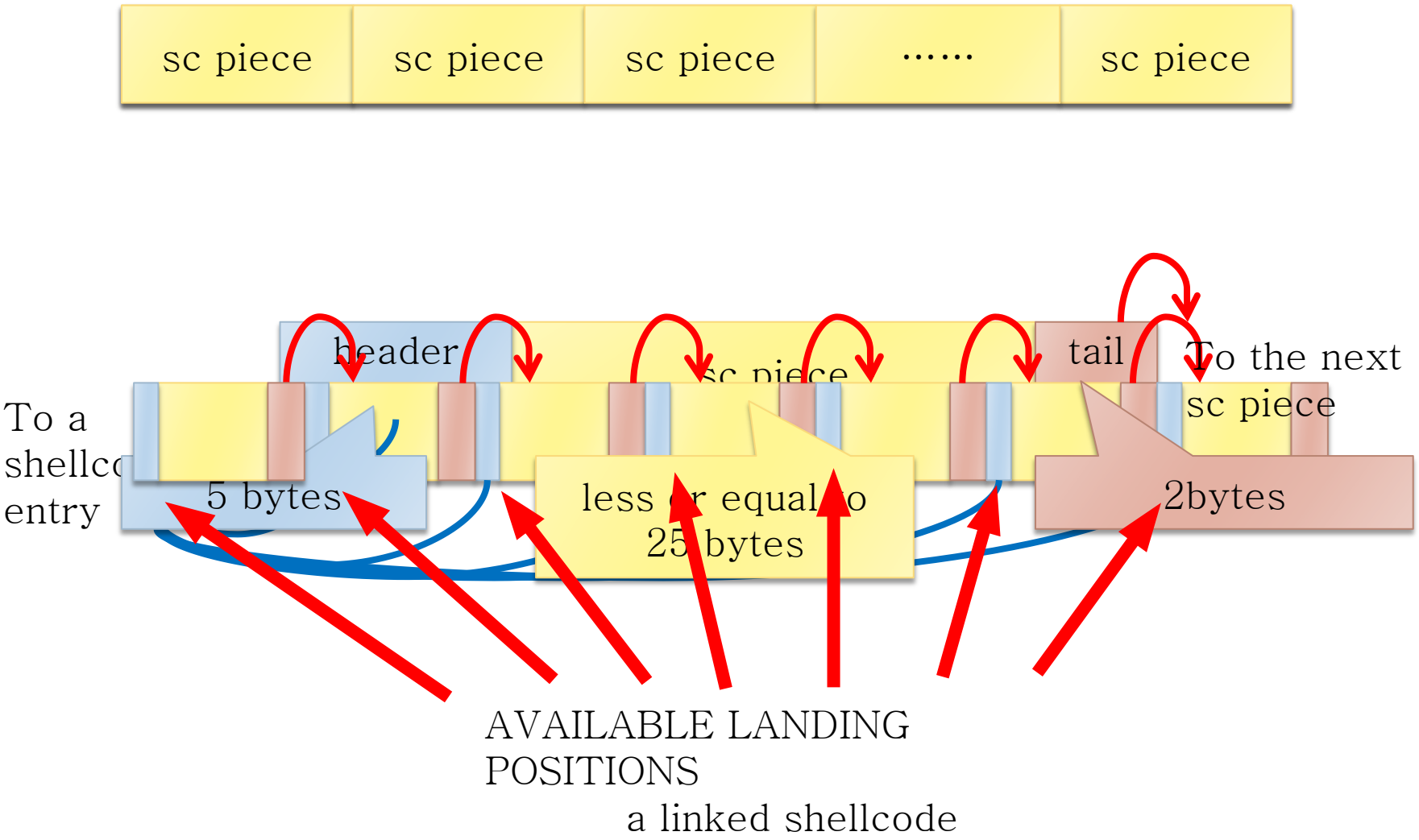
NSA ≈ sizeof(shellcode) / 512K ≈ **0%**



