HITCON CTF 2020
Archangel Michael's Storage

Play with segment heap

Angelboy

angelboy@chroot.org @scwuaptx

Description

Environment

- Windows x64 on Windows Server 20H2
  - DEP
  - ASLR
  - CFG
- Private Heap
  - Independent memoy pool
- Segment Heap
  - reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\MichaelStorage.exe" /v FrontEndHeapDebugOptions /t REG_DWORD /d 0x8 /f

Description

Archangel Michael's Storage

- A simple datastorate
    - Allocate Storage
    - Set value to storage
    - Get value from storage
    - Destroy Storage

Description

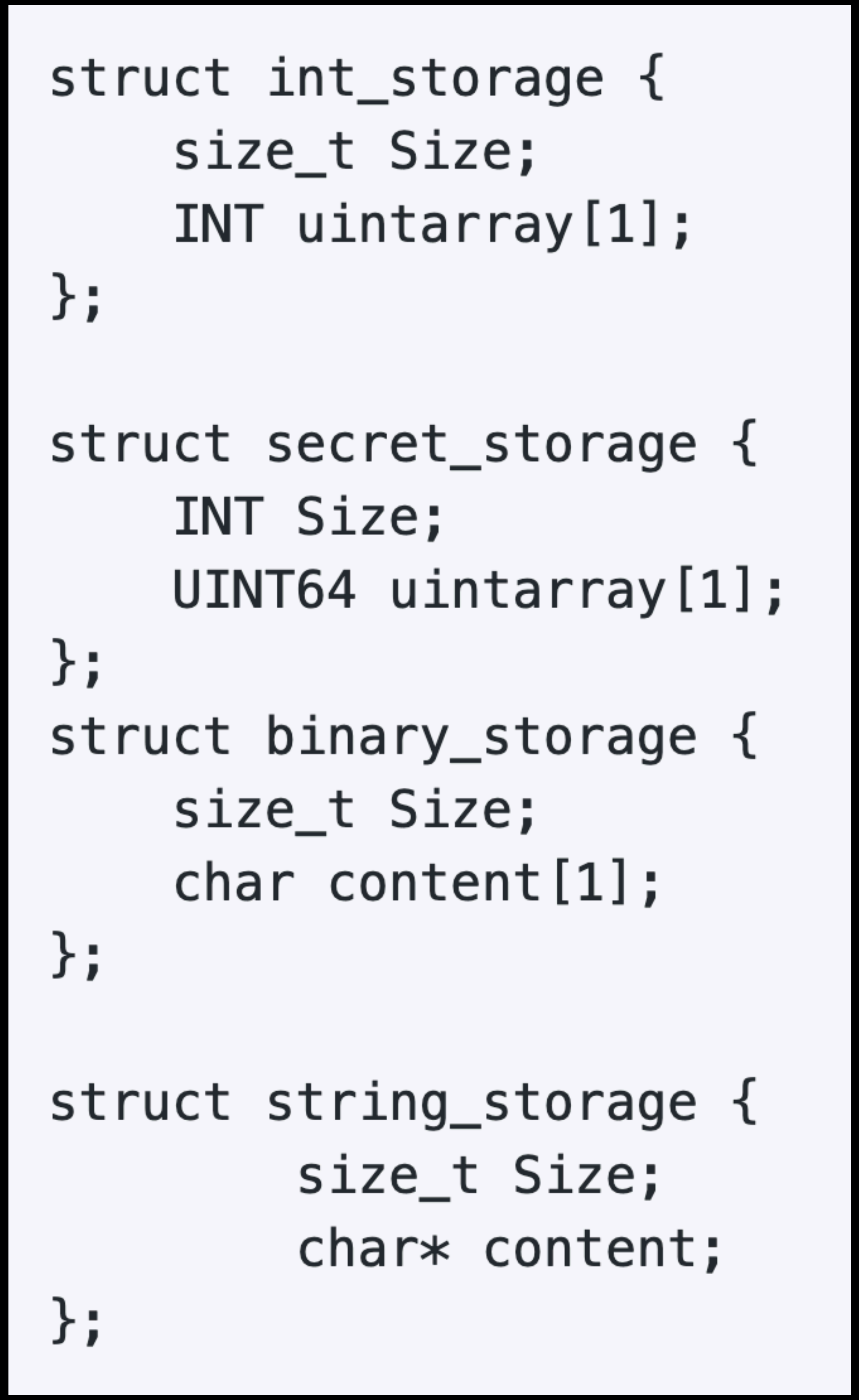
Archangel Michael's Storage

- A simple datastorate
  - Allocate Storage
    - allocate a specific type storage
      - Integer
      - Secret
      - Binary
      - String

Description

Archangel Michael's Storage

- Structure
  - The size of integer,secret and binary structure are variable.
    - The data are store in the structure.
  - The size of string is fixed and the data is additional memory block which will be allocated when you allocate the storage.



```
struct int_storage {
    size_t Size;
    INT uintarray[1];
};

struct secret_storage {
    INT Size;
    UINT64 uintarray[1];
};
struct binary_storage {
    size_t Size;
    char content[1];
};

struct string_storage {
        size_t Size;
        char* content;
};
```

Description

Archangel Michael's Storage

- A simple datastorate
    - Set value to storage
        - Set a value to a storage
    - Get
        - Get a value from a storage
        - Only for string storage