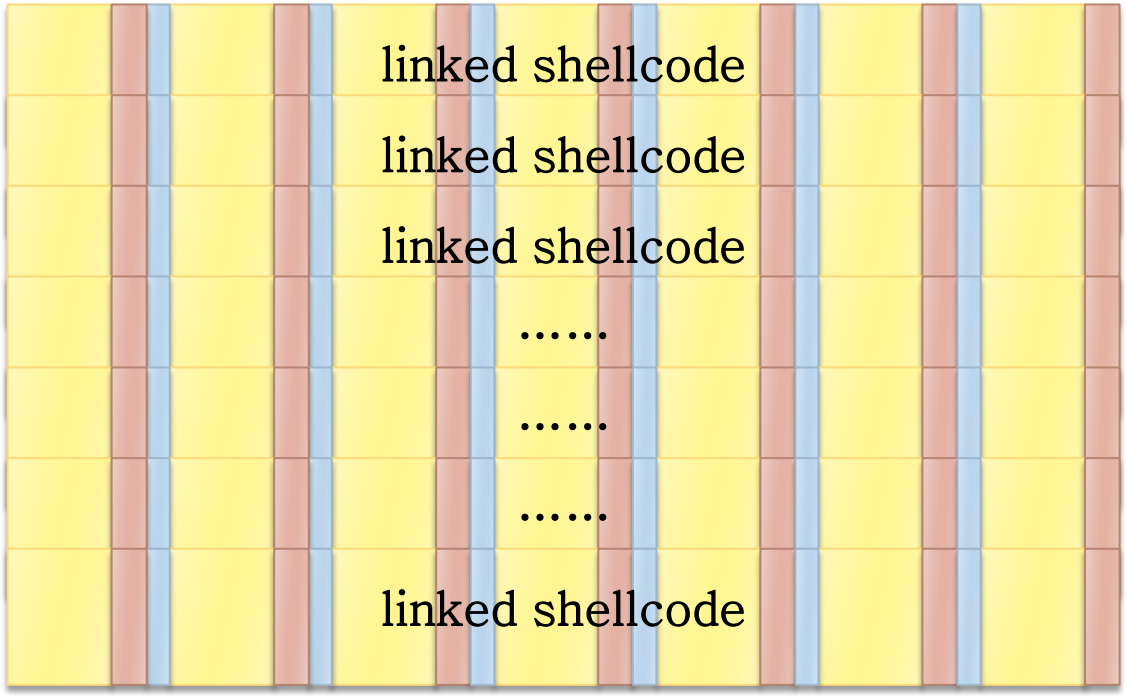
A Mix Type of TypeA, B, C



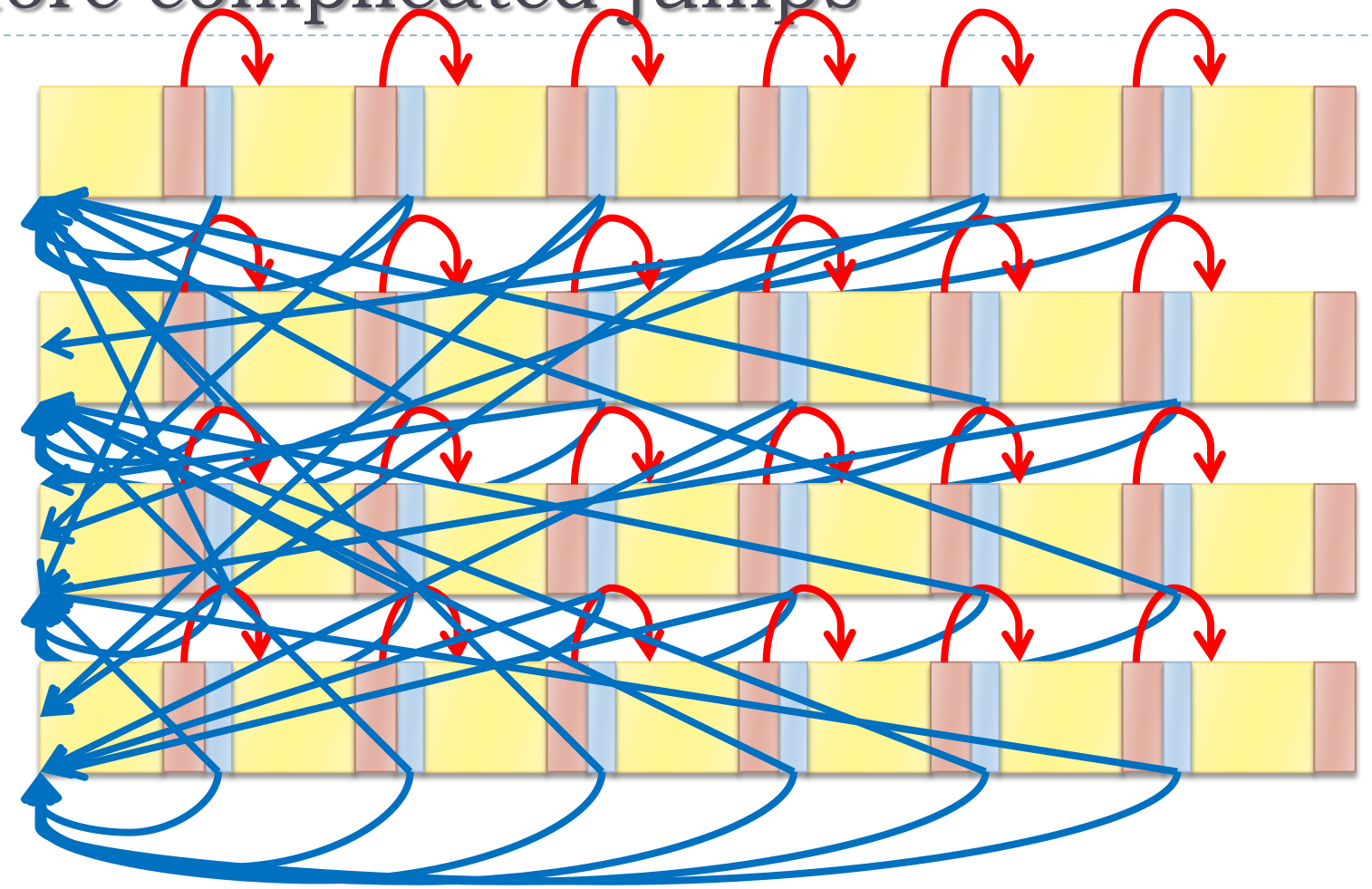
The TypeD Attack (alignment size = 32bytes)



The TypeD Attack (alignment size = 32bytes)