Description

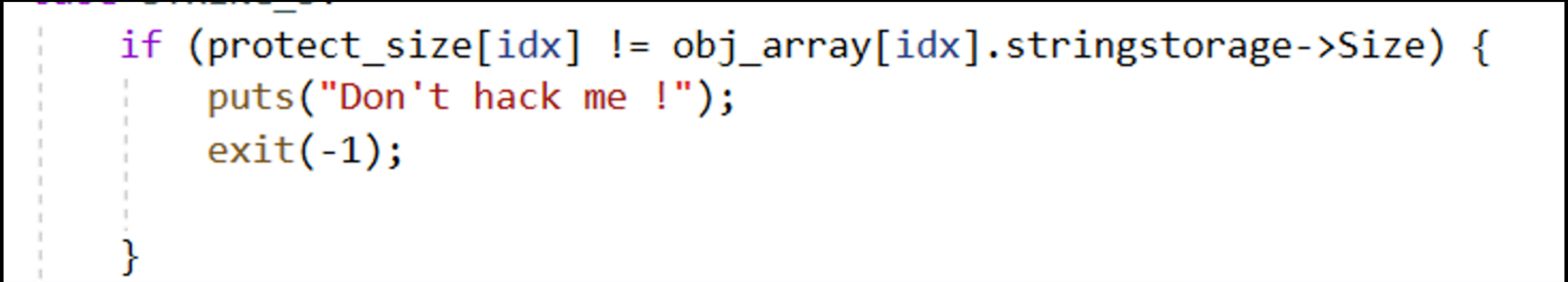
Archangel Michael's Storage

- A simple datastorate
    - Destory Storage
        - destory a storage

Description

Archangel Michael's Storage

- A simple datastorate
    - Security check
        - If the size of storage is changed after allocated, it will be considered illegal.
        - It will be check when you set value or get value.



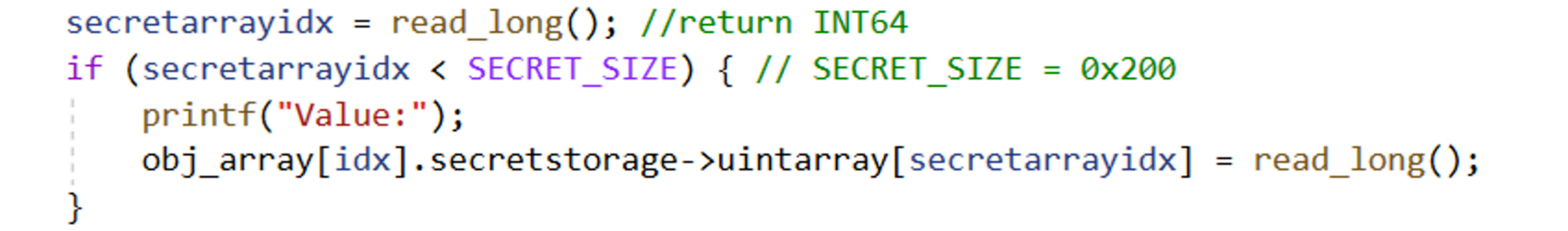
```
    if (protect_size[idx] != obj_array[idx].stringstorage->Size) {
        puts("Don't hack me !");
        exit(-1);

    }
```

Vulnerability

Archangel Michael's Storage

- Out of bound write
  - It does not check negtive index when you set a value in the secret storage. It will lead to out of bound write. You can write int64 data to previous memory block.



```
secretarrayidx = read_long(); //return INT64
if (secretarrayidx < SECRET_SIZE) { // SECRET_SIZE = 0x200
    printf("Value:");
    obj_array[idx].secretstorage->uintarray[secretarrayidx] = read_long();
}
```

Exploitation

Plan

- It looks very very easy !
  - We can use oob to overwrite string pointer with anything !
    - But ….
      - We don’t know any address…

Exploitation

Plan

- It looks very very easy !
  - We can use oob to overwrite string pointer with anything !
    - But ….
      - We don’t know any address…
    - So we need do leak first !
      - Create overlap chunk is easy way !

Exploitation

Create overlap chunk

- How to Create overlap chunk ?
  - Because it use private heap, we can easy use the oob write to write the metadata of the segment.
  - There are many idea that you easily think of:
    - Corrupt LFH bitmap
    - Abusing VS chunk header

Exploitation

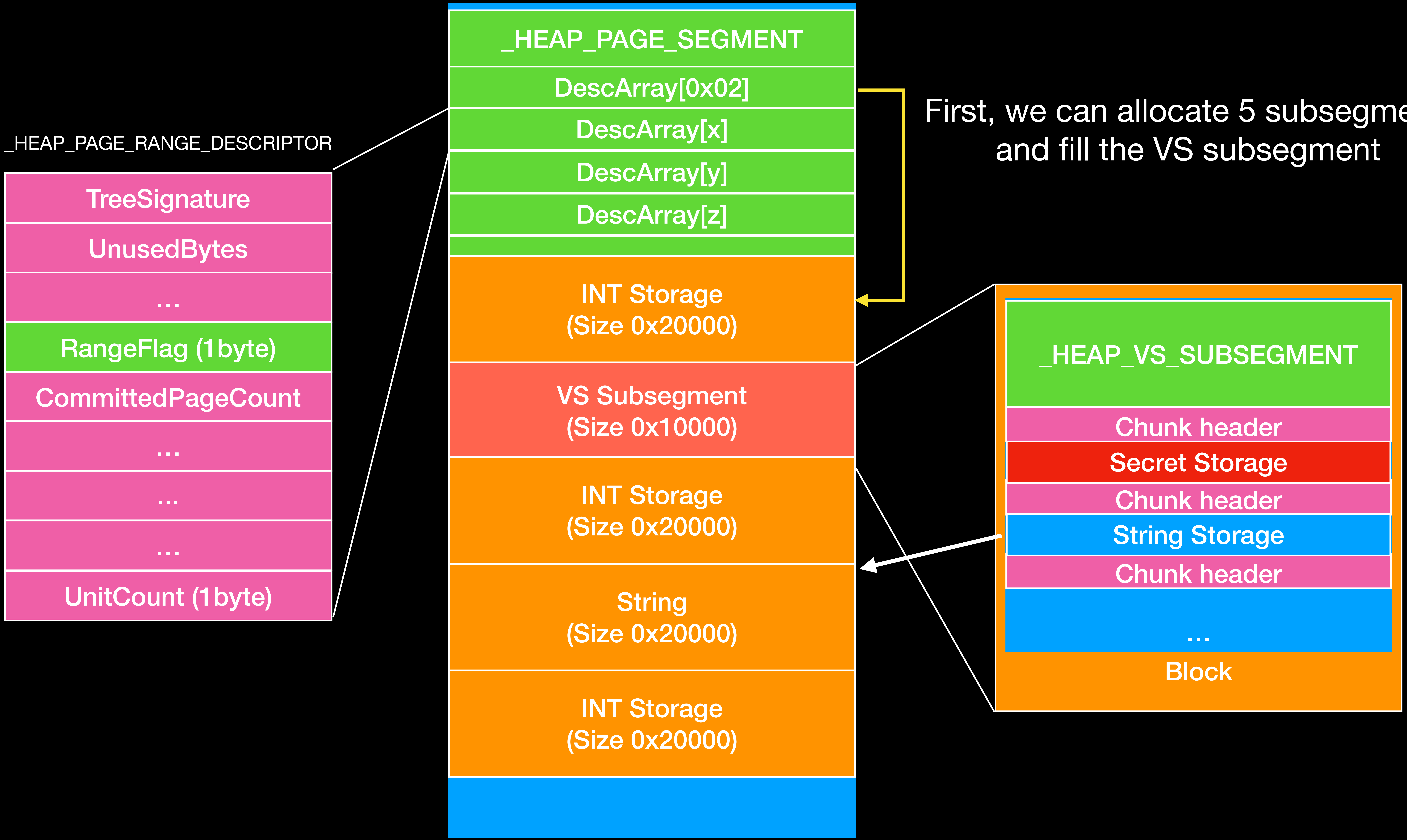
Create overlap chunk

- How to Create overlap chunk ?
  - Because it use private heap, we can easy use the oob write to write the metadata of the segment.
  - But there are many problems you will encounter
    - Corrupt LFH bitmap
      - Randomness of LFH chunk
    - Abusing VS chunk header
      - Chunk header encoding