More complicated jumps



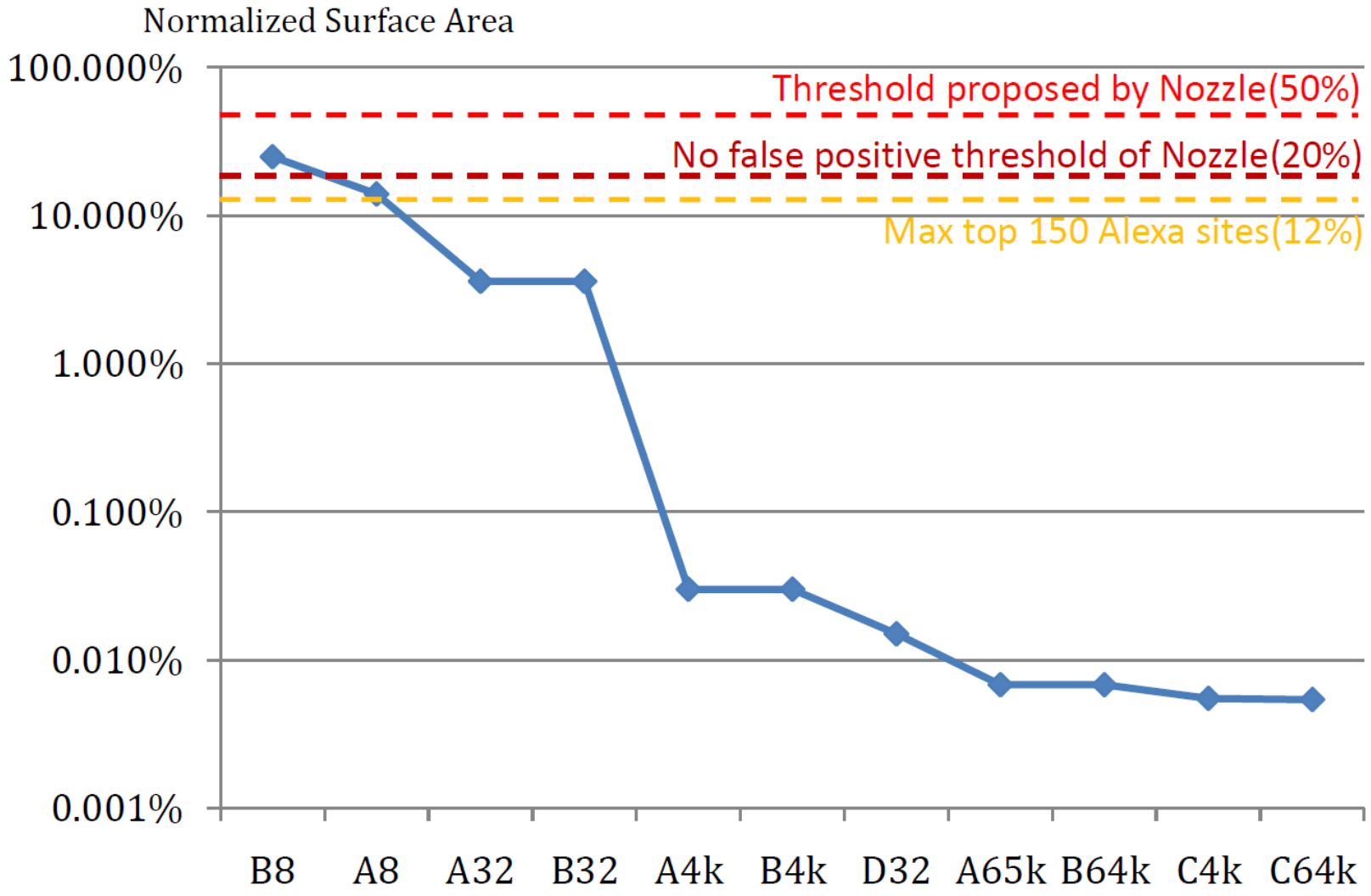
Relationship between Attack Type and Alignment Size

- Summarize

Alignment size	Type A	Type B	Type C	Type D
64 kbytes	√	√	√	√
32 bytes	√	√	X	√
8 bytes	√	√	X	X
4 bytes	√	X	X	X

- Small amount of instructions keep high success rate with help of memory allocation granularity.