Exploitation

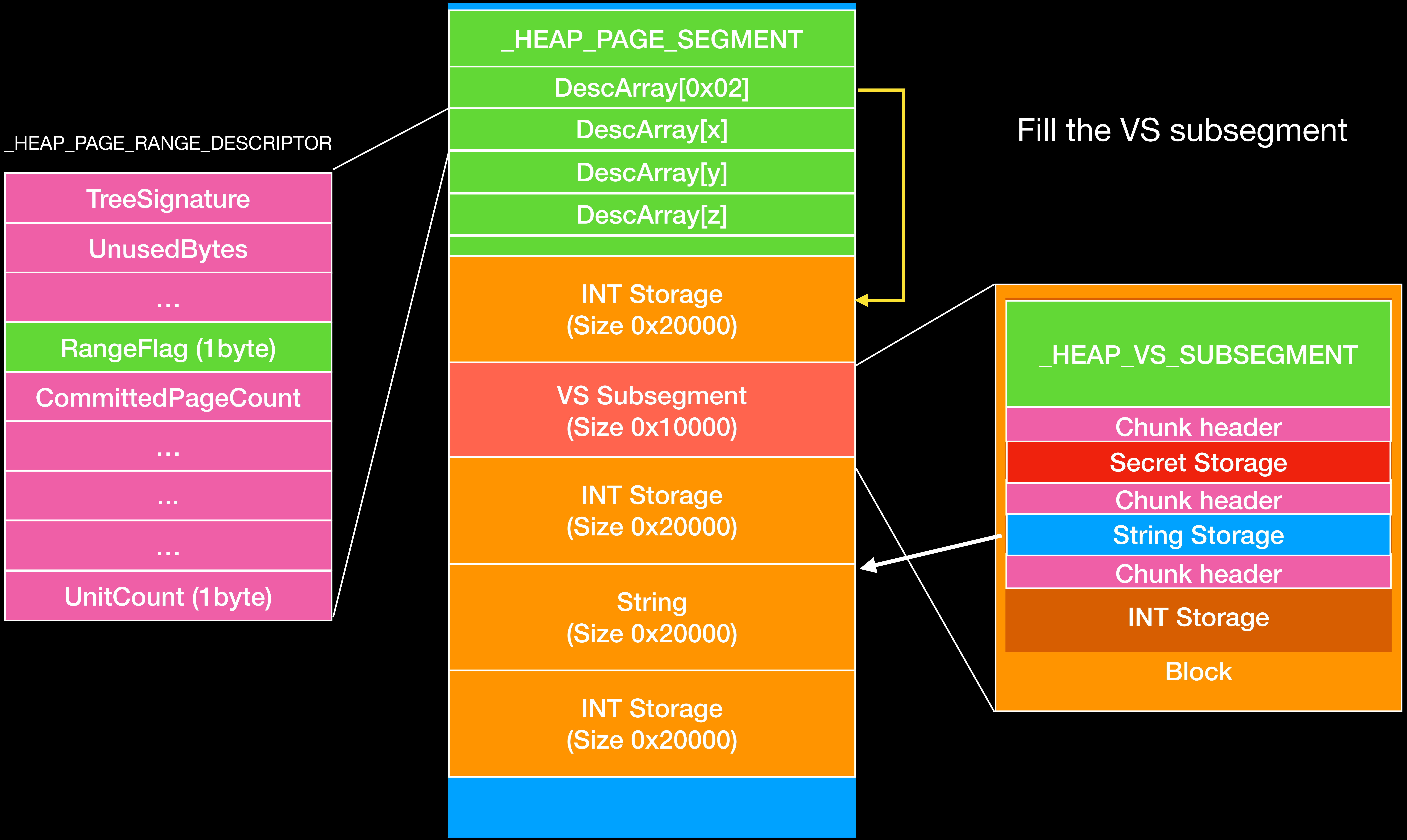
Create overlap chunk

- Corrupt meta data in segment allocator
  - Our target is _HEAP_PAGE_RANGE_DESCRIPTOR.
  - We can overwrite the _HEAP_PAGE_RANGE_DESCRIPTOR->UnitCount to make a large subsegment and free it.
  - It will release the next subsegment which is being used. And then create it again we will get overlap chunk.

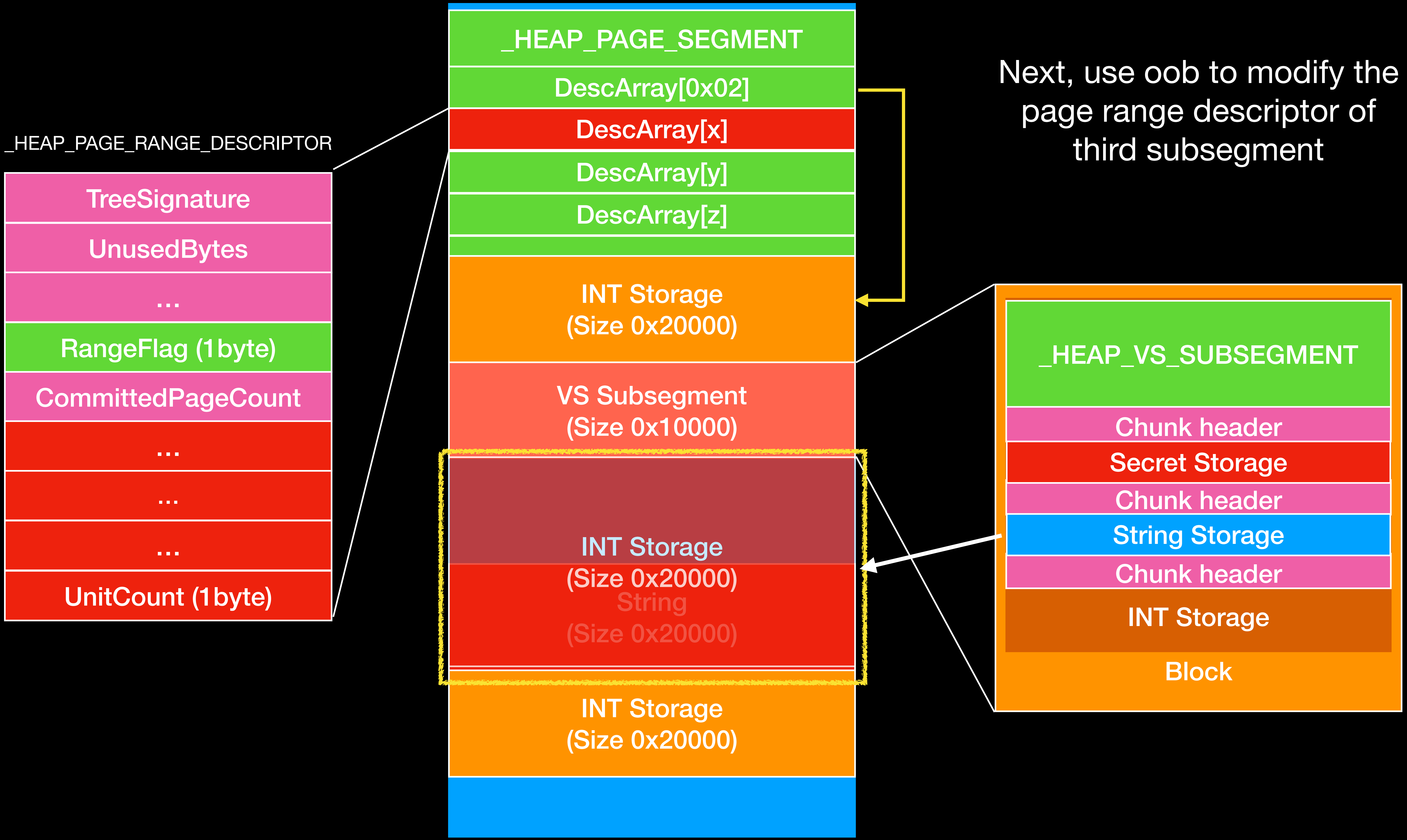


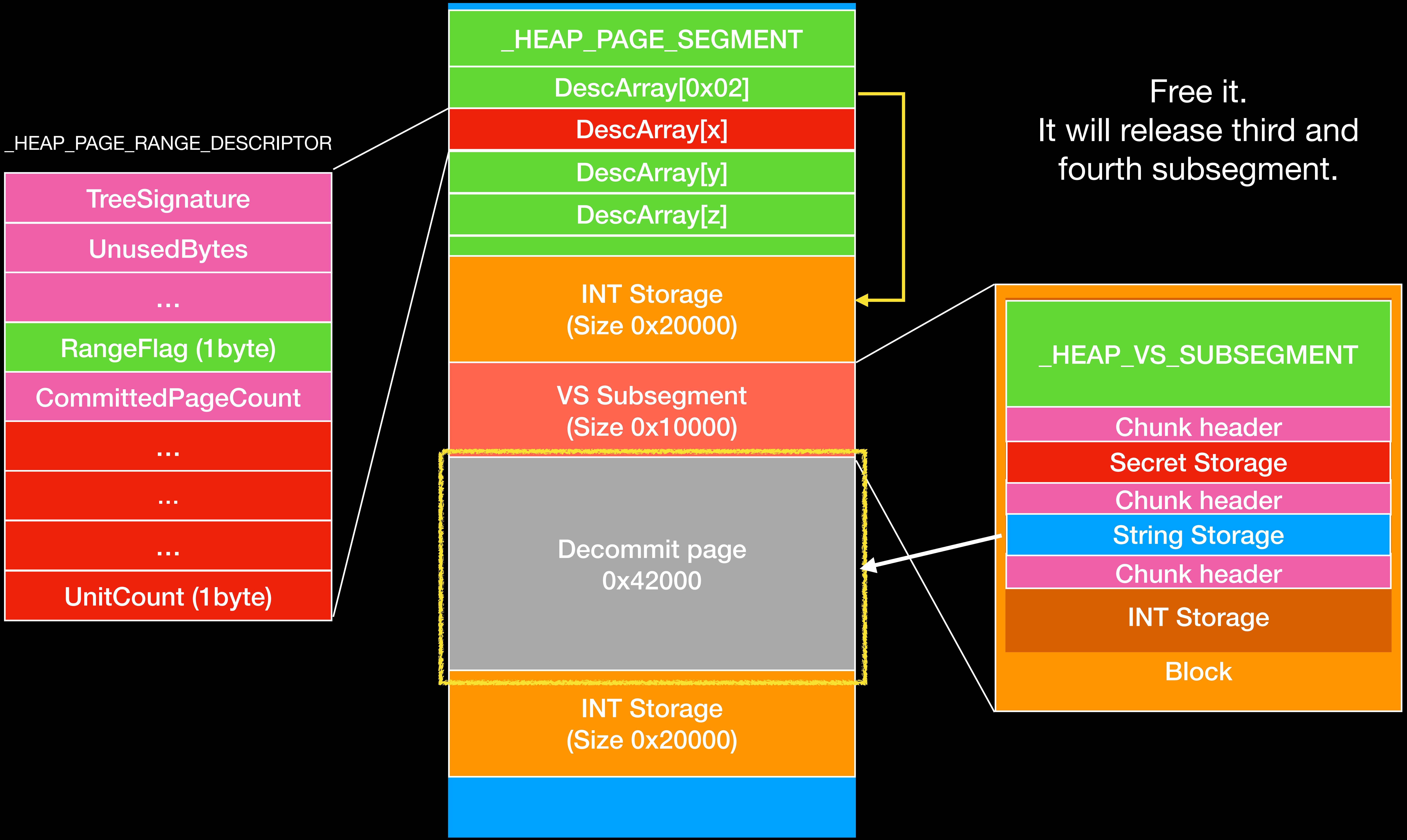
First, we can allocate 5 subsegment and fill the VS subsegment