Evaluation of Heap Taichi Attack Surface Area Calculation



Detecting Heap Taichi Attacks

Detecting Heap Taichi

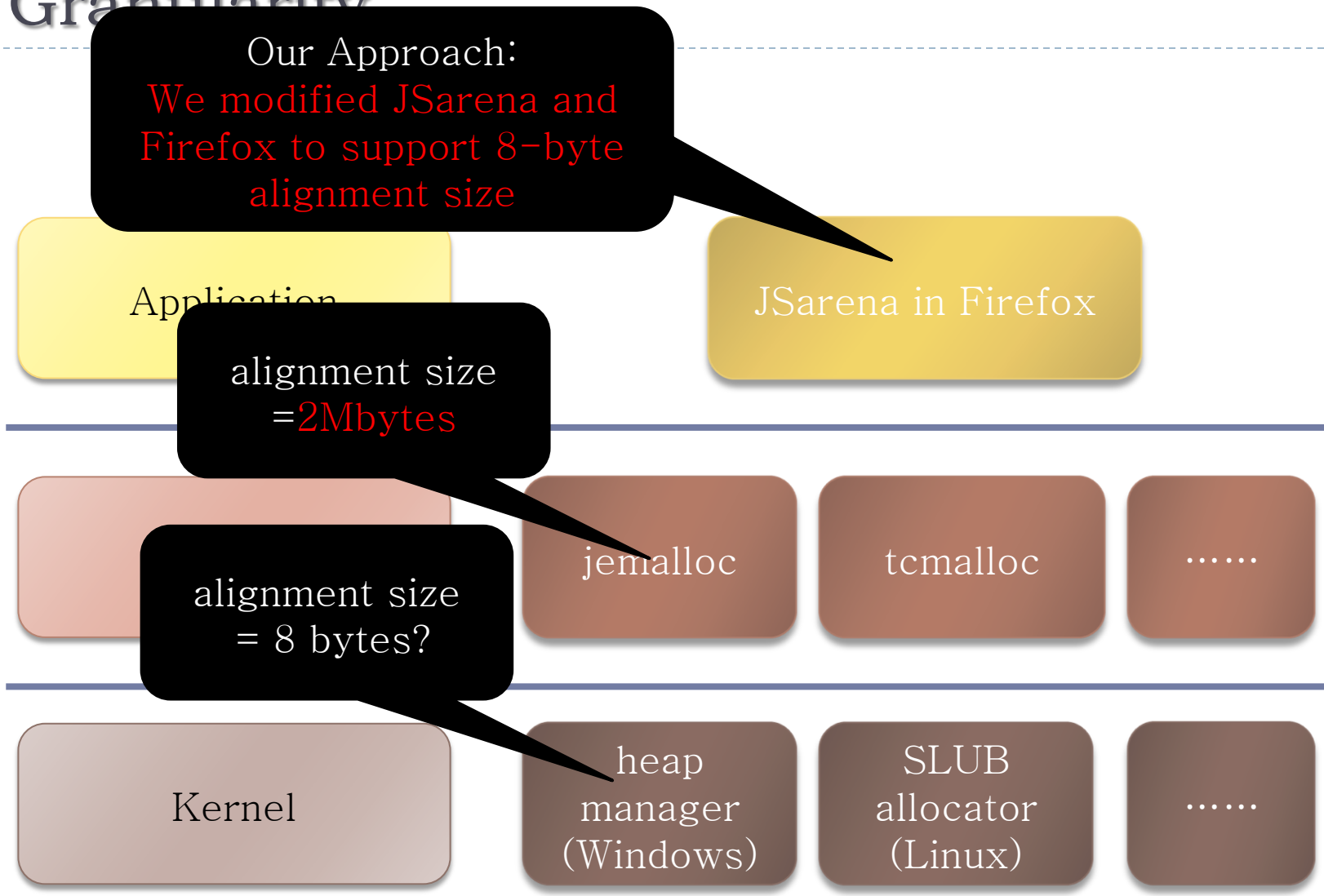
Alignment size	Type A	Type B	Type C	Type D
64 kbytes	√	√	√	√
32 bytes	√	√	X	√
8 bytes	√	√	X	X

1. Decrease the alignment size.
2. Improve the normalized surface area calculation.

Coar… granularity helps hide the shellcode and jump instructions.