Fill the VS subsegment



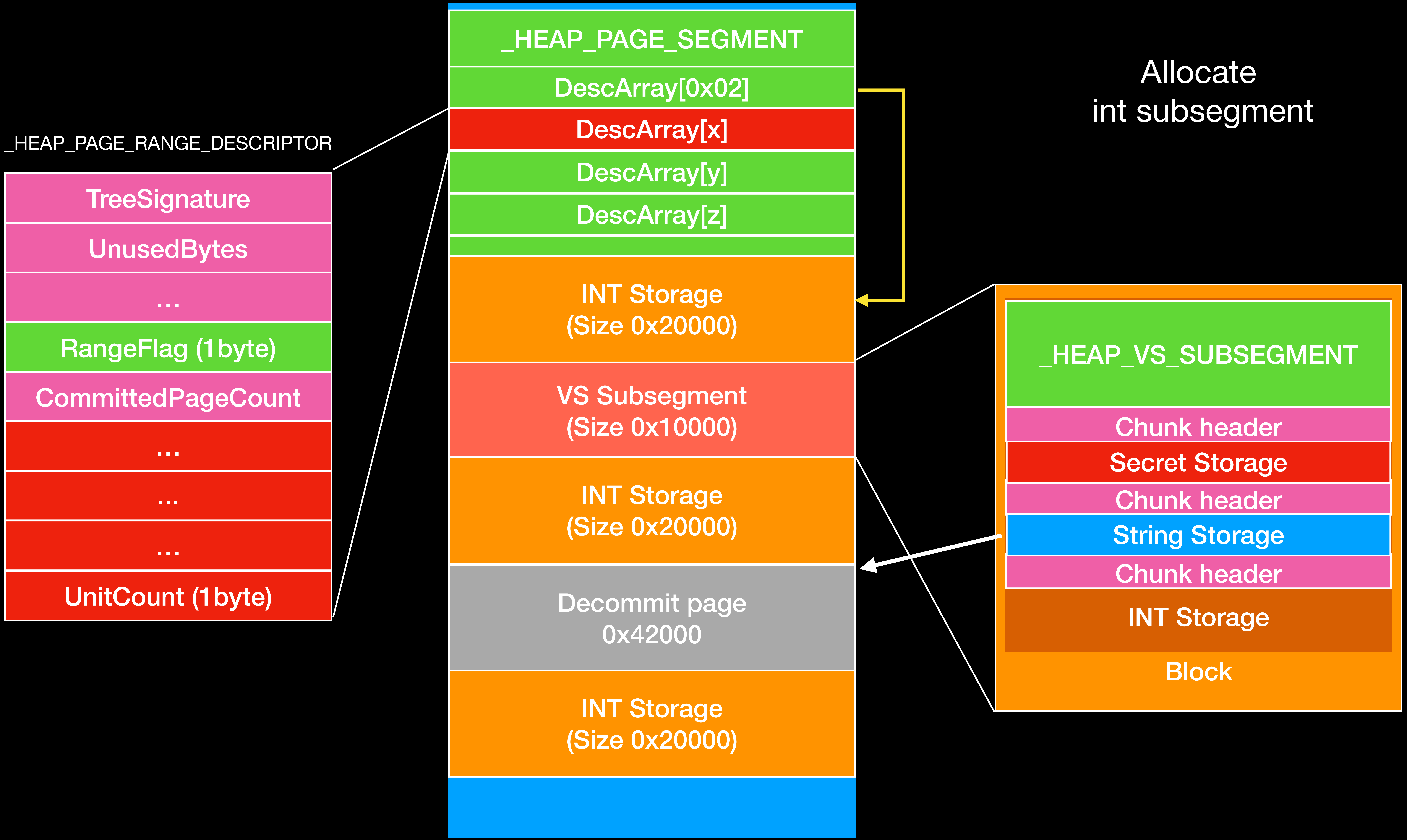
Next, use oob to modify the page range descriptor of third subsegment



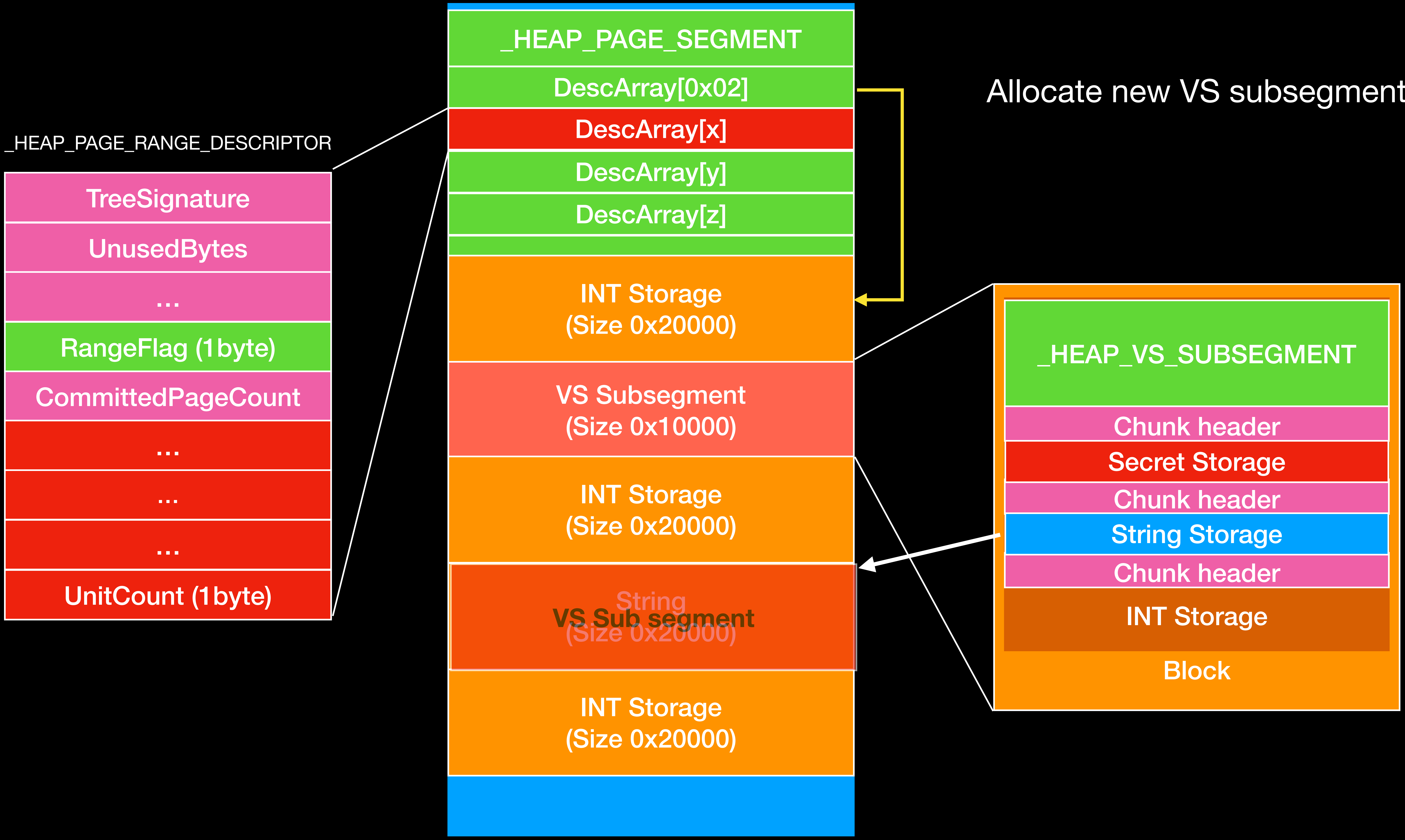


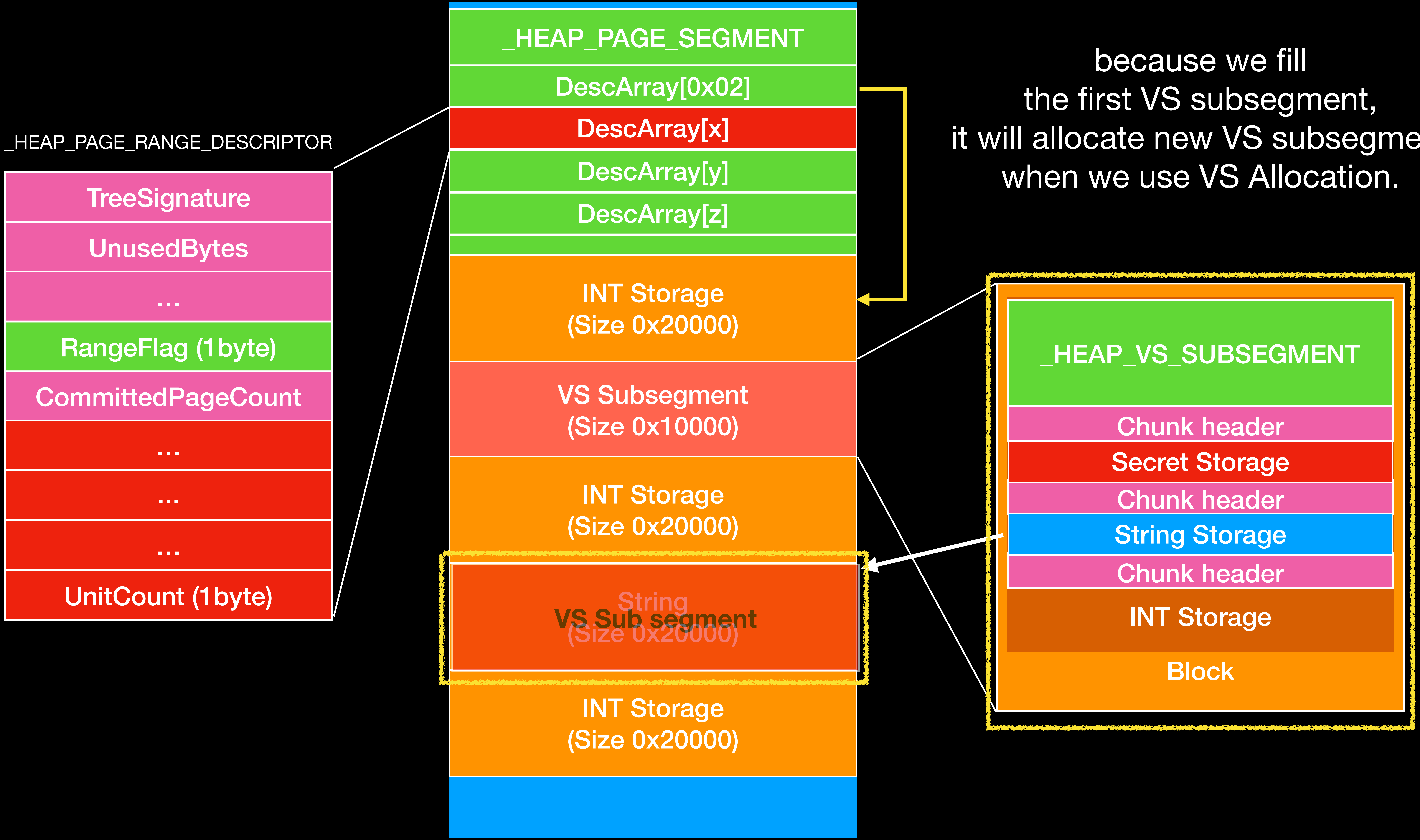
Free it.
It will release third and
fourth subsegment.