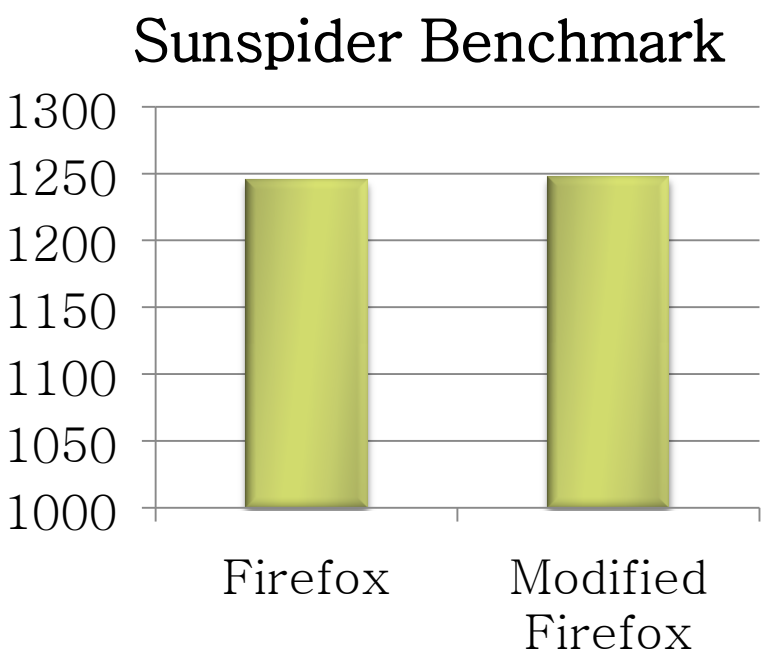
…using multiple copies of shellcode

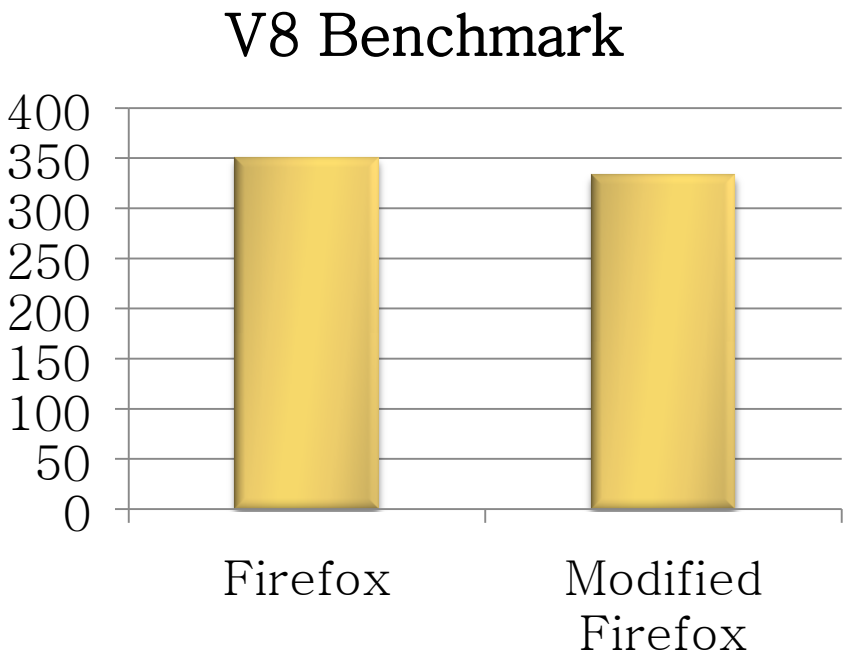
Difficulty in Lower the Memory Allocation Granularity