Allocate int subsegment



Allocate new VS subsegment



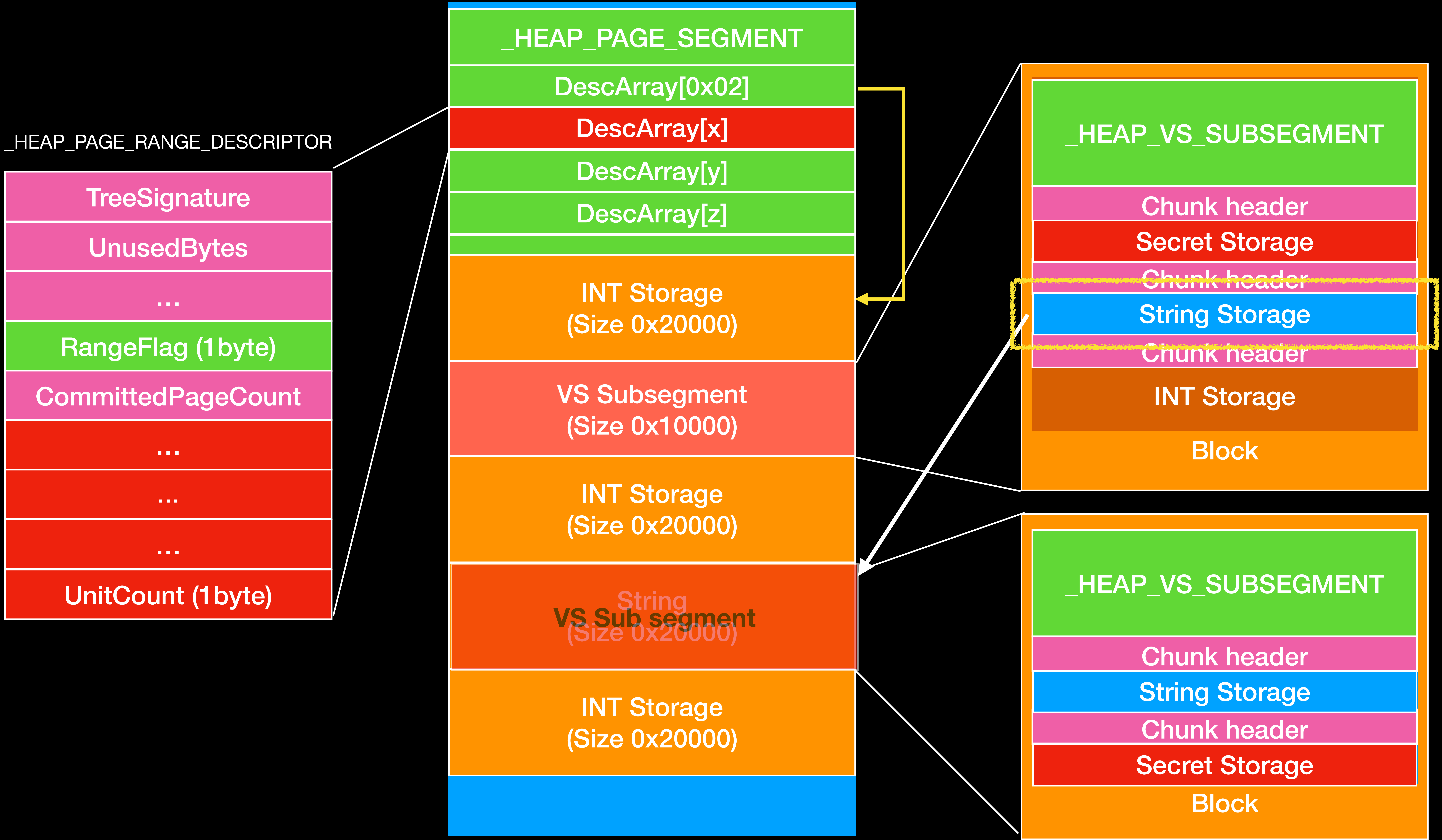


because we fill
the first VS subsegment,
it will allocate new VS subsegment
when we use VS Allocation.

Exploitation

Create overlap chunk

- Now we can allocate new string storage structure in the new VS subsegment.
- We have a overlap chunk and we can use the first string storage to leak something. We also can use secret storage to avoid null byte terminate.
  - We can use it to leak heap address



_HEAP_PAGE_RANGE_DESCRIPTOR

TreeSignature

UnusedBytes

...

RangeFlag (1byte)

CommittedPageCount

...

...

...

UnitCount (1byte)

_HEAP_PAGE_SEGMENT

DescArray[0x02]

DescArray[x]

DescArray[y]

DescArray[z]

INT Storage
(Size 0x20000)

VS Subsegment
(Size 0x10000)

INT Storage
(Size 0x20000)

String
(Size 0x20000)

VS Sub segment

INT Storage
(Size 0x20000)

_HEAP_VS_SUBSEGMENT

Chunk header

Secret Storage

Chunk header

String Storage

Chunk header

INT Storage

Block

_HEAP_VS_SUBSEGMENT

Chunk header

String Storage

Chunk header

Secret Storage

Block

Exploitation

Arbitrary memory reading and writing

- After we create overlap chunk, we can do arbitrary memory reading and writing by using string storage and secret storage.
    - After we can do arbitrary memory reading, we can use it to leak _HEAP_VS_SUBSEGMENT->Flink to get _SEGMENT_HEAP
    - We can leak ntdll from _SEGMENT_HEAP->LfhContext->AffinityModArray

Exploitation

Control RIP

- After we have arbitrary memory writing we can overwrite return address on stack with ROP

Exploitation

Another solution

- From Balsn
  - Corrupt VS subsegment header