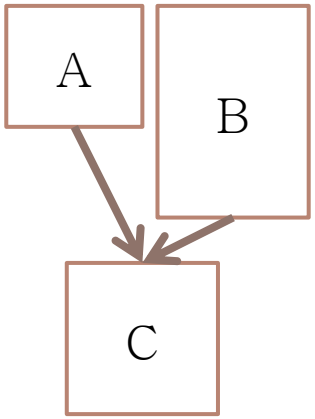
8-byte alignment is feasible

- The performance overhead is less than 5%

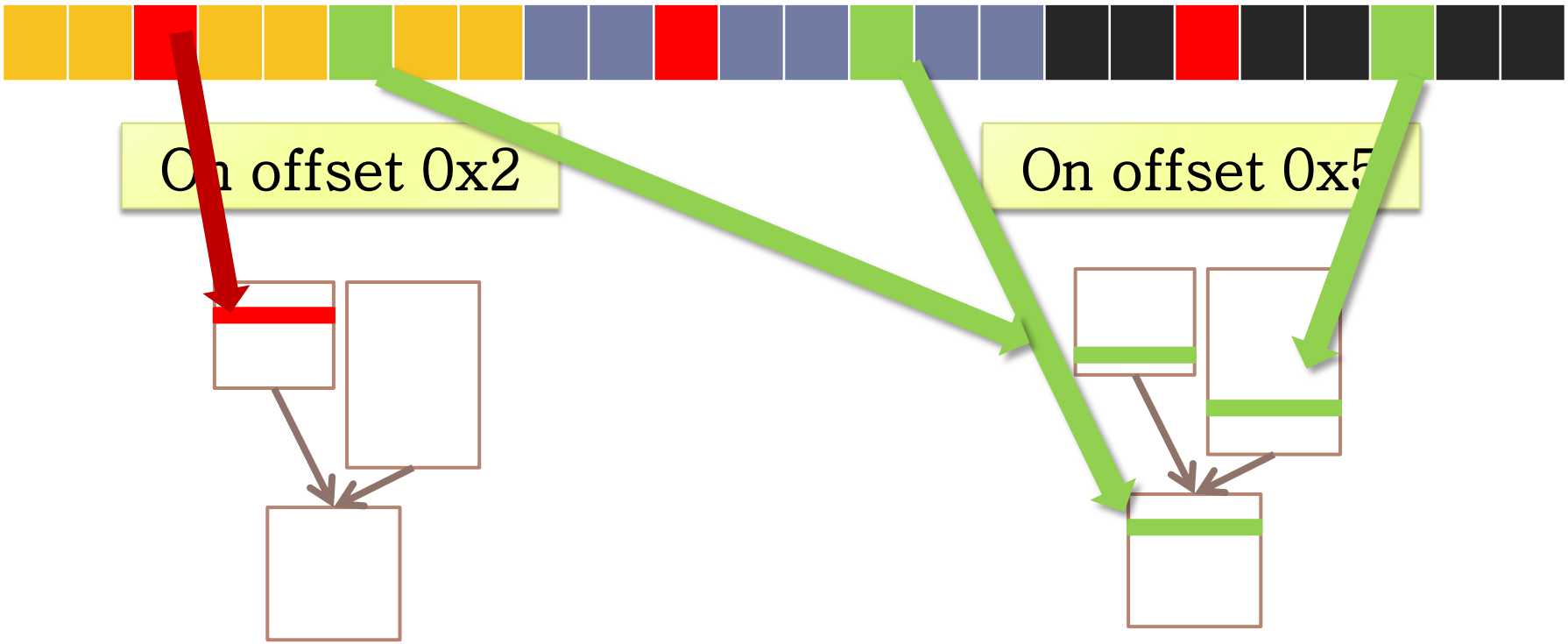




Enhance Nozzle
(Alignment size = 8 bytes)



Enhance Nozzle (Alignment size = 8 bytes)



Success rate on offset 0x2 = 1/3 = 33%

RELATED WORK

- Shellcode-based detection
  - Egele et, al. DIMVA'09
  - Sam Small et, al. CCS'09
  - Yingbo Song et,al. CCS'07
- Heap spraying with ASLR
  - The granularity is still 64KBytes
- Heap spraying with DEP
  - DEP can be bypassed
    - Peter Vreugdenhil, Pwn2Own
- Solid Heap Memory Allocator
  - DieHard, DieHarder
- Memory exploit detection and prevention

Conclusion

- We analysis the weakness of memory alignment size.
- We present Heap Taichi, a new heap-spraying skill utilizing the weakness of memory alignments.
- We present an enhanced nozzle algorithm. The new algorithm can recognize these Heap Taichi attacks cooperating with finer memory allocation granularity.


Thank you!